НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

"На правах рукопису"
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА

“___”_____2023р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності 122 “Комп’ютерні науки”

на тему Модель генерації клітинних автоматів

Виконав: студент 2 курсу, групи ТР-21мп
Рисак Денис Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н. Ольга ЗАЛЕВСЬКА _____
(посада, вчене звання, науковий ступінь, ім’я ПРІЗВИЩЕ) (підпис)

Рецензент доцент, д.т.н. Євген ГАВРИЛКО _____
(посада, вчене звання, науковий ступінь, ім’я) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2023

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
Кафедри ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ
Рівень вищої освіти другий, (магістерський)
За освітньою програмою "Цифрові технології в енергетиці"
Спеціальності 122 Комп’ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
«____» _____ 2023р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Рисак Денис Сергійович
(прізвище, ім’я, по батькові)

1. Тема дисертації Модель генерації клітинних автоматів

Науковий керівник Залевська Ольга Валеріївна к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “06”листопада 2022 року №5152-с
2. Строк подання студентом дисертації 18 грудня 2023року
3. Вихідні дані до роботи готова система для роботи та аналізу клітинних автоматів

4. Перелік питань, які потрібно розробити моделі клітинних автоматів, система записників, база даних, інтерфейс системи

5. Орієнтований перелік ілюстративного матеріалу
Постановка задачі, Засоби розробки, Збір даних, База даних, Архітектура системи, Інтерфейс, Модульні тести, Висновки.

6. Орієнтований перелік публікацій

7. Дата видачі завдання «24» жовтня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської роботи	Строки виконання етапів магістерської дисертації	Примітки
1.	Отримати завдання	10.06.2023	
2.	Практика	01.09.2023 – 26.10.2023	
3.	Збір інформації	15.09.2023 – 10.10.2023	
4.	Аналіз вимог завдання, розробка методів і засобів розв'язання оставленої задачі	10.10.2023	
5.	Розробка та тестування програмного продукту	11.10.2023- 25.10.2023	
6.	Виконання розділів дисертації (практична частина, загальні висновки, список джерел)	26.10.2023	
7.	Написання основних розділів автореферату	27.10.2023- 12.11.2023	
8.	Перевірка дисертації науковим керівником		
9.	Подання в електронному вигляді роботи та анотації до неї на перевірку нормоконтролера та плагіат (UNICHECK)		
10.	Надання документів на засідання кафедру		
11.	Предзахист магістерської дисертації та допуск до захисту дисертації	04.12.2023	
12.	Подання магістерської дисертації рецензенту. Отримання рецензії.		
13.	Подання пакету документів по магістерської дисертаціїта супровідних до неї документів до захисту в ЕК		
14.	Захист магістерської дисертації		

Студент _____ Денис РИСАК
(підпис) (ім'я ПРІЗВИЩЕ)

Науковий керівник _____ Ольга ЗАЛЕВСЬКА
(підпис) (ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається зі вступу, чотирьох розділів, висновку, списку використаних джерел; містить 93 сторінок, 19 рисунки, 35 таблиці. Список

Метою роботи є створення моделі генерації клітинних автоматів, дослідження їх алгоритмів, візуалізації процесу розвитку моделі та розробка системи для задання початкових умов.

Для досягнення мети було поставлено наступні задачі:

- провести аналіз існуючих алгоритмів генерації клітинних автоматів, що описують динамічні системи;
- встановити переваги та недоліки існуючого програмного забезпечення для візуалізації клітинних автоматів;
- розробити програмне забезпечення для візуалізації процесу генерації моделі клітинного автомату, що позбавлена недоліків та надає можливість користувачу контролювати процес розвитку та задавати початкові умови;
- побудувати модель поширення вірусного захворювання використовуючи створений застосунок. Згенеровану вибірку дослідити за допомогою регресійного аналізу. Регресійну модель вибірки дослідити на стійкість та адекватність.

Ключові слова: клітинний автомат, JavaScript, ExpressJS, ReactJS, NodeJS, PostgreSQL.

ABSTRACT

The explanatory note consists of an introduction, four sections, a conclusion, a list of used sources; it comprises 93 pages, 19 figures, 35 tables. The list of used sources includes 30 bibliographic entries.

The goal of the work is to create a model for the generation of cellular automata, investigate their algorithms, visualize the model's development process, and develop a system for setting initial conditions.

To achieve this goal, the following tasks were set:

- Conduct an analysis of existing algorithms for generating cellular automata that describe dynamic systems.
- Identify the advantages and disadvantages of existing software for visualizing cellular automata.
- Develop software for visualizing the process of generating a cellular automaton model that is free from drawbacks and allows the user to control the development process and set initial conditions.
- Build a model of the spread of a viral disease using the created application. Investigate the generated sample using regression analysis. Examine the regression model of the sample for robustness and adequacy.

Keywords: cellular automaton, JavaScript, ExpressJS, ReactJS, NodeJS, PostgreSQL.

ЗМІСТ

ВСТУП 6

1 МОДЕЛІ КЛІТИННИХ АВТОМАТІВ, ЩО ВІДОБРАЖАЮТЬ ПОВЕДІНКУ ДИНАМІЧНИХ СИСТЕМ 8

1.1 Моделі клітинних автоматів 8

1.2 Опис математичних методів 14

1.3 Висновки до розділу 21

2 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВІЗУАЛІЗАЦІЇ КЛІТИННИХ АВТОМАТІВ 22

2.1 Постановка задачі 23

2.2 Підсистеми програмного продукту 27

2.3 Засоби розробки 36

2.4 Мова програмування JavaScript 37

2.5 Локальный сервер разработки Vite 39

2.6 Бібліотека ReactJS 41

2.7 Фреймворк Node.js 42

2.8 Фреймворк Express.js 44

2.9 База даних PostgreSQL 47

2.10 Бібліотека Material UI 50

2.11 Об'єктно-реляційний мапер Sequelize 53

2.12 Стейт-менеджер Effector.js 55

2.13 Лінтер ESLint 56

2.14 Бібліотека Jsonwebtoken 57

2.15 Висновки до розділу 59

3 ВІЗУАЛІЗАЦІЯ ПРОЦЕСУ ПОШИРЕННЯ ВІРУСУ 60
3.1 Постановка задачі та побудова моделі 60
3.2 Статистичні дані вибірки 61
3.3 Встановлення взаємозв’язку за допомогою регресійного аналізу..... 62
3.4 Висновки до розділу 68
4 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ 69
4.1 Інсталяція програмного забезпечення 69
4.2 Вимоги до програмного забезпечення комп’ютера..... 69
4.3 Запуск системи 70
4.4 Діаграма прецедентів..... 70
4.5 Демонстрація функціоналу 73
4.6 Висновки до розділу 80
5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ 81
5.1 Опис ідеї стартап проекту 81
5.2 Технологічний аудит мети реалізованого додатку..... 83
5.3 Проведення аналізу запуску розробленого стартапу 84
5.4 Ринкова стратегія проекту..... 92
5.5 Створення маркетингового плану для стартап-проекту 95
5.6 Висновки до розділу 98
ВИСНОВКИ..... 99
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 100

ВСТУП

Модель генерації клітинних автоматів, або, як її частіше називають, клітинні автомати, представляє собою цікавий та потужний інструмент для вивчення, аналізу та моделювання різноманітних процесів, що відбуваються в природі та суспільстві. Ця концепція виникла в результаті поєднання інтересів математиків, фізиків, інженерів та біологів, і з часом стала необхідним інструментом для вирішення різних завдань, починаючи від дослідження елементарних фізичних процесів та закінчуючи моделюванням складних соціальних явищ.

Історія клітинних автоматів веде свій початок ще з середини 20-го століття, коли математик та фізик Джон фон Нейман спроектував перший автомат, який здатний реплікувати самих себе, враховуючи правила, які визначають його функціонування. Цей революційний винахід мав величезний вплив на розвиток обчислювальної науки та математики, і з часом спонукав до створення безлічі різноманітних клітинних автоматів, кожен з яких спрямований на вирішення конкретних завдань.

Суть клітинних автоматів полягає в тому, що простий набір правил визначає, як клітинки, розташовані на решітці, взаємодіють одна з одною та як їх стани можуть змінюватися з плином часу. Це просте, але потужне поняття дозволяє моделювати широкий спектр явищ, починаючи від дифузії речовин у фізиці та закінчуючи поширенням інформації в соціальних мережах. Клітинні автомати стали не лише інструментом для розв'язання теоретичних завдань, але й знаходять широке застосування у практиці, включаючи моделювання процесів у природі, створення шифрів, аналіз біологічних систем та багато іншого.

У наш час клітинні автомати залишаються актуальними та цікавими для дослідження, оскільки вони дозволяють краще розуміти складні процеси навколишнього світу та розробляти нові методи розв'язання реальних завдань. Використовуючи клітинні автомати, науковці можуть аналізувати, як системи

еволюціонують з часом, як вони реагують на зміни в середовищі, та приймати інформовані рішення в різних галузях науки та технології.

У цьому контексті, розгляд клітинних автоматів відкриває перед нами захоплюючий світ абстрактного моделювання, де математика зливається з природознавством та інформатикою, надаючи новий погляд на розвиток науки та технології. В подальших розділах ми розглянемо докладніше, як саме працюють клітинні автомати, їх застосування в різних галузях та їх потенційну роль в майбутньому.

1 МОДЕЛІ КЛІТИННИХ АВТОМАТІВ, ЩО ВІДОБРАЖАЮТЬ ПОВЕДІНКУ ДИНАМІЧНИХ СИСТЕМ

Клітинні автомати, унікальний клас математичних моделей, які засновані на простих правилах еволюції станів клітин на сітці, надають можливості для розуміння та моделювання різноманітних систем. Однією з ключових особливостей клітинних автоматів є їхня здатність до відтворення складних інтеракцій та динамічних процесів, що робить їх надзвичайно корисними в різних областях, включаючи науку, техніку та мистецтво.

У цьому розділі ми детально розглянемо різні моделі клітинних автоматів та їхні особливості. Вивчення цих моделей дозволить нам краще розуміти, як клітинні автомати можуть бути застосовані у практиці, включаючи їхню роль у моделюванні процесів, розв'язанні складних завдань та вдосконаленні технологічних рішень. Далі представлено докладний огляд ключових моделей клітинних автоматів, що дозволить читачеві глибше вникнути в їхні можливості та потенційні застосування.

1.1 Моделі клітинних автоматів

Клітинні автомати (Cellular Automata) - це математична модель, яка складається з сітки клітин (як правило, у вигляді квадратних або гексагональних) і правил, які визначають, які зміни відбуваються в кожній клітині на основі стану її сусідів. Кожна клітина може мати обмежену кількість станів Клітини оновлюються одночасно у дискретних часових ітераціях.

Основні компоненти клітинного автомата:

- **Сітка клітин:** Це двовимірна або тривимірна структура, у якій розташовані клітини. Кожна клітина може мати свій власний стан.
- **Правила еволюції:** Це набір правил, які визначають, як змінюється стан кожної клітини в кожен момент часу в залежності від стану її сусідів. Правила можуть бути простими або складними, і вони визначають динаміку системи.
- **Початковий стан:** Початковий стан сітки клітин визначається в момент початку моделювання. Він може бути випадковим або визначатися конкретним способом.
- **Крок часу:** Це величина, яка визначає, який інтервал часу пройшов між кожним кроком моделювання. На кожному кроці часу виконуються правила еволюції, і система переходить в новий стан.

Класифікація клітинних автоматів також може залежати від їхніх характеристик, таких як тип сітки, розмірність, тип правил та інші. Клітинні автомати застосовуються в різних областях, включаючи теоретичну біологію, комп'ютерні науки, фізику та інші галузі для моделювання та вивчення складних систем.

Деякі з можливих критеріїв класифікації:

Розмірність:

- 1D (одновимірні): Клітини розташовані на лінії.
- 2D (двовимірні): Клітини розташовані у вигляді площини.
- 3D (тривимірні): Клітини розташовані у просторі.

Тип сітки:

- Регулярна сітка: Клітини розташовані у визначеному порядку.
- Нерегулярна сітка: Клітини розташовані без конкретного порядку.

Стани клітин:

- Дискретні стани: Клітини можуть приймати обмежений набір станів.
- Неперервні стани: Клітини можуть приймати будь-які значення в певному діапазоні.

Правила еволюції:

- Детерміновані правила: Стан кожної клітини визначається чітко визначеними правилами.
- Ймовірнісні правила: Ймовірність переходу в новий стан може варіюватися.

Типи класифікаційних задач:

- Класифікація за призначенням: Моделі, призначені для конкретних завдань, наприклад, моделювання конкретних фізичних систем чи біологічних процесів.
- Універсальні класифікатори: Моделі, які можуть застосовуватися для широкого спектру задач.

Застосування:

- Теоретичні дослідження: Моделі, які використовуються для дослідження властивостей клітинних автоматів та їхніх можливих застосувань.
- Застосування в конкретних галузях: Моделі, розроблені для вирішення конкретних завдань у біології, фізиці, комп'ютерних науках тощо.

Розглянемо детальніше два клітинних автомати, які представлено у даній роботі.

Клітинний автомат Game of Life

Гра "Життя" (Game of Life): Розроблена британським математиком Джоном Конвеєм у 1970 році, гра "Life" стала однією з найвідоміших клітинних автоматів. Вона використовує правила, які визначають, що клітина залишається "живою" або "мертвою" в залежності від кількості її живих сусідів в оточенні. Гра "Життя" служила як приклад для вивчення емерджентних властивостей та хаотичних систем. В грі "Життя" кожна клітина має вісім сусідів. Правила передбачають, що

жива клітина, яка має менше двох або більше трьох живих сусідів, помирає від "самотності". Якщо у живої клітини два або три живих сусіда, вона залишається живою. Нова жива клітина з'являється, якщо у неї є три живих сусіда. Ці прості правила призводять до виникнення складних структур і паттернів.

Фішка може мати максимум вісім сусідів: чотири ортогональні та чотири діагональні. Конвей визначає три правила для утримання, видалення та додавання нових фішок на шаховому полі. Фішка залишається на своєму місці на шаховому полі (виживає), якщо вона має два або три сусіди. Фішка видаляється з шахового поля (помирає), якщо у неї чотири або більше сусідів або один або жодного сусіда. Фішка додається в порожню клітину на шаховому полі (народжується), якщо у клітини саме три сусіди. Всі народження та смерті фішок відбуваються одночасно, що означає, що гравець видаляє та додає фішки після того, як правила Конвея застосовуються до всіх фішок. Коли всі мертві фішки видаляються, а всі нові фішки розміщуються на шаховому полі, створюється нове "покоління" популяції.

Існують різні патерни, які може розвивати популяція, залежно від початкового розташування фішок. У деяких випадках популяція вимирає. Деякі інші початкові розташування призводять до стійких патернів, які більше не змінюються. Ще однією можливістю є те, що популяція нескінченно виявляє фіксовану кількість патернів у фіксованому порядку. Ці повторюючіся патерни називаються "осциляторами". Є багато інших патернів, які можуть еволюціонувати. Веб-сайт Еріка Вейштейна "Ковзаючий скарб" надає пошуковий каталог більше 200 можливих патернів.

Гра життя стала популярною особливо серед ентузіастів комп'ютерів. Різні версії гри, з модифікованими правилами, були введені для моделювання біологічних процесів, таких як формування патернів. Однією з причин, чому гра так захоплює гравця, є те, що з простих правил та початкових позицій можуть еволюціонувати складні патерни. Проте, коли Гарднер публікував свої статті про Гру життя, цифрові персональні комп'ютери ще не були доступні, і гру грали на шаховому полі, на папері або на великих та дорогих комп'ютерах головного

фрейму. Як стверджують Іво та Івона Бялиницькі-Бірули в своїй книзі "Моделювання реальності: Як комп'ютери відображають життя", це коштувало американським компаніям мільйони доларів, оскільки багато ентузіастів Гри життя витрачали багато коштів на відтворення гри.

Нижче, на рисунку 1.1, представлено можливі досліджені варіації початкових умов в клітинному автоматі “Game of life”. Кожна з них з ходом роботи розробленого алгоритму буде видозмінювати себе, шляхом взаємодії клітин-сусідів. Різні категорії початкових умов будуть давати відмінні від інших кінцеві результати, які можуть бути неочікувані досліднику. Інші варіації початкових умов генерації клітинного автомату є не дослідженими.

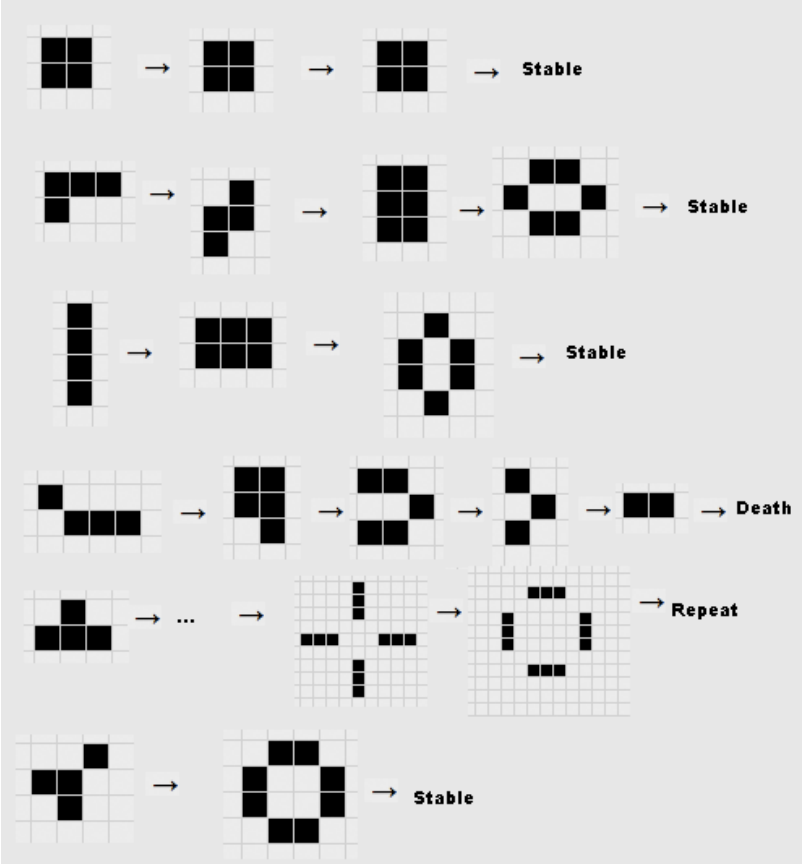

Рисунок 1.1 – Приклади деяких структур в грі “Життя”

Клітинний автомат Langton's Ant

Мураха Ленгтона (Langton's Ant): Створена Крістофером Ленгтоном у 1986 році. "Мураха Ленгтона" - це одноруховий автомат, де мураха рухається по сітці і змінює стан клітини, на якій вона опиняється. Цей автомат став популярним завдяки тому, як мураха поводиться на сітці і створює складні патерни та структури. Якщо мураха опиняється на білій клітині, вона робить поворот на 90 градусів вправо і змінює цю клітину на чорну. Якщо мураха опиняється на чорній клітині, вона робить поворот на 90 градусів вліво і змінює клітину на білу. Мурашка рухається безкінечно, і ця проста система призводить до створення складних геометричних структур.

Мурахи слідкують за псевдовипадковим шляхом до приблизно 10 000 кроків. Зрештою, мурахи починають будувати повторюваний "шосе" з 104 кроків, яке нескінченно повторюється. Усі скінченні початкові конфігурації, які були випробувані, в кінцевому підсумку збігаються до одного і того ж ітераційного патерну. Це свідчить про те, що "шосе" було приманкою для Мурахи Ленгтона, але ніхто не вдалося досягти цього. Відомо тільки, що траєкторія мурахи завжди є нескінченною, незалежно від початкової конфігурації.

Мураха Ленгтона, власне кажучи, є прикладом того, як прості правила можуть призводити до вираженої та прогнозованої системної динаміки. Експерименти з різними початковими конфігураціями підтверджують, що незалежно від вхідних умов, мураха завжди виявляє тенденцію до формування сталих структур.

В цілому, Мураха Ленгтона є цікавим об'єктом вивчення в контексті клітинних автоматів, привертаючи увагу через свою несподівану та властиву складність, яка виникає з простих правил взаємодії.

1.2 Опис математичних методів

У світі клітинних автоматів, таких як Мураха Ленгтона та Гра життя, важливо зазначити, що перед початком реалізації системи слід глибоко опрацювати теоретичний аспект. Однак, подібно до вивчення клітинних автоматів, де правила простих взаємодій можуть породжувати складну динаміку, в аналізі даних також існує потреба у вивченні взаємодій та виборі ключових параметрів.

Коли мова йде про вибір методів збору інформації, ми можемо зробити паралель із вибором правил в клітинному автоматі Мураха Ленгтона. Аналізуючи дані та їх взаємозв'язки, схоже на розгляд динаміки життя в клітинному автоматі Гра життя, де зміни в одній області можуть мати далекосяжні наслідки для системи в цілому.

Основна ідея полягає в тому, щоб вивчати не лише окремі параметри, але й їх взаємодію та вплив на загальний результат. Подібно до того, як клітини в клітинних автоматах реагують одна на одну, так і взаємодія параметрів може суттєво впливати на здатність моделі передбачати результати.

Отже, вибір типу навчання даних та аналіз різних методів передбачення результатів можна порівняти з розвитком правил та стратегій в клітинних автоматах. Аналізуючи різні регресійні моделі, ми можемо визначити, які з них найкраще відповідають вимогам нашої конкретної задачі, подібно до того, як в клітинних автоматах вибирають правила для досягнення певної мети в системі.

Математична модель клітинного автомату Game of life

У клітинному автоматі "Гра життя" функціональні залежності визначають, які стани клітин змінюються наступним чином в залежності від їх поточного стану та оточення.

Залежність стану наступного покоління від поточного стану клітини:

- Якщо клітина жива (знаходиться в стані 1):
  - Якщо має менше двох живих сусідів або більше трьох живих сусідів, вона помирає (становиться мертвою).
  - Якщо має два або три живих сусідів, вона залишається жити.
- Якщо клітина мертва (знаходиться в стані 0):
  - Якщо має точно три живих сусіди, вона оживає (становиться живою).

Залежність від кількості сусідів:

- Кількість живих сусідів для кожної клітини визначає, як буде змінено її стан наступного покоління.
- Це визначає динаміку розвитку структур в грі.

Ці функціональні залежності роблять клітинний автомат "Гра життя" динамічним та цікавим, забезпечуючи можливість еволюції великого спектру структур на основі простих правил взаємодії.

Алгоритм "Гра життя" включає в себе декілька ключових моментів, які визначають еволюцію системи клітин на полі гри:

Початкова конфігурація:

- Початкова розташування живих та мертвих клітин на полі гри визначає початковий стан системи.

Цикл гри:

- Проведення ітераційних циклів, де застосовуються правила взаємодії для кожної клітини на полі.
- Для кожної ітерації визначається новий стан клітин в залежності від їх поточного стану та оточення.

Правила взаємодії:

- Кожна клітина може мати два стани: живий (1) або мертвий (0).
- Залежно від кількості живих сусідів, застосовуються правила для визначення, чи клітина залишиться жити, помре чи воскресне.

Визначення сусідів:

- Розрахунок кількості живих сусідів для кожної клітини.

- Сусіди можуть бути по горизонталі, вертикалі або діагоналі.

Актуалізація стану клітин:

- Зміна стану кожної клітини відповідно до визначених правил взаємодії.
- Це може бути реалізовано у вигляді нового поля гри, яке стає актуальним на наступній ітерації.

Циклічність:

- Процес ітерацій повторюється знову та знову, визначаючи еволюцію системи в часі.
- Система може досягти стабільних структур, еміграції, або взагалі стати хаотичною.

Ці ключові моменти визначають логіку і рушійні сили алгоритму "Гра життя", який дозволяє моделювати цікаві та динамічні патерни на клітинному полі.

Аналіз "Гри життя" може бути здійснений за допомогою різноманітних методів для вивчення динаміки системи, виявлення структур та встановлення властивостей. Ось кілька методів аналізу "Гри життя":

Візуалізація:

- Спостереження за еволюцією структур шляхом візуалізації на полі гри.
- Використання графіків або анімацій для відстеження змін в системі.

Стабільні структури:

- Визначення і вивчення стабільних структур, таких як "блоки", "лічильники", "літаки" та інші.
- Аналіз можливості виникнення та довжини життя таких структур.

Осцилятори:

- Виявлення та вивчення періодичних осциляторів, які повертаються до свого початкового стану через певну кількість ітерацій.

Космічні кораблі:

- Дослідження рухомих структур, таких як "космічні кораблі", які можуть пересуватися на полі гри.

Хаотичні структури:

- Вивчення хаотичних або псевдохаотичних структур, які не стабільні та не періодичні, але також не руйнуються.

Комплексні конструкції:

- Аналіз складних конструкцій, які можуть виникати за участю великої кількості клітин.

Глайдери:

- Вивчення "глайдерів" — конструкцій, які переміщаються на полі гри.

Пошук стартових конфігурацій:

- Знаходження стартових конфігурацій, які можуть призводити до цікавих ефектів або структур.

Структурні зміни:

- Вивчення та аналіз змін у структурах при взаємодії різних елементів.

Приклади складних структур представлено на рисунках 1.2 та 1.3. Вони можуть породжувати нові фігури, які будуть функціонувати самостійно, пересуватися по полю, взаємодіючи та впливаючи на інші функціонуючі паралельно структури. Це дає нескінченний потенціал для досліджень цієї теми, аналізуючи взаємодію двох ізольованих структур, в роботу яких втручається третя та непередбачено впливає на кінцевий результат.

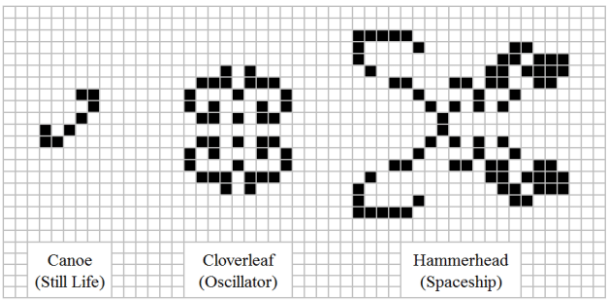

Рисунок 1.2 – Приклади структур

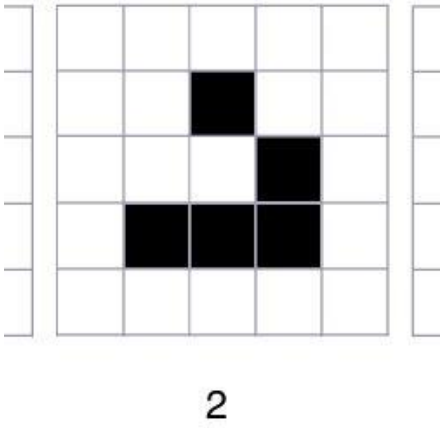

Рисунок 1.3 – Приклад простого глайдера

Ці методи аналізу допомагають розкрити багато аспектів "Гри життя" та досліджувати різноманітні поведінки та структури, що можуть виникати в цьому клітинному автоматі.

Математична модель клітинного автомату Langton ant

Створення клітинного автомату Ленгтона (Langton's Ant) включає в себе використання математичних методів та алгоритмів для визначення правил еволюції системи. Ось підґрунтя математичних методів та алгоритмів, які використовуються у створенні клітинного автомату Ленгтона:

Представлення поля:

- Використання матриці або двовимірного масиву для представлення клітинного поля.
- Кожній клітині призначається значення (біла або чорна).

Алгоритм руху мурахи:

- Опис алгоритму руху мурахи відповідно до правил автомата.
- Визначення кроків мурахи при зміні стану клітини.

Взаємодія з полем:

- Реалізація алгоритму, який визначає взаємодію мурахи з клітинами на полі.
- Визначення змін у стані клітини та напрямку руху мурахи в залежності від кольору клітини.

Візуалізація:

- Використання алгоритмів для візуалізації еволюції системи на полі.
- Відображення кожного кроку руху мурахи та змін в стані клітин.

Управління ітераціями:

- Реалізація алгоритму для керування ітераціями гри.
- Прийняття рішення про кількість ітерацій або умови завершення гри.

Аналіз та вивід результатів:

- Реалізація алгоритмів для аналізу та виведення результатів гри.
- Вивід статистики, виявлення закономірностей чи особливостей руху мурахи.

У клітинному автоматі Ленгтона (Langton's Ant) функціональні залежності визначають, як кожна клітина поводиться на кожному кроці, взаємодіючи з оточенням. В цьому автоматі залежність визначається простими правилами, що визначають, як змінюється стан клітини і як рухається "мураха" на полі. Основні функціональні залежності в автоматі Ленгтона:

Рух мурахи:

- Якщо клітина, на якій знаходиться мураха, біла, то вона перефарбовується на чорний колір, мураха повертається наліво (проти годинникової стрілки) і рухається на одну клітину вперед.
- Якщо клітина чорна, то вона перефарбовується на білий колір, мураха повертається направо (за годинниковою стрілкою) і рухається на одну клітину вперед.

Стан клітини:

- Кожна клітина може бути у двох станах: біла або чорна.
- Зміна стану клітини визначається діями мурахи.

Детермінізм:

- Поведінка автомата є детермінованою, тобто для однакової початкової конфігурації і однакових правил гра завжди розвиватиметься однаково.

Безкінечність поля:

- Поле гри автомата Ленгтона є безкінечним, тобто мураха може рухатися в будь-якому напрямку без обмежень.

Ці прості функціональні залежності визначають основні правила руху і зміни стану клітин у клітинному автоматі Ленгтона.

Аналіз клітинного автомата Ленгтона (Langton's Ant) може включати в себе різноманітні методи для вивчення та розуміння його динаміки та властивостей. Ось деякі методи аналізу автомата Ленгтона:

Візуалізація:

- Спостереження за рухом мурахи та змінами на полі гри через візуальне представлення.
- Створення анімацій або графіків для відстеження еволюції системи.

Аналіз структур:

- Вивчення формування та властивостей структур, які можуть виникати в ході гри.

Дослідження поведінки мурахи:

- Аналіз траєкторії та міркувань мурахи в різних початкових умовах.
- Виявлення закономірностей та регулярностей у русі мурахи.

Вивчення періодичних структур:

- Виявлення періодичних властивостей гри, таких як циклічні або повторювані патерни.

Розгляд модифікацій:

- Дослідження властивостей модифікованих версій автомата Ленгтона зі зміненими правилами чи параметрами.

Пошук стартових конфігурацій:

- Аналіз ефектів та результатів при певних початкових конфігураціях.
- Пошук початкових умов, які призводять до цікавих динамічних станів.

Статистичний аналіз:

- Збір та аналіз статистичних даних щодо руху мурахи та змін в системі протягом ітерацій.

Ці методи можуть допомогти розкрити різноманіття характеристик та властивостей клітинного автомата Ленгтона, роблячи аналіз системи більш повним та зрозумілим.

Приклад роботи п’яти мурах представлено на рисунку 1.4.

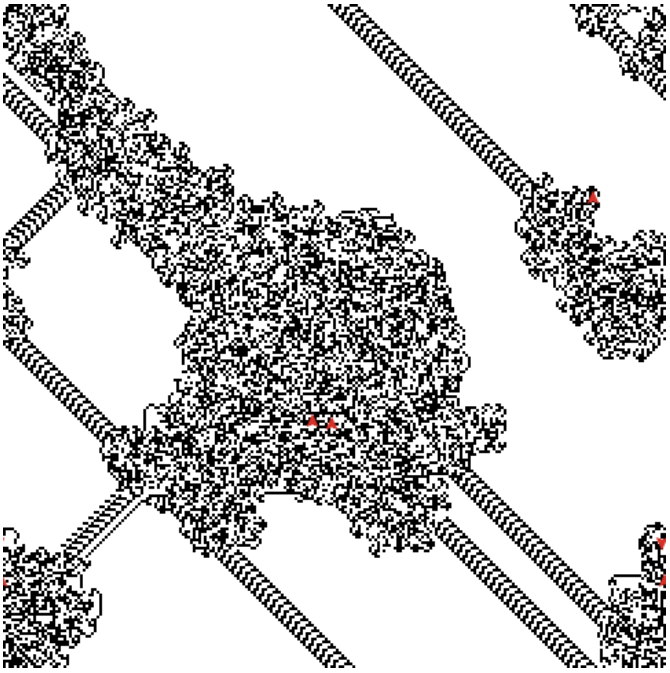

Рисунок 1.4 – Приклад роботи п’ятьох мурах

1.3 Висновки до розділу

У результаті аналізу було виявлено, що моделі клітинних автоматів володіють значним потенціалом для моделювання складних систем, відображення динаміки процесів та розв'язання великої кількості завдань, пов'язаних з обробкою інформації та аналізом даних.

Встановлено переваги та обмеження кожної конкретної моделі клітинного автомата, що допомагає визначити їхню ефективність в різних областях застосування. Результати можуть служити основою для подальших наукових досліджень динамічних систем з метою розширення їхнього застосування.

Було обрано дві моделі клітинних автоматів для подальшої візуалізації їх роботи. Цей вибір спричинений розповсюдженістю та широким застосуванням даних алгоритмів для опису динамічних систем та існуванням такого початкового положення клітин автоматів, розвиток яких потребує подальшого дослідження.

2 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВІЗУАЛІЗАЦІЇ КЛІТИННИХ АВТОМАТІВ

Клітинні автомати - це модель обчислень, яка базується на простому наборі правил для еволюції станів клітин на сітці. Ця концепція зазвичай використовується в теорії обчислень та математичних дослідженнях. Однак, в певних випадках, клітинні автомати можуть бути використані для комерційних цілей.

Ось кілька можливих застосувань:

- **Моделювання процесів:** Клітинні автомати можна використовувати для моделювання різних фізичних, хімічних або біологічних процесів. Наприклад, вони можуть бути використані для моделювання росту популяції, розповсюдження захворювань чи взаємодії між об'єктами.
- **Криптографія:** В деяких випадках клітинні автомати використовуються для створення криптографічних алгоритмів або генерації ключів. Властивості хаосу та випадковості у клітинних автоматах можуть бути використані для криптографічних цілей.
- **Штучний інтелект:** Деякі клітинні автомати можуть використовуватися для моделювання штучного інтелекту та еволюції поведінки систем. Вони можуть застосовуватися в галузі робототехніки або автоматичного управління.
- **Ігрова індустрія:** У галузі розробки відеоігор можливі застосування клітинних автоматів для створення складних та динамічних ігрових світів. Вони можуть контролювати різні аспекти гри, такі як поведінка противників, генерація ландшафту чи динамічна зміна умов гри.
- **Графічний дизайн:** Клітинні автомати можна використовувати для створення цікавих та абстрактних графічних патернів чи дизайнів.

- **Симуляції виробництва та логістики:** В деяких випадках клітинні автомати можуть бути використані для симуляції процесів виробництва або логістики, де можна моделювати рух об'єктів, взаємодію між різними частинами системи та інші параметри.

Перед використанням клітинних автоматів в комерційних цілях важливо враховувати їх обмеження та визначити, чи цей метод є найефективнішим для конкретної задачі. Також, зазвичай необхідно враховувати можливі проблеми з продуктивністю та ресурсами, особливо при великому масштабі.

2.1 Постановка задачі

В зв’язку з недостатнім дослідженням існуючих алгоритмів клітинних автоматів та відсутності можливого для цього застосунку, постає задача розробки візуальної моделі клітинного автомата з врахуванням початкових умов заданих користувачем. Також необхідно забезпечити ймовірне випадкове втручання в процес побудови.

Ключовою метою є візуалізація генерації та автоматизація побудови клітинних автоматів для подальшого дослідження їх розвитку, вивчення їх властивостей. Розробка програмного забезпечення для існуючих моделей та алгоритмів дозволяє дослідити відомі процеси при різних початкових умовах.

Основні завдання включають:

- **Аналіз існуючих клітинних автоматів:** Дослідження наявних моделей клітинних автоматів для розуміння їх основних принципів та можливостей.
- **Розробка існуючих клітинних автоматів:** Розробка існуючих моделей та правил взаємодії клітинок з метою досягнення конкретних цілей в дослідженні з врахуванням завдання користувачем початкових умов.

- **Аналіз результатів:** Оцінка отриманих результатів, їх інтерпретація та виведення висновків щодо досліджуваної проблеми.
- **Застосування в практиці:** Розгляд можливостей використання клітинних автоматів у реальних задачах, таких як оптимізація процесів, передбачення подій, або розробка нових технологій.

Для вирішення поставленої задачі необхідно виконати наступні завдання: проаналізувати існуючі клітинні автомати та їх алгоритми, розробити функціонал для побудови та візуалізації клітинних автоматів, розробити програмне забезпечення для існуючих моделей з можливість їх аналізу.

Аналіз відомих програмних продуктів

Використання клітинних автоматів у різних галузях науки та технологій стає все більш актуальним та розповсюдженим завдяки розвитку програмних продуктів, спрямованих на моделювання та аналіз таких систем. Ці програмні засоби надають дослідникам, інженерам, інформатикам та багатьом іншим можливість ефективно вивчати та використовувати клітинні автомати для вирішення різноманітних завдань.

Найпоширеніші програми, що працюють з клітинними автоматами, включають такі:

- **Game of Life Simulator (Golly):** Golly - це одна з найвідоміших та потужних програм для моделювання клітинних автоматів, заснована на відомій "Грі життя" Конвея. Вона дозволяє вам створювати, відтворювати та аналізувати клітинні автомати різної складності та застосування.
- **Cellular Automata Studio:** Ця програма спрямована на створення та аналіз клітинних автоматів, дозволяючи користувачам створювати власні правила та відстежувати еволюцію систем.

- **MATLAB:** MATLAB - популярний інструмент у сфері наукових досліджень, який надає можливість роботи з клітинними автоматами завдяки великій кількості доступних пакетів і функцій.
- **Wolfram Mathematica:** Програма Mathematica включає в себе багато функцій для роботи з клітинними автоматами, включаючи властивий ним мову програмування Wolfram Language.
- **NetLogo:** NetLogo - це спеціалізоване середовище для агентно-орієнтованого моделювання, яке включає підтримку клітинних автоматів та інших важливих концепцій.

Ці програмні продукти надають можливість аналізувати імітації різних систем, від відтворення біологічних процесів до моделювання соціальних явищ та процесів в інших галузях науки. Вони допомагають дослідникам та фахівцям вирішувати складні завдання та аналізувати системи, які змінюються з часом, що робить їх незамінними інструментами для дослідження та моделювання.

Нижче наведена порівняльна таблиця програм для роботи з клітинними автоматами. Кожен з цих додатків має свої недоліки та переваги, тому користувач може обрати найзручніше середовище для дослідження, опираючись на свої вимоги.

Переваги та недоліки занесемо до таблиці 2.1.

Таблиця 2.1 - Переваги та недоліки програм

Програма	Переваги	Недоліки
Golly	• Велика спільнота користувачів та активна підтримка. • Широкі можливості для моделювання клітинних автоматів. • Відкритий код.	• Інтерфейс може виглядати складним для новачків.
Cellular Automata Studio	• Простий інтерфейс. • Зручність для створення та аналізу клітинних автоматів. • Підтримка різних правил та параметрів.	• Може виявитися обмеженим для складних систем.
MATLAB	• Велика функціональність для наукових досліджень. • Широкі можливості моделювання клітинних автоматів.	• Вимагає знань MATLAB для використання
Wolfram Mathematica	• Інтегрована середовище розробки та обчислення. • Вбудовані функції для роботи з клітинними автоматами.	• Висока вартість ліцензії. • Не відкритий код.

2.2 Підсистеми програмного продукту

У цьому розділі ми детально розглянемо ключові підсистеми нашого програмного продукту, вивчаючи їхню функціональність, взаємодію та внутрішню архітектуру. Аналіз підсистем дозволить нам краще зрозуміти, як кожна частина вносить вклад у загальний успіх продукту та як їхні взаємодії сприяють досягненню поставлених цілей. Звертаючись до кожної підсистеми окремо, ми зможемо визначити їхні сильні та слабкі сторони, а також виявити можливості для подальшого вдосконалення та оптимізації.

Сервіс моделі генерації клітинних автоматів

Сервіс має надати користувачу можливість вільно працювати та вести дослідження на основі візуалізованих даних клітинних автоматів. Надати можливість вільно змінювати початкові умови та характеристики поля. Забезпечити стабільну роботу та зрозумілий інтерфейс.

Веб-додаток

Метою створення веб-додатку є створення зручного інтерфейсу для користування моделлю клітинних автоматів. Основною задачею веб-додатку є створення, відображення та аналіз клітинних автоматів для різноманітних досліджень та застосувань.

Для досягнення цієї мети були визначені такі завдання при розробці веб-додатку:

- **Реєстрація та авторизація користувачів:** Додаток повинен надати можливість користувачам реєструватися та авторизуватися для збереження їхніх історичних даних та налаштувань.
- **Робота з клітинними автоматами:** Користувачам слід надати можливість вільно проводити дослідження з даними клітинними автоматами, власноруч обираючи налаштування для них.
- **Аналіз та візуалізація:** Веб-додаток повинен дозволяти користувачам аналізувати та візуалізувати еволюцію клітинних автоматів, включаючи відстеження змін в часі.
- **Збереження результатів та історії:** Користувачі повинні мати можливість зберігати свої спостереження, а також переглядати історію своїх досліджень та аналізувати розвиток клітинних автоматів з часом.
- **Інтерфейс для роботи з клітинними автоматами:** Веб-додаток має надати зручний інтерфейс налаштування та аналізу клітинних автоматів, що включає інструменти для редагування правил та початкових умов.
- **Безпека та конфіденційність:** Важливим завданням є забезпечення безпеки даних користувачів та конфіденційності їхньої інформації.

Ці завдання допоможуть створити веб-додаток, який надасть можливість користувачам працювати з клітинними автоматами в онлайн-режимі та використовувати їх для вивчення та дослідження різних процесів.

Отже, метою створення веб-додатку є надання користувачам зручного інтерфейсу для вивчення та аналізу клітинних автоматів. Для досягнення цієї мети визначено ряд завдань, включаючи реєстрацію та авторизацію користувачів, роботу з клітинними автоматами, аналіз та візуалізацію їх еволюції, збереження результатів та історії, створення зручного інтерфейсу для користувачів та забезпечення безпеки даних.

Цей веб-додаток створює можливість для користувачів вивчати та досліджувати клітинні автомати, використовуючи їх для аналізу різних процесів, і

забезпечує їх зручний та безпечний дослідницький інструмент в онлайн-середовищі.

Бек-енд сервіс

Бек-енд сервіс в даній програмі є важливою складовою, яка відповідає за опрацювання бізнес-логіки та забезпечення взаємодії між користувачем та базою даних. В контексті нашого програмного продукту, який включає функціонал авторизації та реєстрації користувачів, а також роботу з файлами-фотографіями та використання PostgreSQL для зберігання даних, бек-енд сервіс грає важливу роль у забезпеченні його функціональності та надійності.

- **Авторизація та реєстрація користувачів:** Бек-енд сервіс відповідає за обробку запитів користувачів щодо авторизації та реєстрації. Він перевіряє введені дані, взаємодіє з базою даних для перевірки існуючих облікових записів і створення нових користувачів.
- **Робота з файлами:** Бек-енд сервіс надає можливість користувачам завантажувати фотографії та зберігає їх у системі. Він також відповідає за збереження шляхів до фотографій в базі даних, щоб забезпечити доступ до них у подальшому.
- **Взаємодія з PostgreSQL:** Оскільки для зберігання даних використовується PostgreSQL, бек-енд сервіс взаємодіє з цією системою керування базами даних. Він виконує запити для збереження, витягування та оновлення інформації, що стосується користувачів, фотографій та інших даних.
- **Обробка помилок:** Бек-енд сервіс також відповідає за обробку помилок та відправку відповідних повідомлень користувачам. Він забезпечує надійну роботу програми, запобігаючи критичним

помилкам та забезпечуючи зручний інтерфейс для користувачів у випадку непередбачених ситуацій.

Бек-енд сервіс взаємодіє з фронт-енд частиною програми, забезпечуючи її функціональність та надійність, і є важливим компонентом для створення успішного веб-додатку, який відповідає потребам користувачів.

База даних

У дослідженні використовується система управління базами даних PostgreSQL для забезпечення ефективного зберігання та обробки інформації, пов'язаної з моделлю генерації клітинних автоматів. PostgreSQL використовується для виконання наступних задач у програмі дипломної роботи:

- **Зберігання даних користувачів:** PostgreSQL використовується для створення бази даних, яка містить інформацію про користувачів системи. Це включає основні особисті дані, ідентифікатори та параметри, необхідні для ідентифікації та індивідуалізації користувачів.
- **Зберігання записів користувачів про клітинні автомати:** PostgreSQL служить для зберігання історії та динаміки еволюції клітинних автоматів, що виникають в результаті взаємодії користувачів з моделлю. Це включає в себе дані про початкові конфігурації, внесені зміни, та інші параметри, які дозволяють відстежувати розвиток автоматів у часі.
- **Забезпечення надійності та цілісності даних:** PostgreSQL забезпечує надійність зберігання даних та цілісність інформації, завдяки використанню транзакцій та механізмів контролю доступу. Це дозволяє уникнути втрати даних та забезпечити консистентність записів.

- **Оптимізація запитів для аналізу результатів:** PostgreSQL використовується для оптимізації та виконання запитів, пов'язаних з аналізом та візуалізацією результатів моделі генерації клітинних автоматів. Це дозволяє швидко та ефективно отримувати дані для подальшого наукового аналізу.

Сервіс JWT-автентифікації

JWT (JSON Web Token) - це відкритий стандарт (RFC 7519) для створення токенів доступу, які можуть містити інформацію та підпис цієї інформації. JWT став популярним механізмом аутентифікації та авторизації в розподілених системах, особливо в архітектурі клієнт-сервер та веб-застосунках. Використання JWT спрощує автентифікацію та передачу даних про авторизацію між сторонами.

JSON Web Token складається з трьох компонентів:

1. **Header (заголовок):** Містить два поля: тип токена (typ) і алгоритм підпису (alg). Заголовок базується на форматі JSON та Base64Url.
   Приклад:
   ```
   {
    "typ": "JWT",
    "alg": "HS256"
   }
   ```
2. **Payload (навантаження):** Містить корисну інформацію (корисні дані) та також представлене у форматі JSON та Base64Url. Ця частина містить стандартні заяви (claims) та власні заяви (custom claims), які можуть містити будь-яку корисну інформацію.
   Приклад:
   ```
   {
    "sub": "1234567890",
   ```

```
  "name": "John Doe",
  "iat": 1516239022
}
```

3. **Signature (підпис):** Створюється за допомогою алгоритму, вказаного у заголовку, та секретного ключа. Він гарантує, що токен не був змінений під час передачі.

JWT використовується для:

- **Аутентифікації:** Після успішної аутентифікації користувача сервер може створити JWT, який містить інформацію про користувача та його права доступу. Цей токен може бути включений до кожного запиту для додаткової автентифікації та авторизації.
- **Авторизації:** JWT може містити інформацію про те, які ресурси чи послуги користувач має право використовувати. Це дозволяє контролювати доступ користувача до конкретних ресурсів.
- **Інформаційного обміну:** JWT може використовуватися для безпечного обміну інформацією між сторонами. Інформацію може включати певні деталі про користувача чи інші дані.

JWT використовується для забезпечення безпеки та цілісності даних між двома сторонами та може бути використаний для реалізації системи одноразового входу, аутентифікації користувача та передачі даних авторизації. Однією з основних переваг JWT є те, що він може бути легко використаний у веб-застосунках, де він може бути збережений у куках або локальному сховищі браузера.

Стейт-менеджер

Стейт-менеджер (State Manager) - це поняття, яке широко використовується в розробці програмного забезпечення, зокрема в інтерфейсах користувача. Він

відноситься до засобу управління станом додатку, тобто інформацією, яка визначає його поточний стан та поведінку.

Основні аспекти стейт-менеджера включають:

- **Зберігання стану:** Стейт-менеджер відповідає за ефективне зберігання та оновлення стану додатку. Це може включати в себе дані, такі як стан користувача, параметри введення та інші важливі інформаційні частини.
- **Моделювання взаємодії:** Він визначає спосіб, як додаток реагує на події, змінює свій стан та сповіщає компоненти чи модулі про ці зміни. Це робить можливим визначення логіки та правил, які контролюють потік даних та взаємодію елементів інтерфейсу.
- **Постачання даних компонентам:** Стейт-менеджер може постачати поточний стан додатку компонентам та модулям, що дозволяє їм відображати та взаємодіяти з цим станом.
- **Обробка асинхронних операцій:** Деякі стейт-менеджери дозволяють ефективно обробляти асинхронні операції та управляти їхнім впливом на стан додатку.

Стейт-менеджери часто використовуються у веб-розробці з фреймворками, такими як React, Angular або Vue.js. У контексті дипломної роботи, аналіз та вибір стейт-менеджера може виглядати як важливий аспект проекту, оскільки він впливає на ефективність та організацію логіки додатку.

У ReactJS немає дефолтного стейт-менеджера у сенсі окремої системи, яка автоматично використовується для управління станом. Замість цього, React надає вбудовану систему стану, яка базується на використанні локального стану компонентів та контексту.

Кожен компонент у React може мати свій власний локальний стан, який визначається та оновлюється за допомогою методу setState. Це дозволяє кожному компоненту управляти своїм внутрішнім станом.

Контекст у React дозволяє передавати дані через компоненти безпосередньо між дочірніми та батьківськими компонентами, що робить його корисним для управління глобальним станом додатку.

Механізм роботи CORS

CORS (Cross-Origin Resource Sharing) - це механізм, який дозволяє веб-сторінці запитувати ресурси з іншого домену, ніж той, з якого була завантажена сама сторінка. Безпека браузера забороняє такі запити через політику одного джерела (Same-Origin Policy). CORS впроваджується для зручності взаємодії між різними доменами веб-додатків та сервісів.

Основні концепції та принципи CORS:

- **Same-Origin Policy (SOP):** Це безпека браузера, яка обмежує веб-сторінки у взаємодії з ресурсами (наприклад, запитами AJAX) на інших доменах. Без CORS, браузер блокує такі запити.
- **Cross-Origin Requests:** Коли веб-сторінка відправляє AJAX-запит на інший домен, браузер спочатку виконує "префлайт" запит (OPTIONS-запит) для визначення, чи дозволяє сервер отримувати крос-доменні запити. Після цього він виконує основний запит (GET, POST, тощо).
- **CORS Headers:** Сервер повинен повертати спеціальні HTTP-заголовки, які дозволяють чи блокують крос-доменні запити. Найбільш важливі заголовки CORS:
- **Access-Control-Allow-Origin:** Вказує, які домени мають доступ до ресурсу. Значення може бути конкретним доменом, "*" (для дозволу доступу всім) чи списком доменів.
- **Access-Control-Allow-Methods:** Вказує, які HTTP-методи дозволені при взаємодії з ресурсом (GET, POST, PUT, DELETE, тощо).

- **Access-Control-Allow-Headers:** Вказує, які HTTP-заголовки дозволені при взаємодії з ресурсом.
- **Access-Control-Allow-Credentials:** Вказує, чи можна передавати кредити (наприклад, куки чи HTTP-автентифікацію) разом із запитами.
- **Simple Requests та Preflight Requests:** Прості запити (Simple Requests) можуть бути виконані без префлайт-запитів. Вони задовольняють певні умови, такі як використання певних методів та заголовків. У інших випадках виконується префлайт-запит для визначення дозволу сервера.

CORS використовується для забезпечення безпеки браузера та дозволу взаємодії між різними доменами веб-додатків. Його важливість полягає в забезпеченні гнучкості та можливості обміну ресурсами між клієнтською та серверною стороною, що розташовані на різних доменах.

Взаємодія підсистем програмного продукта

Фронт-енд та Бек-енд частини додатку взаємодіють між собою через HTTP запити та відповіді. Користувач, взаємодіючи з веб-додатком через свій браузер, створює запити, які надсилаються на Бек-енд. Бек-енд обробляє ці запити, виконує бізнес-логіку, доступ до бази даних та інші операції, після чого надсилає відповіді назад на Фронт-енд. Фронт-енд, у свою чергу, обробляє ці відповіді та відображає користувачу результати на веб-сторінці.

Цей обмін даними дозволяє створити зручний та функціональний інтерфейс для користувача та забезпечити виконання всіх необхідних операцій на стороні сервера. Взаємодія між Фронт-енд та Бек-енд дозволяє створити повноцінний веб-додаток, який відповідає потребам користувачів і надійно функціонує.

Розглянемо схему взаємодії клієнтської та серверної частини через HTTP на рисунку 2.1.

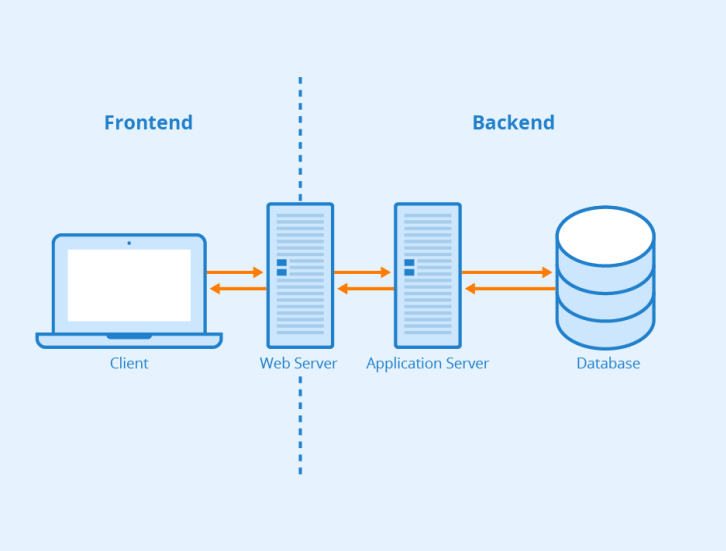

Рисунок 2.1 – Графічне представлення взаємодії через HTTP

2.3 Засоби розробки

В якості мови програмування для розробки веб-додатків і створення інтерактивних веб-сторінок було обрано JavaScript. JavaScript є однією з найважливіших мов програмування в області веб-розробки і відіграє ключову роль у створенні сучасних інтернет-ресурсів.

Засоби розробки програмного продукту є важливими компонентами процесу створення веб-додатку та визначають технологічний стек, який програмісти використовують для розробки. В нашому випадку ми використовували наступні засоби:

- **Vite:** Vite - це сучасний і швидкий інструмент для розробки фронт-енду, який дозволяє легко налаштувати середовище розробки та автоматично оптимізувати ресурси для швидкого завантаження сторінок в браузері. Він є популярним вибором для розробки веб-додатків з використанням фреймворків, таких як React.

- **ReactJS:** React - це бібліотека для створення інтерфейсів користувача (UI), яка дозволяє легко створювати компоненти та ефективно оновлювати відображення відповідно до змін даних. Вона є популярним інструментом для розробки фронт-енду великої кількості веб-додатків і сайтів.
- **Node.js:** Node.js - це середовище виконання JavaScript, яке дозволяє розробникам створювати серверну частину програмного продукту з використанням JavaScript. Ви можете використовувати Node.js для створення бек-енду вашого веб-додатку та обробки запитів від клієнтів.
- **Express.js:** Express.js - це популярний фреймворк для створення веб-серверів та розробки API на базі Node.js. Він надає широкий спектр інструментів для реалізації серверної логіки, маршрутизації та обробки запитів, що дозволяє розробникам швидко створювати потужні бек-енд додатки.

Засоби розробки, які ми використовуємо, забезпечують зручність і ефективність процесу розробки та допомагають створити функціональний та швидкий веб-додаток з фронт-ендом на React та бек-ендом на Node.js та Express.js. Тому розглянемо їх детальніше.

2.4 Мова програмування JavaScript

JavaScript був створений з метою надання можливості веб-розробникам створювати динамічні та інтерактивні сторінки, які взаємодіють з користувачами. Він працює безпосередньо в браузері, що дозволяє виконувати код на стороні клієнта. Ця мова відома своєю легкістю використання та широким спектром можливостей, які вона пропонує для розробки.

JavaScript використовується для розробки фронт-енду веб-додатків, включаючи створення інтерфейсів користувача, обробку подій, валідацію даних та взаємодію з веб-серверами для отримання та відправлення даних. Він також знаходить застосування в розробці серверної частини веб-додатків за допомогою платформи Node.js.

У наш час JavaScript став важливою складовою розробки веб-додатків і використовується для створення різних типів додатків, від соціальних мереж і електронної комерції до медіа-ресурсів та ігор. Він продовжує розвиватися і набувати нових функцій, що робить його надзвичайно важливим інструментом для веб-розробників у всьому світі.

JavaScript - це високорівнева, інтерпретована мова програмування, яка використовується для створення інтерактивних веб-сторінок та веб-додатків. Він є однією з найпопулярніших мов програмування на сьогоднішній день і грає важливу роль в розвитку веб-технологій. JavaScript був розроблений Бренданом Айком в 1995 році для браузера Netscape Navigator. Спочатку мова мала назву LiveScript, але пізніше була перейменована в JavaScript, щоб використовувати популярність мови Java. У 1997 році JavaScript був стандартизований Ecma International, і стандарт назвали ECMAScript. Ця стандартизація допомогла виробникам браузерів створити єдиний стандарт для виконання JavaScript.

Сфера використання:

- JavaScript використовується в основному для створення інтерактивних веб-сторінок та веб-додатків. Він дозволяє додавати функціональність до сторінок, взаємодіяти з користувачами та взаємодіяти з веб-серверами для отримання та надсилання даних.
- JavaScript також використовується для розробки серверної частини веб-додатків з використанням платформи Node.js. Це дозволяє розробникам створювати повноцінні бек-енд додатки на базі JavaScript.

- Мова також використовується в розробці мобільних додатків з використанням фреймворків, таких як React Native та Ionic.

Переваги JavaScript:

- **Широке використання:** JavaScript практично необхідний для розробки веб-додатків та веб-сайтів, що робить його дуже популярною мовою.
- **Інтерактивність:** JavaScript дозволяє створювати інтерактивні та динамічні веб-сторінки, що покращує користувацький досвід.
- **Швидкість розробки:** Можливість швидко реагувати на зміни та перевіряти їх в браузері робить розробку більш ефективною.

Недоліки JavaScript:

- **Споживання ресурсів:** JavaScript може бути вимогливим до ресурсів, що може призвести до повільної роботи додатку на старих апаратних пристроях.
- **Безпека:** через можливість виконання зловмисним кодом на стороні клієнта, JavaScript потребує додаткових заходів для забезпечення безпеки.

JavaScript є незамінним інструментом для розробників веб-додатків та веб-сайтів, і він продовжує розвиватися та розширювати свою сферу використання в інших областях розробки.

2.5 **Локальный сервер разработки Vite**

Vite - це інноваційний інструмент для розробки фронт-енду, який спрощує процес створення веб-додатків і покращує продуктивність розробника. Він відомий своєю швидкістю та зручністю використання. Давайте розглянемо переваги та недоліки Vite, а також порівняємо його з аналогами.

Переваги Vite:

- **Швидкість розробки:** Основна перевага Vite - це швидкість. Він використовує сучасний підхід до розробки, який дозволяє без затримок оновлювати зміни у браузері. Ви більше не чекаєте на перекомпіляцію коду під час розробки.
- **Підтримка ES модулів:** Vite підтримує ES модулі, що дозволяє використовувати сучасні стандарти JavaScript та зменшити завантаження браузера.
- **Легкість конфігурації:** Vite постачається з мінімальною конфігурацією за замовчуванням, що дозволяє швидко розпочати роботу. Проте, ви можете налаштувати його за вашими потребами.
- **Підтримка разом з іншими фреймворками:** Vite ідеально підходить для розробки в з'єднанні з популярними фреймворками, такими як React, Vue.js, та інші.
- **Розділена структура проекту:** Vite сприяє розділенню коду на маленькі, незалежні модулі, що покращує підтримку та розширюваність проекту.

Недоліки Vite:

- **Потребує Node.js:** Vite вимагає встановлення Node.js для роботи.
- **Відсутність підтримки старих браузерів:** Однією з недоліків Vite є те, що він орієнтований на сучасні браузери і не надає засобів для автоматичного поліпшення сумісності зі старими версіями браузерів.

Порівняння Vite з аналогами представлено на таблиці 2.2.

Таблиця 2.2 - Порівняльна таблиця Vite з аналогами:

Категорія	Vite	Webpack	Parcel
Швидкість	Висока	Помірна	Середня
Підтримка ESмодулів	Так	Так	Так
Конфігурація	Проста	Складна	Мінімальна

Таблиця 2.2 (продовження)

Підтримка фреймворків	Так	Так	Обмежена
Сумісність зі старими браузерами	Ні	Так	Так

2.6 Бібліотека ReactJS

ReactJS - це популярна бібліотека JavaScript для розробки інтерфейсів користувача. Вона відома своєю декларативністю та компонентною структурою, що дозволяє розробникам легко будувати і підтримувати складні інтерфейси. Ось докладніше про ReactJS, його переваги, недоліки та порівняльну таблицю з аналогами.

Переваги ReactJS:

- **Декларативність:** React використовує декларативний підхід до створення інтерфейсів, що дозволяє описувати, як повинен виглядати інтерфейс у певний момент часу, а не як досягти цього стану.
- **Компонентна структура:** React розбиває інтерфейс на невеликі компоненти, які можна повторно використовувати. Це полегшує розробку і підтримку коду.
- **Віртуальний DOM:** React використовує віртуальний DOM для ефективного оновлення інтерфейсу. Він порівнює попередній стан DOM з новим та оновлює тільки зміни, що покращує продуктивність.
- **Активна спільнота та екосистема:** React має велику та активну спільноту розробників, яка розробляє багато корисних доповнень та бібліотек для розширення можливостей React.

Недоліки ReactJS:

- **Велика кількість понять:** Для новачків React може здатися складним через багато понять, таких як стейт, пропс, контекст та інші.
- **Крива навчання:** Хоча React є потужним інструментом, він може вимагати часу для вивчення та зрозуміння.

Порівняння ReactJS з аналогами представлено на таблиці 3.2.

Таблиця 2.3 – Порівняльна таблиця ReactJS з аналогами:

Критерій	ReactJS	Angular	Vue.js
Декларативність	Так	Так	Так
Компонентра структура	Так	Так	Так
Віртуальний DOM	Так	Ні	Так
Величина спільноти	Велика	Велика	Велика
Швидкість розробки	Висока	Середня	Висока
Вивчення	Середнє	Складне	Просте

Загалом, ReactJS є потужним інструментом для створення інтерфейсів користувача, зокрема для розробки веб-додатків. Вибір між React та іншими фреймворками залежить від конкретних вимог та особистих вподобань розробника.

2.7 Фреймворк Node.js

Node.js – це середовище виконання JavaScript, яке дозволяє розробникам створювати серверні додатки на мові JavaScript. Ось докладніше про Node.js, його переваги, недоліки та порівняльну таблицю з аналогами.

Переваги Node.js:

- Швидкість виконання: Node.js використовує двигун V8 від Google, що робить його дуже швидким у виконанні JavaScript-коду.
- Потокова обробка: Node.js підтримує асинхронну, подійно-орієнтовану обробку подій, що дозволяє ефективно взаємодіяти з багатьма запитами одночасно.
- Велика бібліотека модулів: Node.js має розширену бібліотеку модулів, що допомагає розробникам вирішувати різні завдання без необхідності створення всього з нуля.
- Спільнота та підтримка: Node.js має активну та велику спільноту розробників, яка надає підтримку та створює допоміжні інструменти та модулі.
- Недоліки Node.js:
- Один потік: Оскільки Node.js працює в одному потоці, велике обчислювальне навантаження може призвести до блокування інших подій.
- Не підходить для CPU-інтенсивних завдань: Node.js оптимізований для асинхронної обробки вводу-виводу і не найкращий вибір для завдань, які потребують великої обчислювальної потужності.

Порівняння Node.js з аналогами представлено на таблиці 3.3.

Таблиця 2.4 – Порівняльна таблиця Node.js з аналогами:

Особливості	Node.js	Django	Ruby on Rails	ASP.NET
Мова програмування	JavaScript	Python	Ruby	C#
Тип сервера	Івент-драйв (event-driven)	Заснований на WSGI (event-driven)	Заснований на Rack (MVC)	Івент-драйв (event-driven)

Таблиця 2.4 (продовження)

Фреймворк	Немає жорсткого фреймворка, але є багато пакетів та бібліотек	Django	Ruby on Rails	ASP.NET MVC, ASP.NET Web API
Спрощений веб-сервер	Так	Так	Так	Так
Вбудована система шаблонів	Ні	Так	Так	Так
Асинхронність	Так	Ні	Ні	Так
Споживання ресурсів	Низьке	Середнє	Високе	Залежить від конфігурації та платформи
Спільнота	Дуже велика	Велика	Велика	Велика

2.8 Фреймворк Express.js

Express.js - це мінімалістичний та гнучкий веб-фреймворк для Node.js, який допомагає створювати веб-додатки та API. Він є одним з найпопулярніших інструментів для створення серверної частини додатків на Node.js. Давайте розглянемо переваги та недоліки Express.js, а також зробимо порівняльну таблицю з аналогами.

Переваги Express.js:

- **Спрощений маршрутизатор:** Express надає простий та зрозумілий маршрутизатор для обробки запитів, що робить його дуже доступним для розробників.
- **Активна спільнота та плагіни:** Express має велику спільноту розробників, яка підтримує та розширює його функціональність через плагіни та доповнення.
- **Підтримка middleware:** Middleware дозволяє легко додавати функціональність до додатків Express, таку як аутентифікація, обробка помилок та журналювання.
- **Велика кількість інтеграцій:** Express може легко інтегруватися з іншими інструментами та базами даних, що полегшує розробку.

Недоліки Express.js:

- **Брак стандарту:** Однією з недоліків Express є відсутність стандарту, тому розробники повинні вирішувати багато архітектурних питань самостійно.
- **Необхідність сторонніх модулів:** Деякі функції, які інші веб-фреймворки можуть надавати "з коробки," в Express потребують встановлення сторонніх модулів.

Порівняння Express.js з аналогами представлено на таблиці 2.5.

Таблиця 2.5 - Порівняльна таблиця Express.js з аналогами:

Характеристика	Express.js	Koa.js	Hapi.js	Nest.js
Спрощеність використання	Дуже простий та легкий у використанні.	Заснований на промісах, дозволяє більше контролю.	Імплементує багато вбудованих функцій, що може робити інтеграцію більшою, але збільшує складність.	Пропонує вбудовану підтримку для розподіленого розвитку та використання TypeScript.
Модульність	Добре структурований, але менше модульний порівняно з іншими.	Більш модульний завдяки асинхронному стилю та мідлвару.	Модульний, дозволяє вам використовувати лише те, що потрібно.	Сильно модульний, використовує модульність Angular.
Продуктивність	Досить висока продуктивність	Має швидке виконання завдяки асинхронності.	Добра продуктивність, особливо для більших застосунків.	Забезпечує високу продуктивність завдяки використанню TypeScript та вбудованим оптимізаціям.

Таблиця 2.5 (продовження)

Підтримка Middleware	Багато готових мідлварів, широкий вибір.	Має менше готових мідлварів, але легко інтегрується з іншими.	Має велику бібліотеку мідлварів.	Забезпечує готовий набір мідлварів.
Активність розвитку	Активний розвиток та велика спільнота.	Активний розвиток, але менше популярний.	Припинено розвиток в 2021 році.	Швидкий розвиток та регулярні оновлення.

Загалом, Express.js є популярним та потужним веб-фреймворком для розробки серверної частини додатків на Node.js. Вибір між Express та іншими аналогами залежить від конкретних потреб та вимог проекту.

2.9 База даних PostgreSQL

PostgreSQL, часто називається "Postgres", виникла в 1986 році на кафедрі комп'ютерних наук у Колумбійському університеті. Michael Stonebraker і його команда розпочали роботу над поєднанням можливостей традиційних реляційних баз даних з підтримкою об'єктно-орієнтованих можливостей.

У 1996 році проект був перейменований в PostgreSQL, для визначення його підтримки мови SQL (SQL - Structured Query Language).

PostgreSQL - це продукт з відкритим вихідним кодом, що означає, що його вихідний код доступний для загального використання та модифікації.

PostgreSQL підтримує розширення стандарту SQL, включаючи продвинуті функції та операції.

Додаткові модулі та розширення можна легко додавати, що дозволяє користувачам адаптувати базу даних до своїх потреб.

PostgreSQL відзначається високою ступенем безпеки та надійності завдяки своїй підтримці ACID (Atomicity, Consistency, Isolation, Durability).

База даних може працювати в розподіленому середовищі та забезпечує можливість реплікації для створення резервних копій та забезпечення вищої доступності.

PostgreSQL має велику та активну спільноту користувачів та розробників, які постійно вносять внески у розвиток та підтримку системи.

PostgreSQL визначається своєю гнучкістю, надійністю та відкритістю, що робить його популярним в різноманітних проектах, від малих веб-додатків до великих корпоративних систем управління базами даних.

Переваги PostgreSQL:

- **Відкритий вихідний код:** PostgreSQL є відкритим вихідним кодом, що дозволяє користувачам безкоштовно використовувати, змінювати та розповсюджувати його.
- **Розширені можливості SQL:** PostgreSQL підтримує багато функцій та операцій стандарту SQL, а також надає ряд додаткових можливостей, таких як віконні функції та розширені типи даних.
- **Розширюваність:** Є можливість додавати власні функції, типи даних та мови програмування, що робить PostgreSQL дуже гнучким.
- **Транзакційна безпека:** PostgreSQL підтримує транзакції з рівнем ізоляції та механізми відновлення, що забезпечує консистентність та надійність даних.
- **Розподілені системи та Реплікація:** PostgreSQL підтримує роботу в розподіленому середовищі та має вбудовані можливості реплікації для створення резервних копій та забезпечення вищої доступності.

- **Активна Спільнота:** Існує широка та активна спільнота користувачів та розробників, що сприяє постійному розвитку та підтримці системи.

Недоліки PostgreSQL:

- **Вище споживання ресурсів:** Деякі операції можуть вимагати більше ресурсів порівняно з іншими базами даних, що може впливати на продуктивність в обмежених ресурсах середовища.
- **Складність установки та конфігурації:** Настроювання та установка PostgreSQL може виявитися складним завданням для новачків у порівнянні з іншими системами управління базами даних.
- **Відсутність графічного інтерфейсу за замовчуванням:** PostgreSQL не надає повноцінного графічного інтерфейсу для адміністрування, хоча існують сторонні інструменти для цього.
- **Обмеження щодо масштабування на рівні горизонталі:** Хоча PostgreSQL підтримує реплікацію, масштабування на рівні горизонталі може виявитися викликом для деяких сценаріїв.
- **Потреба відокремленого сервера для кожного клієнта:** Кожен клієнт, що взаємодіє з PostgreSQL, пов'язаний із окремим процесом сервера, що може викликати певні обмеження на великих навантажених системах.

Порівняння PostgreSQL з аналогами представлено на таблиці 2.6.

Таблиця 2.6 - Порівняльна таблиця PostgreSQL з аналогами:

Особливості	PostgreSQL	MySQL	Oracle DB	MSSQL
Тип системи управління базами даних	Об'єктно-реляційна (ORDBMS)	Реляційна (RDBMS)	Об'єктно-реляційна (ORDBMS)	Реляційна (RDBMS)
Відкритий вихідний код	Так	Так	Ні	Ні

Таблиця 2.6 (продовження)

Мова програмування	C	C/C++	C/C++, Java	C/C++, .NET
Підтримка стандарту SQL	Так	Так	Так	Так
Розширені можливості SQL	Так	Так	Так	Так
Розширюваність	Так	Так	Так	Так
Транзакційна безпека	Так	Так	Так	Так
Реплікація	Так	Так	Так	Так
Робота в розподіленому середовищі	Так	Так	Так	Так
Спільнота	Активна	Дуже активна	Дуже активна	Дуже активна
Ліцензія	PostgreSQL License	GNU GPL	Proprietary	Proprietary

2.10 Бібліотека Material UI

Material-UI - це бібліотека компонентів інтерфейсу користувача (UI) для React, яка реалізує дизайн-мову Material Design від Google. Material Design відомий своєю чистотою, зрозумілістю та сучасністю, і Material-UI допомагає розробникам легко і ефективно інтегрувати ці елементи дизайну у свої веб-проекти.

Доцільність використання Material-UI визначається кількома перевагами:

- **Зовнішній вигляд і стиль:** Material-UI надає зовнішній вигляд та стиль, відповідні стандартам Material Design, що дозволяє швидко створювати сучасні та красиві інтерфейси.

- **Гнучкість та налаштовуваність:** Бібліотека пропонує широкий вибір готових компонентів, які можна легко адаптувати та налаштовувати відповідно до потреб проекту.
- **Висока продуктивність:** Material-UI розроблено для оптимізації продуктивності та ефективності використання ресурсів.
- **Активна спільнота та підтримка:** Як популярна бібліотека, Material-UI користується активною спільнотою розробників, а це означає швидку виправлення помилок, постійне оновлення та розвиток.
- **Інтеграція з React:** Оскільки Material-UI побудовано на базі React, він легко інтегрується з проектами, які вже використовують цю бібліотеку для розробки інтерфейсу.
- **Доступність:** Material-UI дбає про доступність і надає інструменти для створення веб-сайтів, які відповідають стандартам доступності.

Узагальнюючи, Material-UI - це потужний інструмент для розробки інтерфейсів, який спрощує процес створення зручних та естетичних веб-додатків, дозволяючи розробникам фокусуватися на функціональності, а не на вигляді.

Розробники віддають перевагу внедренню Material-UI (MUI) у свої додатки з кількох причин. По-перше, MUI пропонує різноманіття готових компонентів користувальницького інтерфейсу, сприяючи створенню візуально привабливих та дружелюбних дизайнів з швидкою реалізацією. Використання принципів Material Design, таких як використання Material Icons, дозволяє розробникам гармонійно впроваджувати свої системи дизайну, що надає йому перевагу, показану в таблиці 2.7.

Таблиця 2.7 – Порівняльна таблиця Material-UI з його аналогами.

Характеристика	Material-UI	Ant Design	Bootstrap
Дизайн	Material Design від Google	Сучасний та гнучкий стиль	Простий та класичний дизайн
Гнучкість та налаштовуваність	Задовольняє потреби	Велика налаштовуваність	Висока гнучкість і налаштовуваність
Засоби компонентів	Широкий вибір, добре документовані	Багато компонентів, добре документовані	Великий асортимент, добре документовані
Інтеграція з React	Тісна інтеграція, розроблено на базі React	React-специфічний, зручний для React проектів	React-специфічний, можливості для React
Спільнота та підтримка	Велика активна спільнота	Активна та розвинута	Широка та досвідчена
Ресурсоємність	Помірна	Помірна	Помірна

Крім того, MUI надає можливість налаштування теми, що дозволяє легко встановлювати та глобально впроваджувати її для всіх компонентів. Це забезпечує високофункціональний та динамічно налаштовуваний досвід, що дозволяє розробникам однаково налаштовувати кольори теми, інформацію про палітру, властивості поверхні та інші стилі у всіх компонентах.

Крім того, Material-UI відзначається чіткою та зрозумілою документацією, яка містить чіткі інструкції разом з практичними прикладами коду. Цей ресурс є цінним для розробників, які шукають всебічну допомогу.

Крім того, MUI користується широкою підтримкою з регулярними оновленнями, виправленнями помилок та відзивчивою командою. Розробники можуть активно брати участь у формуванні майбутніх доповнень до бібліотеки через опитування зворотнього зв'язку та безпосередній зв'язок через офіційний обліковий запис в Twitter (@MaterialUI). Активна спільнота, яка оточує Material-UI, також надає необхідні посилання для підтримки та приклади використання. Загалом, поєднання готових компонентів, принципів Material Design, налаштовуваних тем, всебічної документації та сильної підтримки спільноти робить Material-UI переважним вибором для багатьох розробників.

2.11 Об'єктно-реляційний мапер Sequelize

Sequelize - це об'єктно-реляційний мапер (ORM) для Node.js, який дозволяє взаємодіяти з базами даних за допомогою JavaScript-об'єктів. Він підтримує різні системи управління базами даних (СУБД), такі як PostgreSQL, MySQL, SQLite та MSSQL, та забезпечує зручний спосіб взаємодії з базами даних, використовуючи об'єктно-орієнтований підхід.

Доцільність використання Sequelize визначається кількома перевагами:

- **Простота використання:** Sequelize дозволяє описувати моделі бази даних за допомогою простого та зрозумілого синтаксису JavaScript, що робить його доступним для розробників навіть з обмеженим досвідом роботи з базами даних.
- **Підтримка різних СУБД:** Sequelize є агностичним до СУБД і підтримує кілька популярних систем управління базами даних, що дозволяє розробникам вибирати підходящий варіант для свого проекту.

- **Синхронізація моделей і бази даних:** Sequelize може автоматично створювати таблиці та поля бази даних на основі описаних моделей, що полегшує розгортання та оновлення бази даних.
- **Підтримка відносин між таблицями:** Sequelize дозволяє легко визначати та використовувати відносини між таблицями, що спрощує роботу зі складними структурами даних.
- **Повний набір операцій бази даних:** Sequelize надає розширений набір операцій для вибірки, вставки, оновлення та видалення даних з бази даних, що дозволяє ефективно взаємодіяти з даними.

Порівняння Sequelize з аналогами представлено на таблиці 3.7.

Таблиця 2.8 – Порівняльна таблиця Sequelize з його аналогами.

Характеристика	Sequelize	TypeORM	Prisma
Тип бази даних	Підтримує різні СУБД, такі як PostgreSQL, MySQL, SQLite, MSSQL	Підтримує різні СУБД, такі як PostgreSQL, MySQL, SQLite, MSSQL, та інші	Орієнтовано на PostgreSQL, MySQL, SQLite
Синтаксис опису моделей	JavaScript	TypeScript та JavaScript	TypeScript та JavaScript
Спосіб міграцій	sequelize-cli	TypeORM CLI	Вбудований механізм міграцій
Відносини	Підтримка відносин між моделями за допомогою асоціацій	Підтримка відносин між моделями, можливість визначення відносин	Декларативна система відносин, визначається у схемі

Таблиця 2.8 (продовження)

Зручність використання	Зазвичай вважається простим та добре документованим	Має крок за кроком навчання та докладну документацію	Завдяки CLI та структурі коду, сприяє зручності
Видання або оновлення	Регулярні випуски з новими функціями та виправленнями	Активна розробка, нові випуски регулярно	Постійна розробка та виправлення помилок

Sequelize став популярним інструментом в світі Node.js розробки завдяки своїй простоті використання, гнучкості та великому спектру функцій, що роблять його доцільним для проектів різного масштабу та складності.

2.12 Стейт-менеджер Effector.js

Effector.js - це бібліотека для управління станом в JavaScript-додатках. Вона розроблена для ефективної роботи зі станом, враховуючи принципи реактивного програмування. Основною метою Effector є полегшення створення масштабованих та продуктивних застосунків.

Основні характеристики Effector.js:

- **Реактивність:** Effector спрямований на реактивний підхід до програмування, де зміни стану спричинюють автоматичні оновлення відповідних компонентів чи функцій.
- **Декларативний синтаксис:** Бібліотека надає декларативний синтаксис для опису логіки та взаємодії між частинами програми, що полегшує зрозуміння та підтримку коду.

- **Єдина точка управління станом:** Використовуючи Effector, ви працюєте зі станом через створені об'єкти, які є централізованими точками для управління станом всього додатку.
- **Ефекти та обробники:** Effector дозволяє легко визначати та обробляти побічні ефекти (effects), такі як асинхронні запити чи обробка даних, що робить його дуже потужним інструментом для комплексних додатків.
- **Масштабованість:** Effector надає можливості для створення великих та масштабованих додатків, полегшуючи організацію та управління станом в середніх і великих проектах.

Доцільність використання Effector.js визначається масштабністю та складністю вашого додатка. Він може бути особливо корисним для великих проектів, де ефективне управління станом та побічними ефектами стає пріоритетом. Однак для менших проектів чи тих, що не вимагають реактивного підходу, інші бібліотеки стейт-менеджменту або вбудовані засоби React можуть бути більш придатними.

2.13 Лінтер ESLint

ESLint - це статичний аналізатор коду для JavaScript, який допомагає виявляти та виправляти помилки, дотримуватися стандартів коду та покращувати загальну якість коду. Його основною метою є автоматизація процесу виявлення помилок та забезпечення спільних конвенцій в коді в командному проекті чи на рівні всього проекту.

Основні функції та переваги ESLint:

- **Виявлення помилок:** ESLint допомагає виявляти помилки та проблеми в коді на ранніх етапах розробки, що полегшує виправлення їх ще до запуску додатку.

- **Стандарти коду:** Ви можете налаштувати ESLint відповідно до конкретних стандартів коду або використовувати популярні конфігурації (наприклад, Airbnb, Standard, тощо). Це допомагає забезпечити єдність та консистентність у коді команди.
- **Кастомізація:** ESLint надає гнучкі можливості кастомізації. Ви можете додавати свої правила, виключати або включати певні правила залежно від потреб проекту.
- **Інтеграція з редакторами:** ESLint легко інтегрується з багатьма редакторами та інтегрованими середовищами розробки (IDE), що дозволяє вам бачити та виправляти помилки прямо в процесі написання коду.
- **Підтримка ESNext та JSX:** ESLint підтримує сучасні функції JavaScript (ES6, ES7, ES8, і т.д.), а також взаємодіє з JSX, що дозволяє використовувати його в розробці React-застосунків.
- **Автоматична поправка:** ESLint може автоматично виправляти багато з виявлених ним помилок, що полегшує процес покращення якості коду.

Доцільність використання ESLint визначається величезною користю, яку він приносить в процес розробки. Використання ESLint рекомендується для будь-якого проекту, незалежно від його розміру. Він допомагає уникати загальних помилок, покращує зрозумілість та читабельність коду, а також сприяє впровадженню кращих практик та стандартів в команді розробників.

2.14 Бібліотека Jsonwebtoken

Jsonwebtoken - це бібліотека для створення та перевірки JSON Web Tokens (JWT) у JavaScript-додатках. JWT - це стандарт, який визначає компактний та самостійний спосіб представлення інформації між сторонами у вигляді об'єкта.

Використання JSON Web Tokens стало популярним для реалізації механізмів аутентифікації та авторизації в веб-застосунках та API.

Основні принципи та можливості jsonwebtoken:

- **Створення та Підписання токенів:** jsonwebtoken дозволяє створювати JWT токени та підписувати їх за допомогою секретного ключа чи приватного ключа (у випадку використання асиметричного шифрування).
- **Валідація та Розшифрування токенів:** Бібліотека дозволяє виконувати верифікацію та розшифрування токенів за допомогою секретного ключа чи публічного ключа.
- **Кастомізація та Керування Правами:** jsonwebtoken дозволяє включати користувацькі дані (claims) у токени, що може використовуватися для передачі інформації про користувача та його права.
- **Підтримка Асиметричного та Симетричного Шифрування:** Бібліотека може працювати як з асиметричними (RSA, ECDSA) так і симетричними (HS256, HS384, HS512) алгоритмами шифрування.
- **Захист від Маніпуляцій:** JSON Web Tokens мають вбудований механізм підпису, який дозволяє перевірити, чи був токен змінений після його створення.

Доцільність використання jsonwebtoken визначається необхідністю в реалізації безпечного та зручного механізму аутентифікації та авторизації в додатку. Використання JWT дозволяє передавати авторизаційну інформацію у форматі, який легко передавати та перевіряти, а також використовувати на клієнтській та серверній сторонах. Однак слід враховувати, що зберігання секретного ключа має бути відповідно захищеним, і у деяких випадках використання HTTPS є обов'язковим для забезпечення безпеки передачі токенів.

2.15 Висновки до розділу

Кожна складова розробленого додатку налаштована для виконання своєї роботи: клієнтська частина(Веб-додаток) дає користувачу доступ до візуального уявлення моделей та функціоналу для їх роботи, в той час серверна відповідає за роботу з базою даних, де зберігаються дані користувача. Стейт-менеджер зберігає та модифікує інформацію для задання конфігурацій та передачу даних до серверу за стандартом JWT.

Локальний сервер розробки Vite забезпечив швидку та зручну розробку, а його відмінна продуктивність сприяла впровадженню інновацій та оптимізації робочого процесу. Використання бібліотеки ReactJS та фреймворку Node.js значно полегшило розробку користувацького і серверного шарів відповідно, забезпечуючи високу якість та швидкодіючість програмного забезпечення.

Фреймворк Express.js додатково забезпечив надійність та ефективність серверної частини додатку, а використання бази даних PostgreSQL виявилося доречним рішенням для забезпечення надійності та масштабованості системи управління даними.

3 ВІЗУАЛІЗАЦІЯ ПРОЦЕСУ ПОШИРЕННЯ ВІРУСУ

3.1 Постановка задачі та побудова моделі

Побудуємо математичну динамічну модель процесу поширення та існування вірусу за допомогою клітинних автоматів.

Постановка задачі:

- Вірус може вижити лише в випадку його подальшого розповсюдження.
- Вірус мутує (переходить до наступного етапу) за умови присутності 2-х заражених осіб навколо нього, або коли всі особи заражені.
- Вірус знаходиться в замкнутому приміщенні квадратної форми. Кількість людей в приміщенні відома – 49.
- Початкова кількість заражених вірусом людей обумовлена випадковими обставинами

Задача: Побудувати візуалізацію моделі процесу поширення вірусу серед людей. Встановити залежність між етапами перетворення вірусу та початкової кількості таких людей.

З точки зору клітинних автоматів, за умови, що заражена вірусом людина відіграє роль клітини, можемо встановити наступні три правила:

1. Правила продовження життя – класичні. Якщо навколо 1 чи 8 живих клітин- клітина вмирає.
2. Початкове поле 7 на 7 (площа- 49 одиниць)
3. Нехай k -початкова кількість клітин, що генеруються рандомом, а m-кількість ітерацій необхідних для переходу автомату до стабільного положення.

3.2 Статистичні дані вибірки

За допомогою розробленого програмного забезпечення зафіксуємо та занесемо до таблиці значення початкової кількості клітин та необхідної кількості ітерацій до моменту переходу в стабільний стан системи, за умови, що розмір поля – 7x7.

Процес генерації та фіксування повторюємо 15 разів. Результати занесено до таблиці 3.1.

Таблиця 3.1 – Результати процесу генерації

Початкова кількість клітин	Кількість Ітерацій
17	43
12	25
13	27
17	18
14	13
15	15
9	7
15	11
18	20
17	12
16	30
18	11
13	22
20	13
18	22

3.3 Встановлення взаємозв’язку за допомогою регресійного аналізу

Рівняння парної регресії.

Нехай початкова кількість клітин це змінна x, а кількість ітерацій -змінна y.

Для встановлення виду рівняння регресії використовуємо метод кінцевих різниць.

Для цього визначимо темп поширення вірусу в таблиці 3.2.

Таблиця 3.2 – Темп поширення вірусу

y_i	Δ^1_t	Δ^2_t	Темп роста
0	-	-	-
43		-	
25	-18	-18	0.581
27	2	20	1.08
18	-9	-11	0.667
13	-5	4	0.722
15	2	7	1.154
7	-8	-10	0.467
11	4	12	1.571
20	9	5	1.818
12	-8	-17	0.6

Таблиця 3.2 (продовження)

30	18	26	2.5
11	-19	-37	0.367
22	11	30	2
13	-9	-20	0.591
22	9	18	1.692

За отриманими даними визначаємо, що між значеннями у є лінійна залежність. Рівняння цієї залежності від початкової кількості клітин матиме вигляд: $y = bt + a$

Для розрахунку параметрів регресії побудуємо розрахункову таблицю 3.3.

Таблиця 3.3 – Розрахункова таблиця

x	y	x2	y2	x*y
0	0	0	0	0
17	43	289	1849	731
12	25	144	625	300
13	27	169	729	351
17	18	289	324	306
14	13	196	169	182
15	15	225	225	225
9	7	81	49	63
15	11	225	121	165
18	20	324	400	360
17	12	289	144	204
16	30	256	900	480
18	11	324	121	198
13	22	169	484	286

Таблиця 3.3 (продовження)

20	13	400	169	260
18	22	324	484	396
232	289	3704	6793	4507

Для наших даних система рівнянь має вигляд

$$16a + 232 - b = 289$$

$$232 - a + 3704 - b = 4507$$

Розвязуючи систему отримуємо значення для a та b.

b = 0.9309

a = 4.5647

Отримуємо коефіцієнти рівняння регресії:

b = 0.9309

a = 4.5647

Остаточний вигляд рівняння регресії:

$$y = 0.9309\,x + 4.5647$$

Знайдемо характеристики регресійного аналізу.

Вибіркові середні:

$$\bar{x} = \frac{\sum x_i}{n} = \frac{232}{16} = 14.5$$

$$\bar{y} = \frac{\sum y_i}{n} = \frac{289}{16} = 18.063$$

$$\overline{xy} = \frac{\sum x_i y_i}{n} = \frac{4507}{16} = 281.688$$

Вибіркові дисперсії:

$$S(x)^2 = \frac{\sum_{i=1}^{n} x_i^2}{n} - \bar{x}^2 = \frac{3704}{16} - 14.5^2 = 21.25$$

$$S(y)^2 = \frac{\sum_{i=1}^{n} y_i^2}{n} - \bar{y}^2 = \frac{6793}{16} - 18.0.63^2 = 98.31$$

Середньоквадратичні відхилення:

$$s_{(x)} = \sqrt{s^2(x)} = \sqrt{21{,}25} = 4.61$$

$$s_{(y)} = \sqrt{s^2(y)} = \sqrt{98.31} = 9.915$$

$$a = y - b \cdot x = 18.063 - 0.9309 \cdot 14.5 = 4.5647$$

Величину випадкової похибки обраховуємо за критерієм Стьюдента.

$$t_{nabl} = 0.433 \cdot \frac{\sqrt{14}}{\sqrt{1 - 0.433^2}} = 1.796$$

З рівнем значимості α=0.05 та кількість степенів вільності k=14 за таблицями визначаємо також критерій Стьюдента.

$$t\text{крит}(n - m - 1; \alpha/2) = t\text{крит}(14; 0.025) = 2.51$$

де m = 1 – кількість залежних змінних.

Оскільки $|t_{набл}| < t_{крит}$, то приймаємо гіпотизу рівності нулю коєфіцієнта кореляції. Тобто існує лінійний взаємозвяхок між змінними.

При аналізі якості моделі регресії використовується теорема про розкладання дисперсії, згідно з якою загальна дисперсія результативної ознаки може бути розкладена на дві складові – пояснену та непояснену рівнянням регресії дисперсії.

Завдання дисперсійного аналізу полягає в аналізі дисперсії залежною змінною:

$$\sum(yi - ycp)2 = \sum(y(x) - ycp)2 + \sum(y - y(x))2$$

Де ∑(yi - ycp)2 - загальна сума квадратів відхилень;

∑(y(x) - ycp)2 - сума квадратів відхилень, обумовлена регресією («пояснена» або «факторна»);

∑(y - y(x))2 - залишкова сума квадратів відхилень.

Висновки обчислень занесемо до таблиці 3.4.

Таблиця 3.4 – Висновки обчислень

Джерело варіації	Сума квадратів	Число ступенів свободи	Дисперсія на 1 ступінь свободи	F-критерій
Модель (пояснена)	294.624	1	294.624	3.227
Залишкова	1278.31	14	91.31	1
Загальна	1572.94	16-1		

Занесемо показники якості рівняння регресії до таблиці 3.5.

Таблиця 3.5 - Показники якості рівняння регресії

Показник	Значення
Коефіцієнт детермінації	0.1873
Середній коефіцієнт еластичності	0.747
Середня помилка апроксимації	Не був розрахований

Прогнозування 5 етапів розвитку вірусу

Інтервальний прогноз першого етапу розвитку:

$$t = 17: (-13.58; 46.24)$$

Точковий прогноз, t = 18: $y(18) = -0.204 * 18 + 19.8 = 16.12$

$$K_2 = 2.51 \cdot 10.55 \sqrt{1 + \frac{1}{16} + \frac{3(16 + 2 \cdot 2 - 1)^2}{16(16^2 - 1)}} = 30.52$$

$$16.12 - 30.52 = -14.4\,;\ 16.12 + 30.52 = 46.64$$

Інтервальний прогноз другого етапу розвитку:

$$t = 18\text{: } (-14.4; 46.64)$$

Точковий прогноз t = 19: $y(19) = -0.204 * 19 + 19.8 = 15.92$

$$K_3 = 2.51 \cdot 10.55 \sqrt{1 + \frac{1}{16} + \frac{3(16 + 2 \cdot 3 - 1)^2}{16(16^2 - 1)}} = 31.19$$

$$15.92 - 31.19 = -15.27\,;\ 15.92 + 31.19 = 47.11$$

Інтервальний прогноз третього етапу розвитку:

$$t = 19\text{: } (-15.27; 47.11)$$

Точечный прогноз, t = 20: $y(20) = -0.204 * 20 + 19.8 = 15.71$

$$K_4 = 2.51 \cdot 10.55 \sqrt{1 + \frac{1}{16} + \frac{3(16 + 2 \cdot 4 - 1)^2}{16(16^2 - 1)}} = 31.91$$

$$15.71 - 31.91 = -16.2\,;\ 15.71 + 31.91 = 47.62$$

Інтервальний прогноз четвертого етапу розвитку::

$$t = 20\text{: } (-16.2; 47.62)$$

Точечный прогноз, t = 21: $y(21) = -0.204 * 21 + 19.8 = 15.51$

$$K_5 = 2.51 \cdot 10.55 \sqrt{1 + \frac{1}{16} + \frac{3(16 + 2 \cdot 5 - 1)^2}{16(16^2 - 1)}} = 32.67$$

$$15.51 - 32.67 = -17.16\,;\ 15.51 + 32.67 = 48.18$$

Інтервальний прогноз п`ятого етапу розвитку:

$$t = 21\text{: } (-17.16; 48.18)$$

3.4 Висновки до розділу

Вивчено залежність Y від X. На етапі початкової обробки даних було обрано модель парної лінійної регресії на основі методу найменших квадратів. Оцінено її параметри також методом найменших квадратів: y = 0.931*x + 4.565 Статистична значущість рівняння перевірена за допомогою коефіцієнта детермінації та критерію Фішера. Встановлено, що у досліджуваної ситуації 18.73% загальної зміни етапів поширення вірусу Y пояснюється зміною початкової кількості заражених людей X. Встановлено також, що параметри моделі статистично значимі. Можлива економічна інтерпретація параметрів моделі - збільшення X на 1 од. призводить до збільшення Y в середньому на 0.931 од. Отримані оцінки рівняння регресії дають змогу використовувати його для прогнозу. При x=3, Y перебуватиме в межах від -8.76 до 23.47 од. та з ймовірністю 95% не вийде за ці межі.

Лінійний коефіцієнт кореляції дорівнює 0.433, отже, зв'язок між ознакою Y та фактором X помірна та пряма. Проте його значимість не підтверджується, тобто. між ознакою Y та фактором X зв'язок відсутній.

Аналіз коефіцієнта еластичності свідчить про несуттєвому впливі X на Y.

4 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Даний додаток може бути запущений на всіх комп’ютерах з присутнім браузером та наявним інтернет підключенням.

Створено інструкції, які дозволяють користувачам встановлювати систему, запускати її та користуватися всіма розробленими функціями.

4.1 Інсталяція програмного забезпечення

Для запуску додатку потрібно інсталювати всі npm-пакети в корінних директоріях та створити нову базу даних, ім’я якої буде збігатися з назвою бази даних в конфігураційному файлі під назвою .env. Для їх інсталювання пакетів потрібно мати завантажений NodeJS, який надає можливість працювати з пакетним менеджером npm. Після успішного завантаження всіх пакетів потрібно запустити серверну та клієнтську частини додатку за допомогою відповідної команди в файлі package.json. Далі сервер сам виконає всі налаштування по створенню потрібних таблиць в базі даних та зв’язків між ними.

4.2 Вимоги до програмного забезпечення комп’ютера

Дана система може працювати на всіх десктопних платформах як Windows, Linux та Mac. Наявність універсальної сумісності дозволяє користувачам обирати оптимальну операційну систему для їхніх потреб, забезпечуючи зручність та гнучкість у використанні програмного продукту.

Визначаючи ці вимоги, ми ставимо перед собою завдання забезпечити швидку та надійну роботу програмного забезпечення на будь-якому обраному комп'ютері, незалежно від його операційної системи.

Мінімальні потреби:

- Процесор: не менше 1 ГГц
- ОЗУ: 1 ГБ для 32-розрядної системи або 2 ГБ для 64-розрядної системи

4.3 Запуск системи

Для запуску системи потрібно просто перейти по URL, який отримуємо після запуску серверної та клієнтської частин.

Переглянемо приклад консолі із запущеним додатком на рисунку 4.1.

Рисунок 4.1 – Посилання на локальний сервер веб-додатку

4.4 Діаграма прецедентів

В основу роботи покладено важливі аспекти взаємодії користувача з інформаційною системою, що визначають функціональність та зручність її

використання. Діаграма прецедентів є ефективним інструментом для моделювання цих взаємодій, визначення ролей та визначення ключових можливостей системи.

Розглянемо діаграму прецедентів з головними функціями додатку на рисунку 4.2.

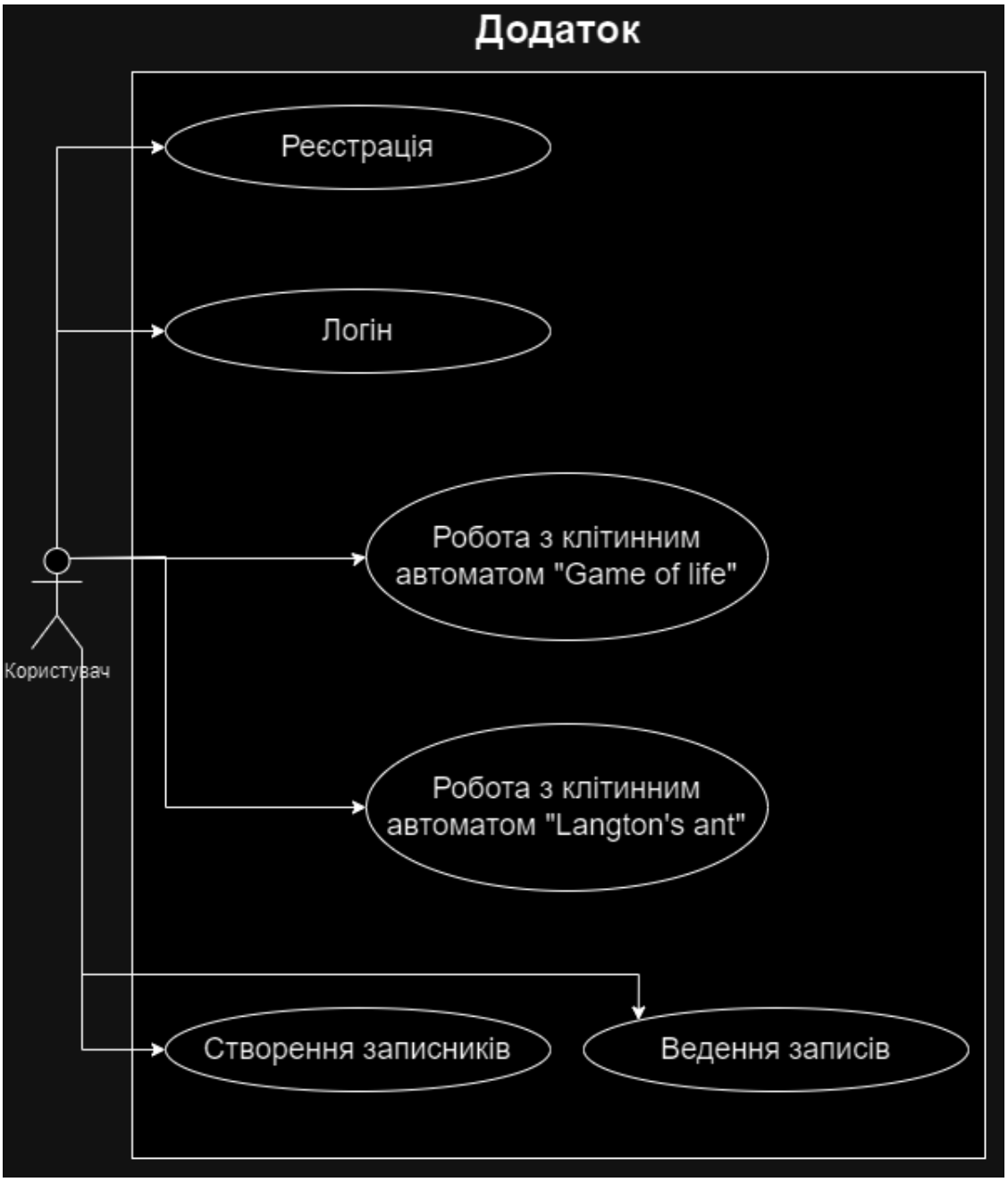

Рисунок 4.2 - Діаграма прецедентів

Реєстрація:

- Користувач може створити новий обліковий запис у системі.
- Введення основної інформації та обрання унікального імені та паролю.

Логін:

- Користувач може увійти в систему за допомогою свого облікового запису.
- Введення імені користувача та паролю для авторизації.

Робота з клітинним автоматом "Game of Life":

- Користувач може створити нову гру в клітинному автоматі "Game of Life".
- Введення початкового стану клітин, вибір параметрів гри (розмір поля, початкові умови і т.д.).
- Запуск та спостереження за еволюцією клітин.

Робота з клітинним автоматом "Langton's Ant":

- Користувач може створити нову гру в клітинному автоматі "Langton's Ant".
- Введення початкового стану автомата, вибір параметрів гри.
- Запуск та спостереження за рухом "мурахи" на полі.

Створення записників:

- Користувач може створювати особисті записники для зберігання своїх нотаток та інших даних.
- Введення заголовків та опису записника.

Ведення записів:

- Користувач може додавати нові записи до існуючих записників.
- Видалення записів, які вже існують.

Ця діаграма прецедентів відображає основні можливості системи, яка об'єднує роботу з клітинними автоматами та функціонал для ведення особистих

записів. Кожен прецедент визначає конкретні дії користувача та сприяє зручній та ефективній роботі з системою.

4.5 Демонстрація функціоналу

Після переходу по посиланню, ми попадаємо на головну сторінку додатку, та маємо змогу працювати з клітинними автоматами, зареєструватися та увійти в аккаунт.

Поки користувач не авторизований, він має доступ тільки до клітинних автоматів, без можливості ведення нотатків та можливості зберегти дані в профілі. Як тільки він пройде реєстрацію та увійде до аккаунту, то буде доступна нова вкладка “Notes”, де можна створювати записники та додавати записи про дослідження в них у вигляді записів.

На рисунку 4.3 ми бачимо сам початковий екран з клітинним автоматом “Langton`s ant” та панель навігації, яка закріплена зліва, та кнопку для реєстрації/входу, модальне вікно якої показано на рисунку 4.4.

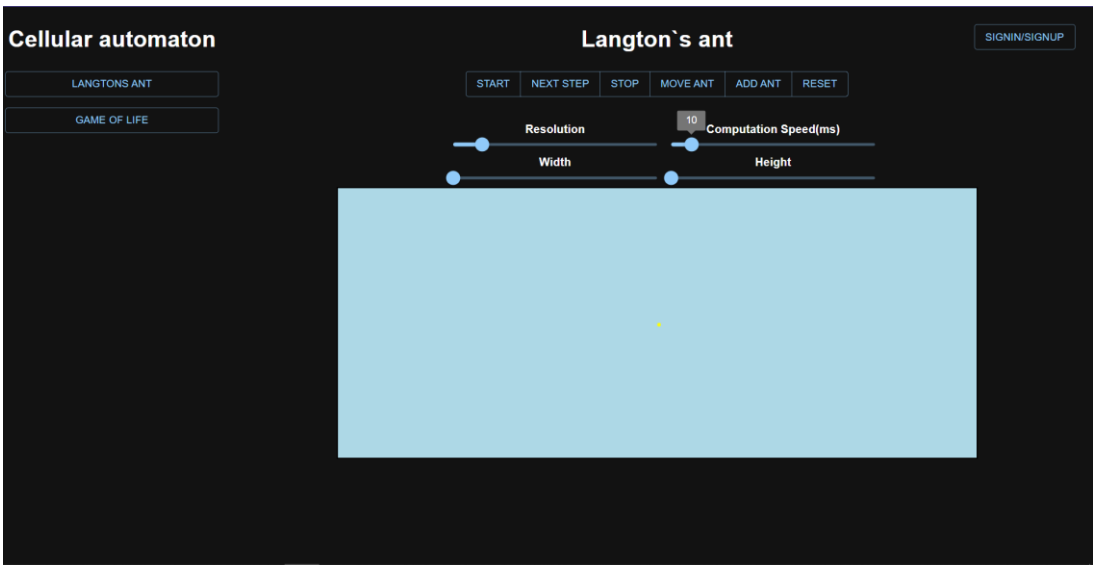

Рисунок 4.3 – Початковий екран веб-додатку

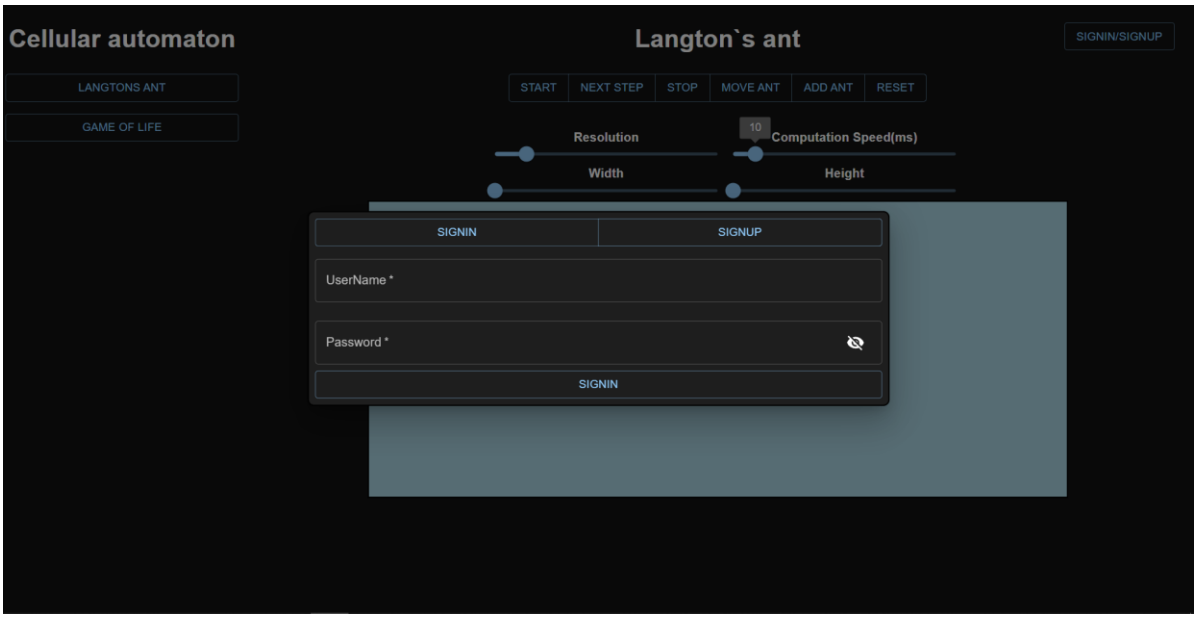

Рисунок 4.4 – Модальне вікно аутентифікації

В даному вікні користувач може вибирати між двох опцій: реєстрація(SignUp), створює новий аккаунт, та авторизація(SignIn), виконує вхід до існуючого акаунту.

На рисунку 4.5 показано вікно, після проходження процесу аутентифікації. Бачимо, що кнопка змінила текст, де вказано ім’я користувача, та можливість вийти з аккаунту. А також в панелі навігації з’явилась нова вкладка під назвою “Notes”.

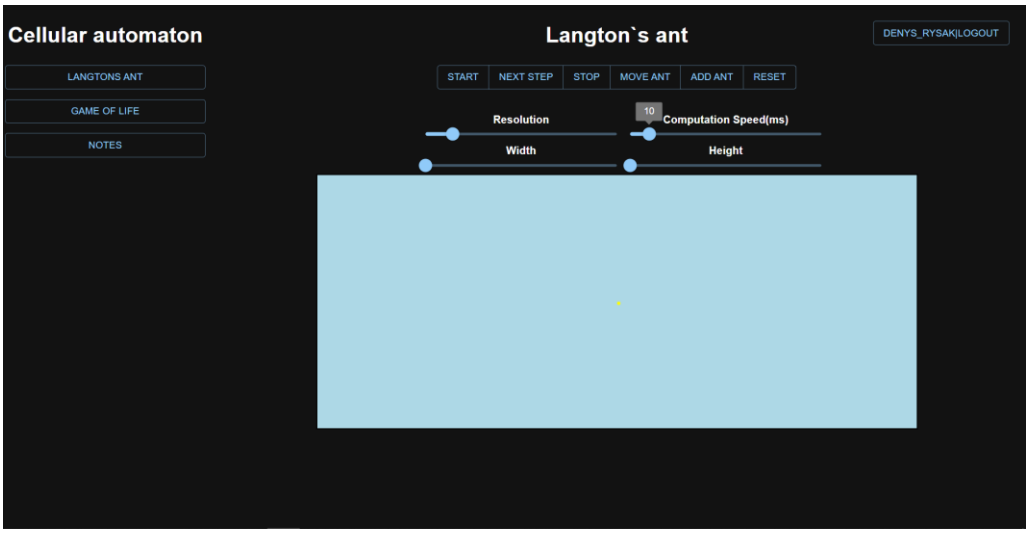

Рисунок 4.5 – Екран авторизованого користувача

Вкладка “Notes” містить в собі список записників, якщо хоть один створено, або екран, який показано на рисунку 4.6. Як бачимо, поки список записників пустий, але авторизований користувач має змогу створити новий, нажавши на відповідну кнопку, яка відкриває модальне вікно, аналогічне до вікна аутентифікації, показано на рисунку 4.7.

У модальному вікні є обов’язкові для заповнення поля: “Title” та “Description”.

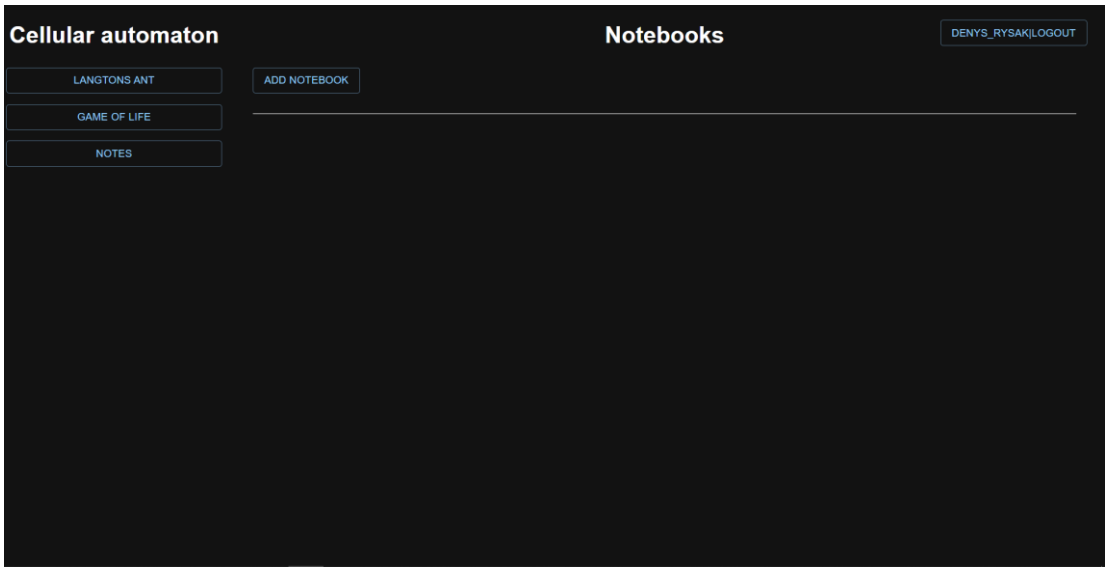

Рисунок 4.6 – Екран із записниками

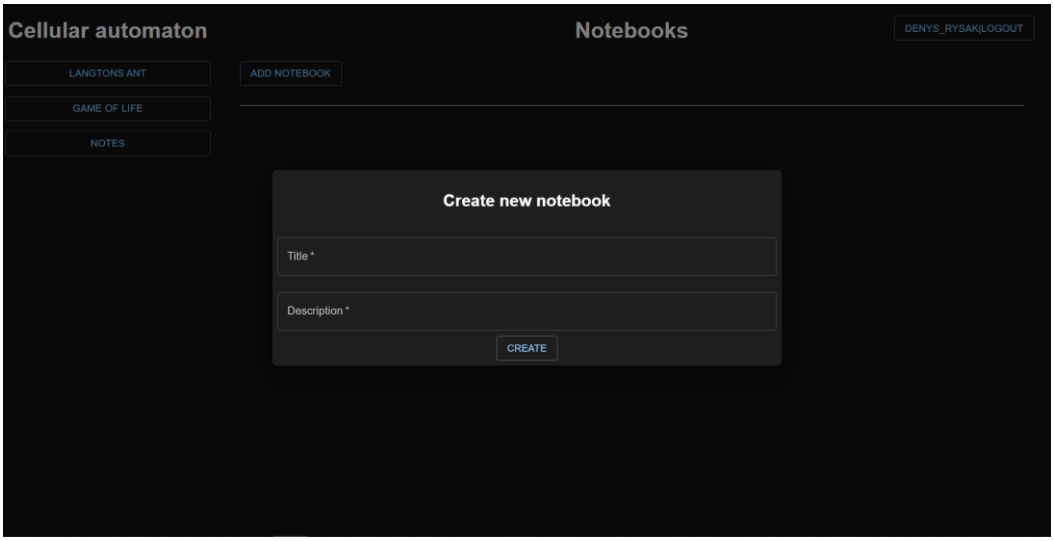

Рисунок 4.7 – Модальне вікно створення записника

Після створення записника відбудеться обновлення списку і користувачу виведеться новий список з новим записником. На рисунку 4.8 ми бачимо новий записник з його назвою, описом та можливістю його відкрити. Після натиску на кнопку “Open” користувач перейде на сторінку конкретного записника.

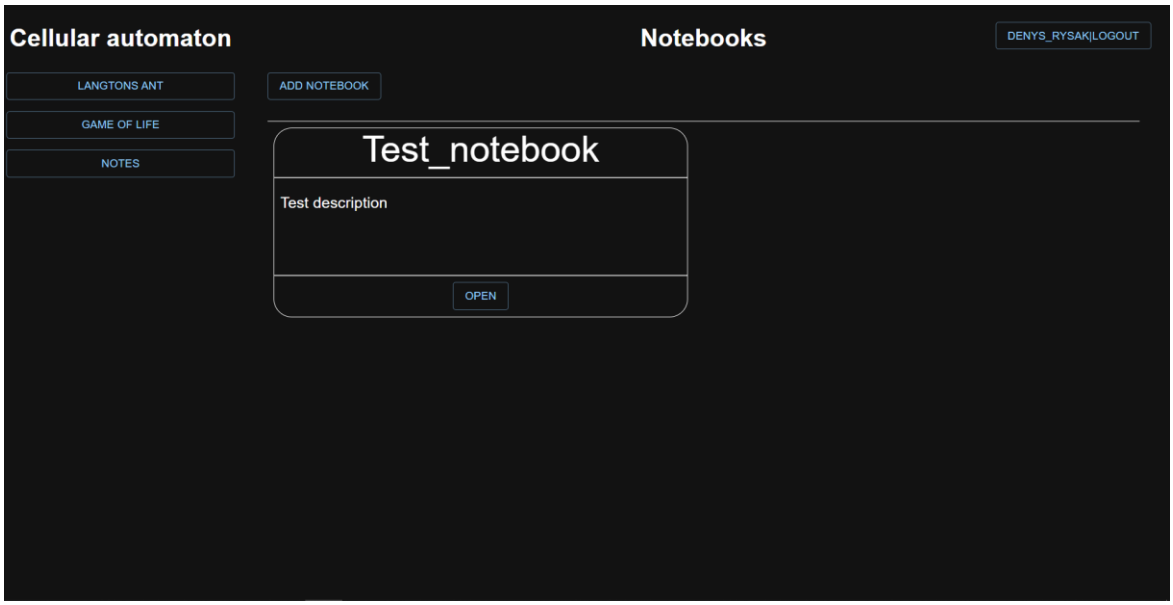

Рисунок 4.8 – Екран з новим записником

На рисунку 4.9 ми бачимо сторінку, де вказано його назву, панель з можливими операціями, які можна провести з цим записником та випадний список з опціями сортування записів записника.

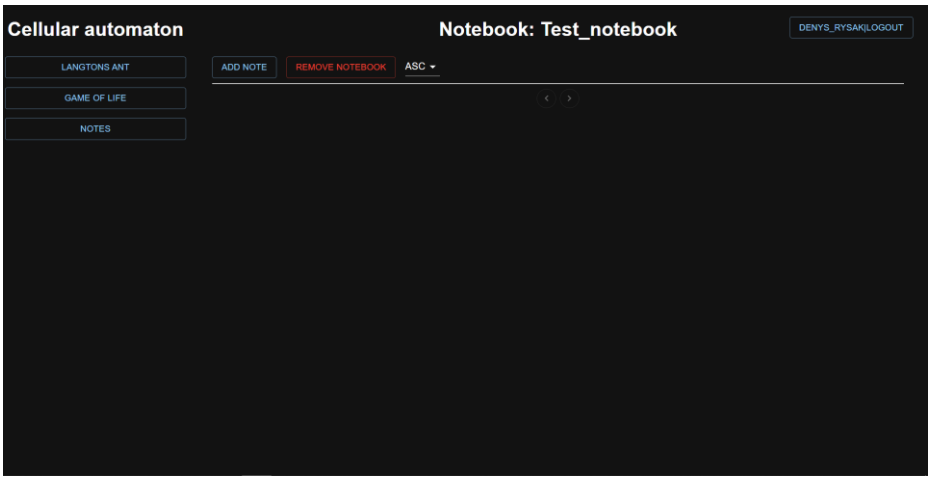

Рисунок 4.9 – Екран записника “Test_notebook”

На даному екрані користувач має змогу додати новий запис у записник та видалити його, нажавши на відповідну кнопку, а також відсортувати записи по даті створення.

Поки записник пустий, то записів не буде. Щоб створити новий запис, потрібно нажати на кнопку “Add note”, після чого відкриється модальне вікно, яке потрібно заповнити, показано на рисунку 4.10.

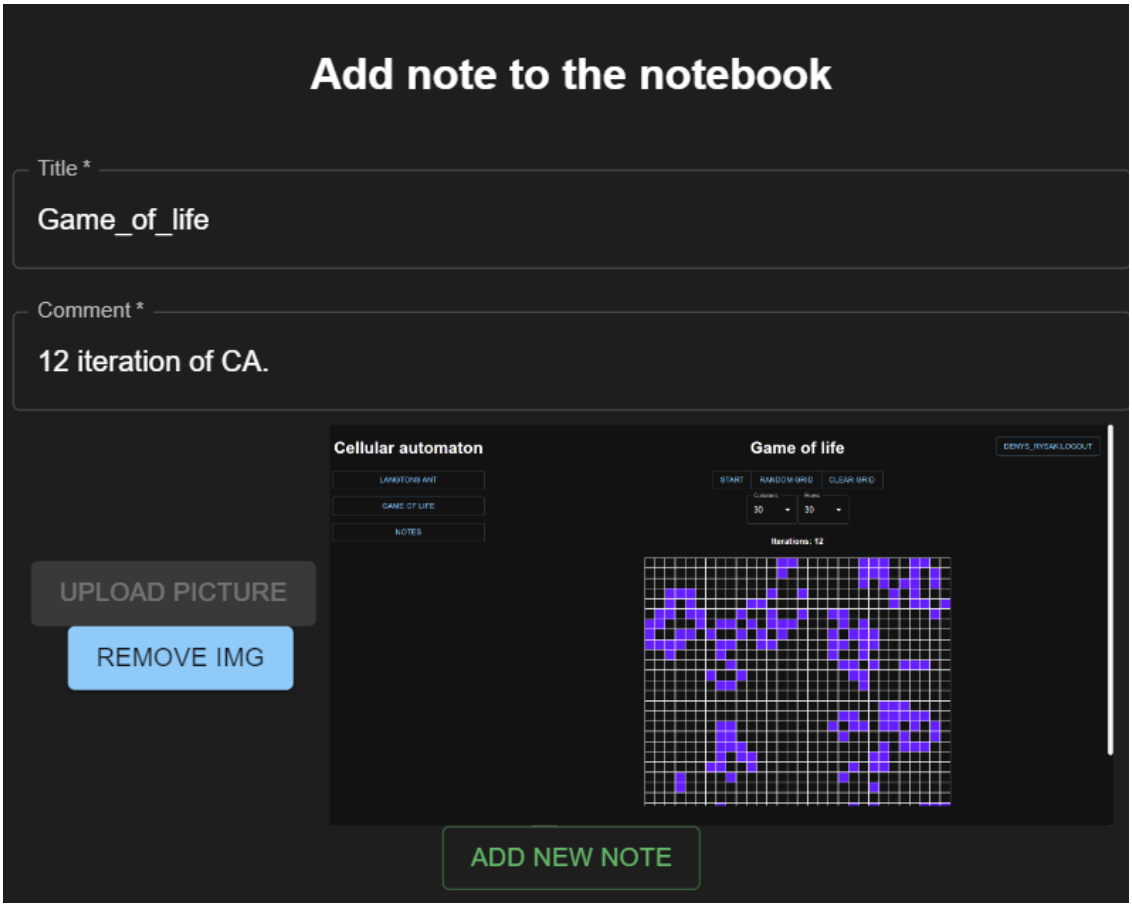

Рисунок 4.10 – Модальне вікно для створення запису

У модальному вікні потрібно вказати назву запису, та коментар до нього, а також додати скриншот роботи клітинного автомату, який хочемо зберегти.

Після цього список записів буде автоматично обновлено, та виведено новий запис у цьому списку. При великій кількості записів, їх буде подано порціями, а користувачу потрібно переміщатися сторінками за допомогою панелі знизу. Показано на рисунку 4.11.

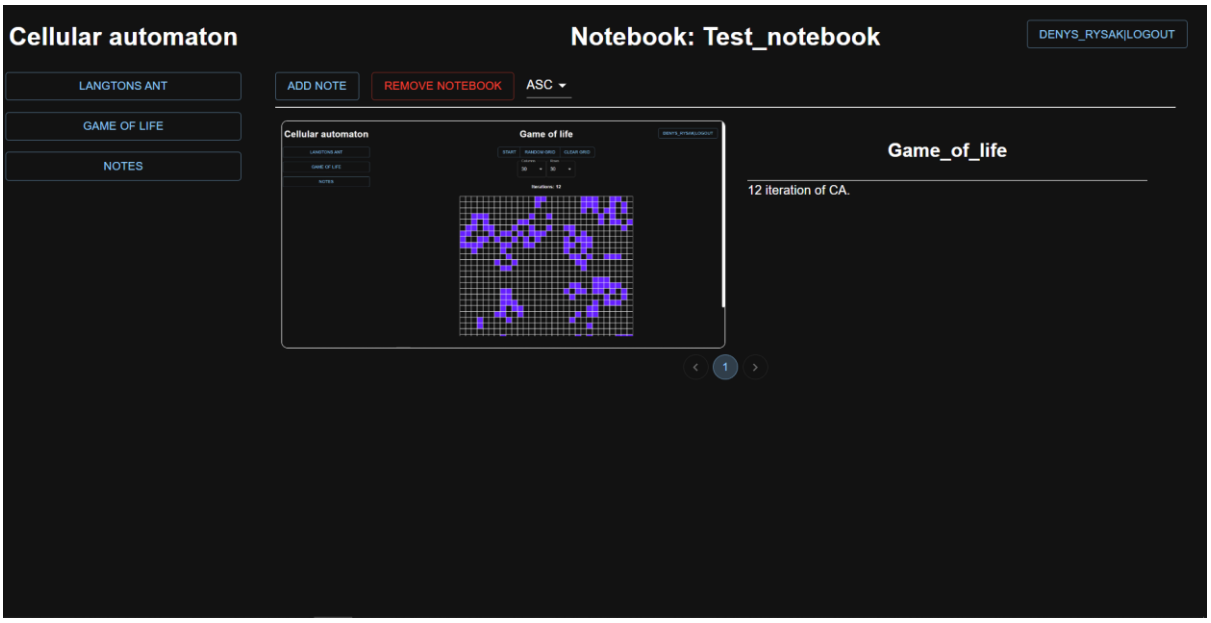

Рисунок 4.11 – Екран з записом та панеллю пагінації

Робота з клітинними автоматами відбувається за допомогою двох головних блоків: поля та набору кнопок управління. Показано на рисунку 4.12.

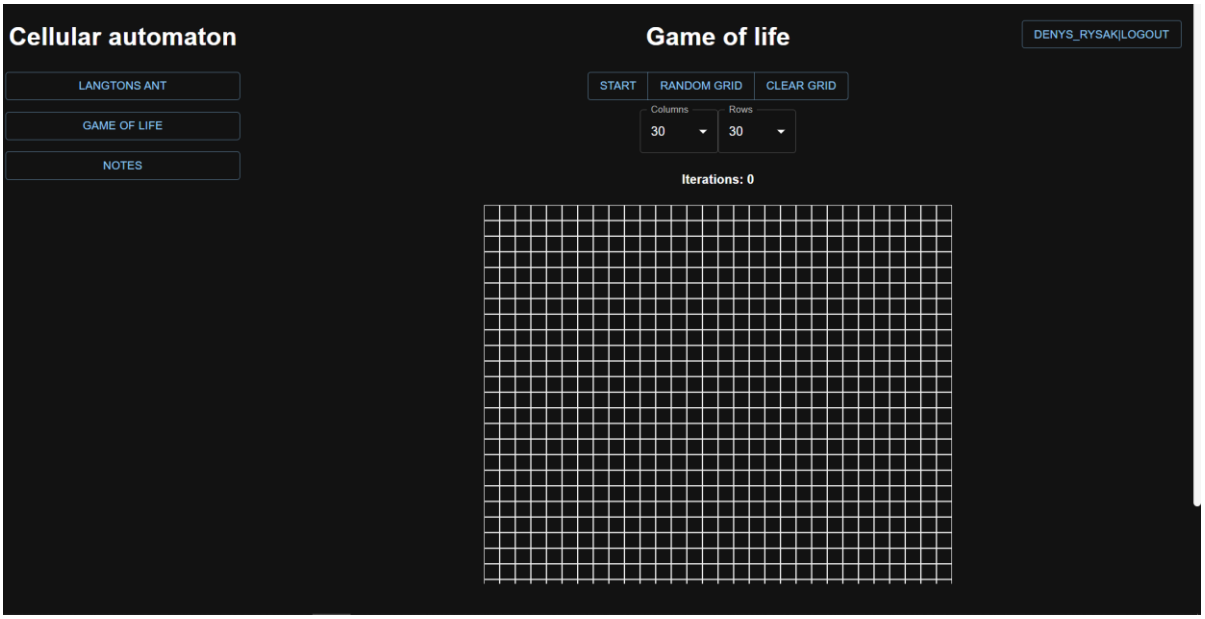

Рисунок 4.12 – Сторінка з клітинним автоматом “Game of life”

Приклад роботи клітинного автомату “Game of life” наведено на рисунку 4.13. На рисунку бачимо кількість ітерацій, які виконав алгоритм, та поле з

“клітинами”. Користувач може вибрати розмір поля за допомогою випадних списків, які описують кількість рядків та стовпців.

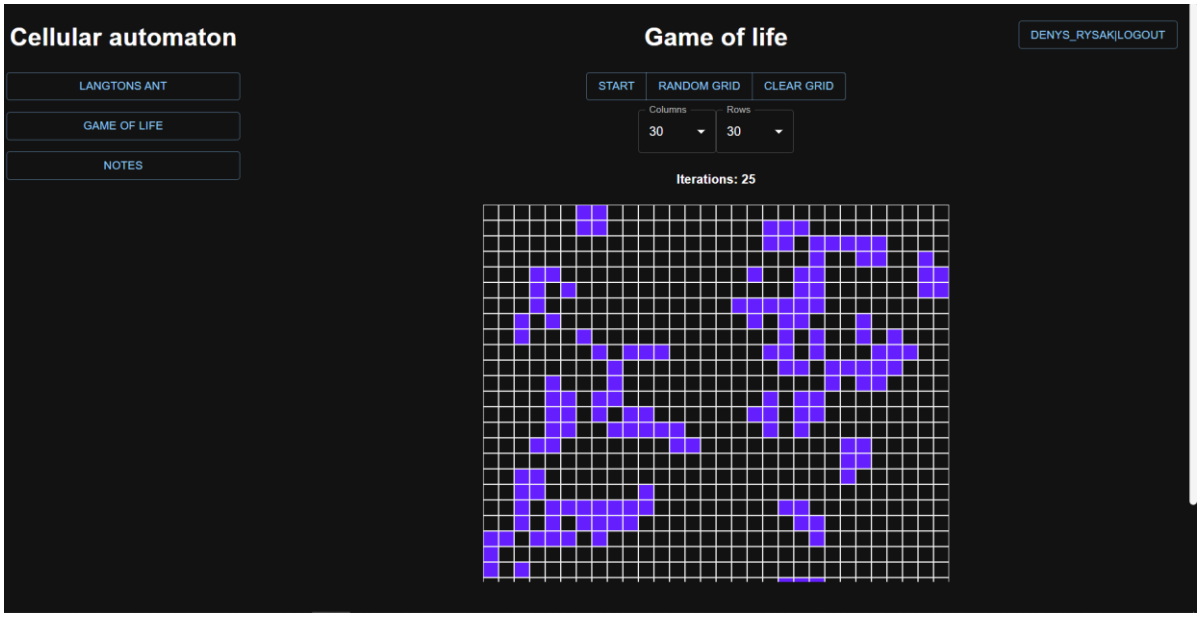

Рисунок 4.13 – Приклад роботи клітинного автомату “Game of life”

Приклад візуалізації розвитку моделі клітинного автомату “Langton`s ant” наведено на рисунку 4.14. Тут користувач має ті ж самі можливості та декілька унікальних, тільки для цього клітинного автомату. Регуляція ширини ра висоти поля відбувається за допомогою відповідних повзунків. Також можна вибрати швидкість роботи алгоритму, налаштовуючи його під свої потреби, та змінювати масштаб. Також можна добавити нову комаху, яка буде впливати на роботу першої, створюючи зовсім інші фігури.

Користувач може виконувати алгоритм покроково, відслідковуючи кожен крок комахи.

Також можна переміщати комаху в інше місце. Це потрібно, наприклад, для втручання однієї комахи в роботу іншої.

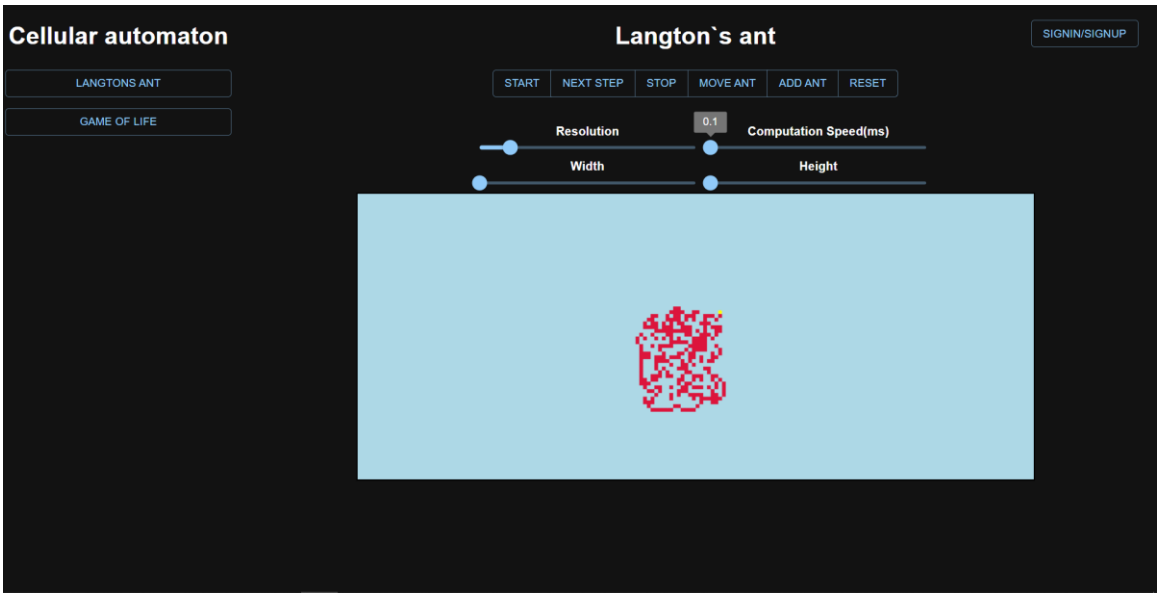

Рисунок 4.14 - Приклад роботи клітинного автомату “Langton`s ant”

4.6 Висновки до розділу

Інсталяція програмного забезпечення дозволяє користувачам швидко та ефективно встановити систему на свої пристрої. Вимоги до програмного забезпечення комп’ютера забезпечує оптимальну продуктивність та забезпечує сумісність системи з різними платформами.

Демонстрація функціоналу надає користувачам можливість ознайомитися з основними можливостями системи. Вона дозволяє візуально представити переваги та можливості програмного продукту, покращуючи користувацький досвід та сприяючи його ефективному використанню.

У цілому, взаємодія користувача з програмною системою включає в себе послідовний ряд етапів, спрямованих на максимальне полегшення використання системи та задоволення потреб користувача.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Ідея стартап-проекту полягає у створенні системи, яка візуалізує роботу алгоритмів клітинних автоматів та дає можливість аналізувати їх, записуючи результати до записника. Така система потрібна для проведення досліджень у сфері клітинних автоматів.

Широка система налаштувань початкових умов та параметрів дасть користувачу гарний інструмент для спостереження, записів та аналізу роботи клітинних автоматів у новітньому та зручному інтерфейсі.

5.1 Опис ідеї стартап проекту

Давайте розглянемо суть концепції, можливі шляхи використання ідеї, відмінність запропонованої ідеї від існуючих на даний момент аналогів та ключові моменти, які користувач системи може отримати. Результати цього аналізу наведені у показаній далі таблиці 5.1.

Таблиця 5.1. – Стартап-проект, основні переваги та ідеї

Ідейний зміст	Сфера застосування	Переваги використання
Створення системи для роботи з клітинними автоматами	Дослідження існуючих структур	Зручний інтерфейс
	Розробка нових структур	Широкий спектр налаштувань початкових умов
	Ведення записів спостереження за роботою алгоритмів	Можливість вести записники та зберігати записи результатів з коментарями

У наш час на світовому ринку існують додатки з даним функціоналом, проте після глибокого аналізу аналогічних застосунків було зроблено висновок, що не кожна з цих програм доступна у вільному доступі, більшість з них – це комерційні проекти з великою ціною за доступ до функціоналу. Також, не кожен інтерфейс проаналізованих програм був повністю зрозумілий користувачу. Тому у даній роботі буде проведена робота по вилученню цих недоліків шляхом розробки цього проекту в форматі Open-source, що робить його безкоштовним для кожного, та зі зручним інтерфейсом, який буде зрозумілий кожному.

Open-source (відкритий код) - це концепція розробки програмного забезпечення, яка передбачає, що вихідний код програмного продукту доступний для громадськості. Це означає, що будь-хто може переглядати, модифікувати та розповсюджувати цей код. Проекти з відкритим кодом стимулюють співпрацю та взаємодію між розробниками з усього світу, сприяючи інноваціям та швидкому розвитку програмного забезпечення. Також вони дозволяють користувачам вільно використовувати, розповсюджувати і модифікувати продукт в межах ліцензій, які забезпечують відкритість та вільність використання.

Результат аналізу у таблиці 5.2.

Таблиця 5.2. – Демонстрація характеристик

Характеристики техніко-економіки	Додатки та застосунки, розроблені конкурентами			Слабкі (W), нейтральні (N) та сильні (S) сторони		
Назва продукту	Розроблена система	MATLAB	Wolfram Mathematica	W	N	S

Таблиця 5.2 (продовження)

Платформа	Кроссплатформенна	Кроссплатформенна	Кроссплатформенна		+	
Ціна	безкоштовний	Standard ($5.25/місяць)	Standard ($940/рік)			+
Простота використання	Висока	Низька	Помірна			+
Open-source проект	Так	Ні	Ні			+
Постійна підтримка	Так	Так	Так	+		

Наразі система, яку ми розробили, пройшла випробування і працює у веб-браузері на будь-якій операційній системі. Вона володіє унікальними характеристиками на ринку, включаючи те, що це проект з відкритим вихідним кодом, має зручний інтерфейс та повністю безкоштовний доступ. Цільове використання системи повністю відповідає поставленим цілям проекту.

5.2 Технологічний аудит мети реалізованого додатку

Для технічного аудиту ідеї проекту необхідно провести перевірку технології, яка буде використана для втілення концепції проекту. Мета - визначити доступність цих технологій та визначити, чи потребують вони додаткового налаштування або вдосконалення. Результати аудиту подані у таблиці 5.3.

Таблиця 5.3. – Технічна повнота ідеї застосунку

Ідея застосунку	Засоби її реалізації	Наявність засобів розробки	Доступність засобів розробки
Розробка застосунку для вільної роботи з клітинними автоматами	Засіб для розробки(IDE) Visual Studio Code та встановлені npm-пакети JavaScript	+	Повністю доступні та безкоштовні продукти

Вибрано інструментарій для впровадження проекту, який включає в себе розробницьке середовище Visual Studio Code та мову програмування JavaScript, узгоджену з рядом додаткових бібліотек.

5.3 Проведення аналізу запуску розробленого стартапу

Оцінка перспектив ринку, які можна використати при впровадженні проекту, та аналіз ринкових загроз, що можуть вплинути на успіх проекту, дозволяють розробити стратегічний план розвитку, враховуючи сучасний стан ринкового оточення, потреби потенційних клієнтів та пропозиції конкуруючих проектів. Давайте проведемо детальний аналіз попиту, використовуючи інформацію, представлену у таблиці 5.4, для отримання глибшого розуміння динаміки ринку.

Таблиця 5.4. – Характеристика можливого ринку для стартапу

Параметри становища ринку	**Оцінка характеристик**
Загальна потреба в продукції	Помірна
Потенційні річні об'єми виробництва в природних одиницях виміру	До 100 копій
Річні обсяги виробництва у грошовому еквіваленті	0 – 7500 $
Якісна оцінка динаміки ринку	Динаміка є стабільною
Визначення наявності обмежень для входу	Не зафіксовано
Специфічні вимоги до стандартизації та сертифікації	Немає
Оцінка середньої норми рентабельності в галузі або на ринку	35%

Згідно з попередньою оцінкою, ринок виглядає досить привабливим для входження. Під час аналізу системи виявлено, що попит на дослідження клітинних автоматів є стабільним, але інноваційний проект може значно збільшити цей попит. Крім того, проводиться ідентифікація потенційних груп клієнтів, аналізуються їх характеристики, і формується приблизний перелік вимог до товару для кожної групи в таблиці 5.5.

Таблиця 5.5. – Модель можливих замовників стартапу

Необхідність, яка формує попит	Цільова аудиторія	Поведінка користувачів	Очікування споживачів від продукту
Проведення аналізу клітинних автоматів	Студенти	Побудова систем на основі моделей клітинних автоматів	зручність і простота використання
	Будь-який користувач, якому цікаві клітинні автомати	Проведення аналізу клітинних автоматів для власних цілей	зручність і простота використання

Після визначення потенційних груп клієнтів необхідно провести аналіз ринкового середовища, створивши таблиці, де враховані фактори, що підтримують успішне впровадження проекту на ринку, а також ті, які можуть стати перешкодою. Результати цього аналізу представлені в таблицях 5.6 та 5.7 відповідно. Отримані відомості визначаються як ключові для подальшого розвитку стратегій та тактик введення продукту на ринок.

Таблиця 5.6 – Загрози та їх фактори

Фактор	Загроза	Вірогідна реакція компанії
Поява конкурентів	Можливе конкурентне виникнення, що може призвести до створення продукту вищої якості.	Постійне вдосконалення та розширення можливостей продукту

Таблиця 5.6 (продовження)

Економічний спад	Низький попит на програмний продукт через економічні чинники	Зниження ціни; Зміна орієнтації на аудиторію іншого ринку.
Зниження репутації компанії	Можливість зростання попиту через конкурентну дію	Організація та впровадження реклам та акцій для програмного продукту

Таблиця 5.7 – Фактори можливостей

Фактор	Зміст загрози	Можлива реакція компанії
Невелика кількість конкурентів	Наразі ринок має обмежену конкуренцію	Розгортати розповсюдження розробленого продукту, розширювати його можливості та функціонал
Відповідні тенденцій ринку	На сучасному етапі IT-ринку потрібна система для роботи з клітинними автоматами	Пропагувати використання створеного продукту

Далі важливо здійснити огляд пропозицій, визначивши основні характеристики конкурентного середовища, як зазначено у таблиці 5.8.

Таблиця 5.8 – Детальний розгляд ситуації на ринку через ступеневий аналіз

Особливості середовища при конкуренції	Прояви даної характеристики	Методи вирішення проблеми
Тип конкуренції: монополія	Відсутність аналогів на ринку	Збільшення попиту на вироблену продукцію

Таблиця 5.8 (продовження)

За рівнем конкурентної боротьби: Локальна	Конкуренція на внутрішньому ринку	Відсутність конкурентів в країні, можливість встановлення власних цін
За галузевою ознакою: Внутрішньогалузева	Продукція спрямована на конкретну сферу	Можливість розширення функціоналу продукту
За інтенсивністю: Марочна	Можливі конкуренти	Аналіз ринків при виході за межі країни
Конкуренція за видами товарів: товарно-видова	Продукт має конкурентів в схожих галузях	Розширення можливостей продукту
За характером конкурентних переваг: Цінова	Цінові аспекти важливі	Можливість підвищення ціни на нові розробки

Після оцінки конкурентного середовища проводиться ретельний аналіз умов конкуренції в галузі, який представлений у таблиці 5.9, використовуючи модель "П'ять сил" Майкла Портера.

Таблиця 5.9 – Галузева конкуренція за М. Портером

Частина галузі	Конкуренти в галузі	Можливі конкуренти	Клієнти	Аналоги(замінники)
	Розробники аналогічних систем	Інноваційні технології	Кожен клієнт важливий	Відсутні.

Таблиця 5.9 (продовження)

Висновки	Напруженість конкурентності є помірною	Усі можливості для входу на ринок доступні. Потенційні конкуренти не виявлені, і терміни їхнього входу на ринок невідомі	Клієнти не мають значного впливу на умови, а продукт виступає як додатковий помічник	Немає обмежень для діяльності

На основі аналізу конкуренції за моделлю М. Портера, викладеного в таблиці 5.9, а також з урахуванням характеристик ідеї проекту з таблиці 5.2, вимог споживачів до товару з таблиці 5.5 та впливу факторів маркетингового середовища з таблиць 5.6 і 5.7, ми визначимо та обґрунтуємо перелік факторів конкурентоспроможності у таблиці 5.10. Цей аналіз буде служити основою для розробки стратегій, спрямованих на підвищення конкурентоздатності продукту на ринку.

Таблиця 5.10. – Пояснення аспектів конкурентоспроможності

Аспект	Пояснення
Мала наявність конкурентів різних компаній на ринку	Відсутність конкуренції

Таблиця 5.10 (продовження)

Системні вимоги до розробленого застосунку	Розроблена програма запускається на будь-якому приладі
Зрозумілість інтерфейсу	Інноваційний та простий для розуміння користувацький інтерфейс
Необхідність постійного підключення до інтернету	Для коректної роботи потрібно мати доступ до мережі

На підставі визначених факторів конкурентоспроможності виконується аналіз як сильних, так і слабких сторін стартапу. Він представлений у таблиці 5.11.

Таблиця 5.11 – Оцінка проекту шляхом порівняльного аналізу

Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
		–3	–2	–1	0	+1	+2	+3
Відсутність конкурентів	17				+			
Системні вимоги	8					+		
Простота використання	20	+						
Унікальний функціонал	17			+				

Завершальним етапом аналізу ринкових можливостей для запровадження проекту є формування SWOT-аналізу. Ця матриця включає в себе аналіз слабких, сильних якостей, можливостей із загрозами на основі виділених ринкових даних, а також інформаційних даних цих сторін, які визначені у таблиці 5.12.

Таблиця 5.12. – Аналіз за допомогою SWOT

Сильні сторони:	Слабкі сторони:
• Умовна безкоштовність; • новітні технології; • зручний інтерфейс • Open-source	• недостатньо коштів в обороті; • нова компанія без репутації; • поки що вузька спеціалізація;
Можливості:	Загрози:
• додаткові послуги; • вихід на нові ринки; • розширення можливостей;	• поява конкурентів; • зниження репутації компанії; • економічний спад; • зменшення попиту;

За даними проведеного SWOT-аналізу розробляємо варіант ринкової стратегії для введення стартапу на ринок, а також приблизно припустимий час їх реалізації на ринку, з урахуванням потенційних конкурентів та їхніх можливих проектів. Визначені альтернативи проаналізовані з урахуванням термінів та ймовірності забезпечення необхідними ресурсами, що представлено в таблиці 5.13.

Таблиця 5.13. – Альтернативи ринкового впровадження стартап-проекту

Аналог ринкової поведінки	Ймовірність забезпечення потрібними ресурсами	Терміни
Пошук нових ринкових сфер	Пошук спонсорів	Після введення на ринок головного продукту
Розширення функціоналу	Пошук спонсорів	Після введення на ринок головного продукту

Як висновок, після ретельного аналізу визначено, що стратегія запуску програмного продукту передбачає його початкову інтродукцію на ринок. Після успішного введення продукту в експлуатацію планується поетапне розширення його функціоналу для задоволення зростаючих потреб користувачів. Крім того, після встановлення стійкого попиту та закріплення на ринку, буде розглядатися можливість реалізації продукту на нових ринках для розширення аудиторії та забезпечення сталого зростання бізнесу.

5.4 Ринкова стратегія проекту

Процес розробки даної стратегії для початку передбачає обирання стратегії для захоплення ринку, точніше, розгортання характеристик цільових аудиторій, визначених у відповідній таблиці 5.14. Цей етап дозволяє детально вивчити потреби та уподобання цільових аудиторій, що стане основою для подальших маркетингових заходів та розробки продукту, що відповідає вимогам споживачів.

Таблиця 5.14. – Визначення цільової аудиторії поміж потенційних споживачів

Опис потенційних споживачів	Їх готовність сприйняти застосунок	Приблизний попит в цільовому сегменті	Напруженість конкуренції в цільовому сегменті	Простота входу у сегмент
Студенти	+	Потреба є	Невелика (практично відсутня)	Просто
Звичайні користувачі	+	Мала потреба	Невелика (практично відсутня)	Дуже просто

На підставі аналізу потенційних груп споживачів (сегментів), ініціатори проекту обирають цільові групи, яким планують представляти свій продукт, та розробляють стратегію охоплення ринку. Для успішної діяльності в обраних сегментах ринку визначається основна стратегія розвитку, яка детально висвітлена у таблиці 5.15.

Таблиця 5.15. – Формулювання основної стратегії для подальшого розвитку

Обраний шлях для просування проекту	Стратегія для охоплення ринку	Основні позиції, які забезпечують конкурентоспроможність у вибраній альтернативі	Фундаментальна стратегія для подальшого розвитку
Вихід на нові ринки	Стратегія диференціації	Розробка унікальних якостей продукту	Стратегія диференціації
Розширення функціоналу	Стратегія диференціації	Розширення функціоналу з додаванням товару кращих властивостей	Стратегія диференціації

Оптімізація стратегії конкурентної поведінки визначається в таблиці 5.16.

Таблиця 5.16. – Формулювання основної стратегії конкурентної поведінки

Чи є проект першим, що входить на ринок?	Ні
Чи буде компанія активно привертати нових споживачів або виводити існуючих від конкурентів?	Так
Чи планує компанія відтворювати основні характеристики товару конкурента?	Так
Тактика взаємодії з конкурентами	Стратегія виклику лідера

Виходячи з вимог споживачів з різних секторів до постачальника та цифрового додатку, враховуючи обрану цільову тактику для розвитку та тактику конкурентноспроможної поведінки в умовах ринку, необхідно пропрацювати стратегію позиціонування, яка буде відображена в таблиці 5.17. Це включає до себе формування ринкової позиції, яка дозволить споживачам розпізнати торговельне лого чи назву проекту на основі їхніх потреб і очікувань.

Таблиця 5.17. – Стратегія створення місця чи образу для продукту на ринку

Споживчі критерії та очікування цільової аудиторії від продукту	Фундаментальний план розвитку	Основні конкурентні переваги власного стартапу	Обрання асоціацій, які сприятимуть формуванню комплексної позиції власного проекту
Всім доступна ціна, простота і зручність використання	Стратегія диференціації	Простота у розумінні і використанні. Сполучення доступності та високих технічних характеристик.	Асоціації з простотою, доступністю та зручністю використання; метрики програмного забезпечення.

Отже, робота над стартап-проектом має бути систематично спланована. За обраною стратегією диференціації планується розробка та розповсюдження програмного продукту з унікальними характеристиками. При цьому, дотримуючись конкурентної стратегії "виклику лідера", фокус буде зосереджений на випуску єдиного типу товару для всіх користувачів.

5.5 Створення маркетингового плану для стартап-проекту

Початковим етапом у процесі імплементації маркетингового додатку стартапу - це розробка концепції продукту. Для досягнення цієї мети, необхідно узагальнити дані, отримані шляхом аналізу конкурентоспроможності продукту у таблиці 5.18.

Таблиця 5.18. – Визначення переваг концепції потенційного товару

Основна мотивація чи необхідність	Переваги або задоволення, які отримує клієнт від використання товару	Основні фактори, які роблять товар привабливішим у порівнянні з аналогічними продуктами на ринку
Експертна оцінка характеристик і якості продукту	Оцінка відносно стандартів	Велика точність рейтингу програмної системи

Наступним етапом є створення 3-рівневої маркетингової моделі продукту, де проводиться уточнення ідеї продукту, його фізичних характеристик та особливостей процесу надання. Це дозволить докладніше визначити якісні та кількісні характеристики товару, а також специфікації його використання та поширення на ринку. Його фізичні компоненти та особливості процесу надання в таблиці 5.19.

Таблиця 5.19. – Розгляд трьох рівнів товарної моделі

Рівні товару	Сутність та складові
Товар за задумом	Система генерації моделі клітинних автоматів

Таблиця 5.19 (продовження)

Товар у реальному виконанні	Властивості/характеристики
	Повністю реалізовано заплановану систему, додано зручний графічний інтерфейс з додатковими можливостями проведення аналізу.
Товар із підкріпленням	До продажу за підпискою: стандартна система для роботи з клітинними автоматами

Розроблена система публікується з програмною частиною, за ідеями Open-source проектів, тому кожен має доступ до коду з можливістю його модифікації, після чого буде проведена перевірка поданих модифікацій групою експертів з розробки.

Визначення цінових меж, які слід враховувати при встановленні ціни на потенційний товар, описано в таблиці 5.20 через аналіз цін на аналогічні товари.

Таблиця 5.20. – Формулювання границь при визначенні цінової політики

Ціновий рівень для аналогічних товарів	Економічний статус або рівень доходів цільової аудиторії споживачів	Нижня та верхня межі встановлення ціни
5 – 1000 $	100 – 1000 $	0 – 100 $

Далі, наступним кроком є визначення оптимальної системи збуту, в рамках якої приймаються рішення, як показано в таблиці 5.21. Визначається вибір

глибини збуту, видів посередників, і розглядається можливість проведення збуту власними силами.

Таблиця 5.21. – Створення механізму збуту

Особливості поведінки цільових клієнтів у процесі закупівель	Обов'язки, які повинен виконувати постачальник товарів у сфері збуту	Кількість проміжних ланок у системі збуту	Ефективна організація системи збуту
Бажання отримати унікальний функціонал	Продаж товарів за підпискою	Тільки виробник	На всі етапи виробництва та збуту продукції

Останньою частиною цієї ринкової програми є реалізація та імплементація ідеї комунікацій, яка базується на раніше обраному підході до позиціонування. Специфіку поведінки клієнтів наведено в таблиці 5.22.

Таблиця 5.22. – Ідея маркетингової взаємодії

Активності та взаємодія цільової аудиторії	Канали, якими споживачі сприймають та обмінюються інформацією	Основні точки, визначені для створення певного образу або місця на ринку	Мета або ціль рекламного повідомлення
Бажання отримати унікальний функціонал	Будь-які	Безкоштовність Легкий і простий у використанні продукт Унікальний функціонал	Залучити якомога більше зацікавлених клієнтів

5.6 Висновки до розділу

З урахуванням відсутності конкуренції на ринку, проект володіє потенціалом для успішної комерціалізації. Попит на розробку програмного забезпечення на основі клітинних автоматів стабільний, але інноваційний підхід до реалізації проекту може сколихнути дослідників, створюючи сприятливі умови для впровадження проекту.

Базова стратегія розвитку проекту полягає в диференціації, що визначається наданням споживачеві необхідного товару. Внаслідок ретельного аналізу аналогічних систем було розроблено кілька унікальних характеристик продукту, що підсилюють його конкурентоспроможність.

Перспективи впровадження враховують стан конкуренції, потенційні групи користувачів та конкурентоспроможність проекту. Це надає можливості впровадження, підкреслюючи важливість розробки створеного програмного продукту.

ВИСНОВКИ

Під час виконання даної роботи була сформульована задача створення веб-додатку для генерації клітинних автоматів та їх аналізу. Для досягнення цієї мети було вирішено ряд задач, таких, як аналіз предметної області, вивчення існуючих рішень та огляд основних засобів розробки.

Розроблений та створений веб-додаток, надає користувачам можливість відображати та аналізувати динамічні процеси, що описуються клітинними автоматами та використовуються для наукових досліджень в різних галузях науки та техніки. Зокрема, користувачі можуть налаштовувати параметри клітинних автоматів, спостерігати за їхньою еволюцією та зберігати результати своїх досліджень. Веб-додаток також надає можливість редагувати правила та початкові умови клітинних автоматів.

Було проведено аналіз моделі клітинного автомату побудованого на полі 7 на 7 з випадковою початковою кількістю клітин. Встановлено зв'язок між початковою кількістю клітин та кількістю ітерацій, що приводить систему до стабільного стану.

Для розробки веб-додатку були використані засоби, які включають в себе Vite та ReactJS для фронт-енду, а також Node.js та бібліотеку Express для бек-енду. Ці інструменти дозволили створити потужний та швидкий додаток зі зручним інтерфейсом.

Розроблений веб-додаток відкриває нові можливості для генерації та аналізу клітинних автоматів, що є важливим інструментом для багатьох наукових досліджень та застосувань. Він дозволяє користувачам легко створювати та вивчати клітинні автомати та сприяє подальшому розвитку цієї галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мейєр Е. CSS – каскадні таблиці стилів. Детальний посібник. 2008. 208 с.
2. Ніксон Р. Створюємо динамічні веб-сайти. 2019. 816 с.
3. Banks A. Learning react: functional web development with react and redux. 2017. 348 с.
4. Brown E. Web development with node and express: leveraging the javascript stack. 2019. 343 с.
5. Casciaro M. Node.js Design Patterns - Second Edition: master best practices to build modular and scalable server-side web applications. 2016. 777 с.
6. Drake J., Worsley J. Practical PostgreSQL. 2002. 636 с.
7. Ferrari L., Pirozzi E. Learn PostgreSQL: Build and manage high-performance database solutions using PostgreSQL 12 and 13. 2020. 652 с.
8. Griffeath D., Moore C. New constructions in cellular automata. 2003. 360 с.
9. Hadeler K.-P., Müller J. Cellular automata: analysis and applications. 2017. 478 с.
10. Hahn E. Express in Action: Writing, building, and testing Node.js applications. 2016. 256 с.
11. Ilachinski A. Cellular automata: a discrete universe. 2001. 840 с.
12. L. Schiff J. Cellular automata: a discrete view of the world. 2008. 288 с.
13. Silva R. Essential postgres: database development using postgresql. 2020. 212 с.
14. Stone S. M. Automating with node.js. 2018. 196 с.
15. Toffoli T. Cellular automata machines: a new environment for modeling. 1987. 200 с.

16. Wilson J. Node.js 8 the right way: practical, server-side javascript that scales. 2018. 336 с.
17. Wolfram S. Cellular automata and complexity: collected papers. 1994. 608 с.
18. Hunter II T. Distributed Systems with Node.js: Building Enterprise-Ready Backend Services. 2020. 377 с.
19. Frain B. Responsive. Web Design with HTML5 and CSS: Develop futureproof responsive websites using the latest HTML5 and CSS techniques. 2020. 408 с.
20. Regina O. Obe. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database. 2017. 314 с
21. ReactJS [Електронний ресурс] – Режим доступу: https://react.dev/learn
22. Node.js [Електронний ресурс] – Режим доступу: https://nodejs.org/en/docs
23. Effector.js [Електронний ресурс] – Режим доступу: https://effector.dev/
24. Sequelize [Електронний ресурс] – Режим доступу: https://sequelize.org/
25. ESLint [Електронний ресурс] – Режим доступу: https://eslint.org/docs/latest/
26. MaterialUI [Електронний ресурс] – Режим доступу: https://mui.com/material-ui/getting-started/
27. Vite.js [Електронний ресурс] – Режим доступу: https://vitejs.dev/guide/
28. Express.js [Електронний ресурс] – Режим доступу: https://expressjs.com/
29. JSON Web Token (JWT) [Електронний ресурс] / [M. Jones, Microsoft, J. Bradley та ін.]. – 2015. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/rfc7519
30. Hypertext Transfer Protocol — HTTP/1.1 [Електронний ресурс] / [R. Fielding, U. Irvine, J. Gettys та ін.]. – 1999. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/rfc2616