Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Клименко Б.С.
«____» _____ 20 р.

Керівник проекту
доктор технічних наук, професор
_____ Єлизавета МЕЛЕШКО
«____» _____ 20 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань *12 "Інформаційні технології"*

Спеціальність *122 "Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерні науки"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
« __ » _____ 20__ року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Клименку Богдану Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж*

2. Керівник роботи *Мелешко Єлизавета Владиславівна, доктор техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу №__-__ від __.__.20__ року

3. Строк подання роботи до захисту __.__.*2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Діаграма процесів	*1 аркуш*
Показники економічної ефективності	*1 аркуш*

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В., к.т.н., доцент	25.10.2023	10.11.2023
Охорона праці	Оришака О.В., к.т.н., доцент	03.11.2023	21.11.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	16.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
«__»_____20 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____20 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Клименко Б.С. Дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для реалізації системи моделювання та аналізу віртуальних соціальних мереж.

Метою розробки є дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж.

Об'єктом дослідження є процес автоматизованого дослідження віртуальних соціальних мереж.

Предметом дослідження є методи моделювання та аналізу віртуальних соціальних мереж.

Методи дослідження базуються на теорії графів, теорії складних мереж, теорії алгоритмів та структур даних, теорії статистики, теорії розробки програмного забезпечення та теорії імітаційного моделювання.

Результат роботи – програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено на мові програмування високого рівня Python.

Ключові слова: комп'ютерна інженерія, імітаційне моделювання, віртуальні соціальні мережі, складні мережі, аналіз даних.

ABSTRACT

Klymenko B.S. Research and software implementation of a system for modelling and analysing virtual social networks. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this master's thesis, software designed for a system for modelling and analysing virtual social networks.

The purpose of the development is the research and program implementation of a system for modelling and analysing virtual social networks.

The object of the research is the process of automated research of virtual social networks.

The subject of the research is the methods of modeling and analysis of virtual social networks.

Research methods are based on the theory of graphs, the theory of complex networks, the theory of algorithms and data structures, the theory of statistics, the theory of software development, and the theory of simulation modeling.

The result of the work is the software implementation of a system for modelling and analysing virtual social networks.

In the process of working on a software model an analysis of existing hardware and software was performed. All components of the software developed are fully described.

User-friendly user interface is developed. These are instructions for working with software.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the high-level programming language – Python.

Keywords: computer engineering, simulation modeling, virtual social networks, complex networks, data analysis.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

SIR – це модель поширення вірусної інформації у соціальній мережі, яка включає в себе стани "Susceptible" (Сприйнятливий), "Infected" (Інфікований) і "Recovered" (Відновлений). Вона описує, як користувачі можуть бути вразливими до інформаційного впливу, як вони інфікуються та відновлюються від впливу.

SIRS – це модель поширення вірусної інформації у соціальній мережі, яка подібна до моделі SIR, але включає додатковий стан "Susceptible" після стану "Recovered". Ця модель описує інфікування, відновлення та вразливість користувачів до впливу з часом.

Вершина – це один з об'єктів або вузлів у графі, який може бути з'єднаний з іншими вершинами ребрами. Вершини представляють собою об'єкти, які аналізуються у контексті графу або мережі. У даній роботі під вершиною графу розуміється абстрактний користувач соціальної мережі.

Вірусна інформація – це інформація або вплив, який поширюється через соціальну мережу, аналогічно поширенню вірусів. Це може бути будь-яка інформація, яка передається від одного користувача до іншого, така як новина, реклама, чутка тощо.

Граф – це математична структура, яка представляється як набір вершин та ребер, що з'єднують ці вершини. Графи використовуються для моделювання та аналізу взаємодії об'єктів у різних сферах, включаючи соціальні мережі.

Імітаційне моделювання – це метод симуляції реальних процесів або систем шляхом створення віртуальних моделей, які відтворюють ці процеси або системи в комп'ютерному середовищі.

Ребро – це з'єднання або зв'язок між двома вершинами у графі. Ребра визначають, які вершини взаємодіють між собою.

Складна мережа – це мережа, яка має складну та нетривіальну структуру, включаючи велику кількість вершин і зв'язків, що взаємодіють між собою, а також кластерні утворення, високий коефіцієнт кластеризації та малий діаметр мережі.

Співтовариство – це підмножина вершин у графі, в якій вершини мають більш інтенсивні зв'язки або взаємодії одна з одною, ніж з вершинами, які не належать до цього співтовариства.

Центральність за близькістю – це міра впливовості вершини у соціальній мережі, яка визначається на основі її відстані до всіх інших вершин у графі.

Центральність за власним вектором – це міра впливовості вершини у соціальній мережі, яка базується на математичному понятті власного вектора графу.

Центральність за степенем – це міра впливовості вершини у соціальній мережі на основі її ступеня, тобто кількості зв'язків з іншими вершинами.

ВСТУП

Актуальність теми. Актуальність дослідження зумовлена зростанням ролі віртуальних соціальних мереж у сучасному світі, де вони стали не тільки головним засобом спілкування, але й важливим джерелом інформації та одним з потужних засобів впливу на суспільство. Збільшення кількості користувачів соціальних мереж, їхньої активності та взаємодії створює потребу в аналізі та розумінні цих великих та складних віртуальних спільнот.

Сучасне використання віртуальних соціальних мереж передбачає використання їх як платформи для реклами, маркетингу, впливу на громадську думку, а також для аналізу та передбачення поведінки користувачів. Вивчення соціальних мереж має важливе значення для розуміння механізмів розповсюдження інформації, вірусної реклами, фейків тощо.

Системи моделювання та аналізу віртуальних соціальних мереж стають важливими інструментами для дослідження та розв'язання різноманітних завдань, включаючи аналіз соціальних взаємодій, створення рекомендацій, передбачення подій, виявлення аномалій та інформаційно-психологічних атак тощо. Такі системи дозволяє дослідникам, бізнесу та урядовим установам отримувати цінну інформацію та вдосконалювати прийняття рішень на основі даних з віртуальних соціальних мереж.

До того ж, розвиток технологій та зростання обсягів даних у соціальних мережах ставлять перед вченими та розробниками нові виклики у сфері обробки, аналізу та візуалізації даних. Тому розробка програмної системи, яка б дозволяла моделювати та аналізувати віртуальні соціальні мережі, є актуальною та важливою задачею в сучасному інформаційному суспільстві.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Дослідження існуючих систем для моделювання та аналізу віртуальних соціальних мереж.

– Розробка методів та алгоритмів моделювання та аналізу віртуальних соціальних мереж.

– Програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж.

Об'єктом дослідження є процес автоматизованого дослідження віртуальних соціальних мереж.

Предметом дослідження є методи моделювання та аналізу віртуальних соціальних мереж.

Методи дослідження базуються на теорії графів, теорії складних мереж, теорії алгоритмів та структур даних, теорії статистики, теорії розробки програмного забезпечення та теорії імітаційного моделювання.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

1. Запропоновано метод імітаційного моделювання структури віртуальних соціальних мереж на основі моделі Воттса-Строгаца, який відрізняється від відомих симуляцією характеристик користувачів та перепідключенням ребер графу з урахуванням схожості вузлів мережі.

2. Розроблено вітчизняний продукт імітаційного моделювання та аналізу віртуальних соціальних мереж, який має більш широкі можливості, на відміну від існуючих аналогів та може бути використаний при дослідженні та прогнозуванні поведінки користувачів веб-ресурсів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моделювання та аналізу віртуальних соціальних мереж.

Достовірність наукових результатів підтверджена теоретичними викладками, результатами комп'ютерного імітаційного моделювання, а також результатами тестування розробленого програмного забезпечення.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній магістерській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Системи моделювання та аналізу віртуальних соціальних мереж мають на меті дослідження та аналіз віртуальних соціальних мереж, які є важливою складовою сучасного інтернету та соціальних медіа. Ці системи надають засоби для аналізу, моделювання, вимірювання та дослідження різних аспектів віртуальних соціальних мереж. Ось деякі з основних призначень систем моделювання та аналізу соціальних мереж:

– *Дослідження соціальної динаміки.* Вони дозволяють вивчати взаємодії та зв'язки між користувачами у віртуальних соціальних мережах, включаючи способи комунікації, вплив та поширення інформації.

– *Аналіз мережевої топології.* Такі системи дозволяють аналізувати мережеву структуру, таку як графи дружби, графи співпраці, графи поширення вірусів тощо. Це допомагає розуміти структуру та динаміку спільнот у соціальних мережах.

– *Моделювання та прогнозування поведінки користувачів.* Ці системи дозволяють створювати моделі для розуміння та прогнозування поведінки користувачів у віртуальних соціальних мережах. Що корисно для маркетингу, реклами, розробки продуктів і послуг, які взаємодіють з користувачами.

– *Вимірювання та аналіз впливу.* Для вивчення впливу осіб, груп чи подій у віртуальних соціальних мережах. Дослідження впливу може стосуватися вирішальних тем, таких як поширення фейкових новин, публічні дискусії, політичні кампанії тощо.

– *Аналіз даних та візуалізація.* Системи надають інструменти для обробки, аналізу та візуалізації даних у віртуальних соціальних мережах. Це допомагає виявляти закономірності, тенденції та розуміти глобальну картину мережі.

– *Розробка та тестування нових алгоритмів та сервісів.* Системи моделювання дозволяють розробникам випробувати та вдосконалювати нові алгоритми для соціальних мереж, а також створювати та тестувати нові соціальні сервіси та платформи.

– *Безпека та кібербезпека.* Аналіз віртуальних соціальних мереж також важливий для виявлення можливих загроз та вразливостей, пов'язаних із кібербезпекою та захистом персональних даних.

Системи моделювання та аналізу віртуальних соціальних мереж використовуються дослідниками, аналітиками, розробниками, управлінцями та соціологами для більшого розуміння різних аспектів соціальної взаємодії.

1.2 Область застосування

Область застосування систем моделювання та аналізу віртуальних соціальних мереж розширюється на багато галузей та цікавить різних користувачів з різними потребами та цілями. Ось декілька з головних областей застосування:

1. Академічні дослідження. Можливі користувачі у цій галузі:

– Дослідники в сфері соціальних наук для аналізу соціальних мереж, спілкування та взаємодії користувачів в Інтернеті.

– Дослідники в галузі кібербезпеки для виявлення кіберзагроз та аналізу кіберподій в соціальних мережах.

2. Освіта. Можливі користувачі у цій галузі:

– Викладачі для використання у лекціях та лабораторних роботах як наочного матеріалу.

– Студенти для вивчення моделювання та мережевого аналізу.

3. Маркетинг і бізнес. Можливі користувачі у цій галузі:

– Маркетингові аналітики для аналізу поведінки покупців та впливу соціальних мереж на споживачів для розробки маркетингових стратегій.

– Підприємці та стартапи для прогнозування популярності продуктів та

послуг та підбору стратегій залучення клієнтів і підвищення своєї видимості.

4. Комунікації і громадські відносини. Можливі користувачі у цій галузі:

– Професіонали у галузі PR для вивчення реакції громадськості та вплив подій через аналіз соціальних мереж.

5. Кібербезпека і правоохоронні органи. Можливі користувачі у цій галузі:

– Аналітики кібербезпеки для виявлення аномальної діяльності та кіберзагроз на основі аналізу соціальних мереж та взаємодії користувачів.

Системи моделювання та аналізу віртуальних соціальних мереж мають широкий спектр застосувань і користувачів, і вони є важливими інструментами для розвитку, дослідження та оптимізації віртуального світу інтернету та соціальних медіа.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми магістерської роботи

Аналіз соціальних мереж – це міждисциплінарна практика, яка знаходиться на межі різних наук, зокрема науки про суспільство, науки про мережевий аналіз, теорії графів, штучного інтелекту, інформаційного пошуку, маркетингу. Мережевий аналіз полягає у вирішенні проблем мережі, структуру якої можна пояснити за допомогою теорії графів. Теорія графів складає собою набір абстрактних понять і методів для аналізу графів. Поєднання усіх цих галузей знань із аналітичними методами та іншими методами, що розроблені для аналізу та відображення аналізу соціальних мереж, являють основу методів соціального аналізу в цілому.

Вивчення суспільства з точки зору вивчення індивідуума, що перебуває у мережевих зв'язках дозволяє знайти логічні пояснення поведінки осіб, виявити закономірності, що не можна зробити при дослідженні лише однієї людини. Завдяки розвитку комп'ютерно-обчислювальної техніки та сучасних технологій, для аналізу суспільства та суспільних явищ відкрилися нові перспективи та можливості. Ряд питань, які не можна було вирішити раніше, стали доступними для вивчення.

Мережевий аналіз знайшов застосування за межами науки про суспільство. Методи мережевого аналізу дозволили вивчати Інтернет-трафік, розповсюдження інформації. Наприклад, у бізнесі аналіз мереж використовують для поліпшення інформаційного потоку, правоохоронні органи для виявлення терористичних мереж, соціальні мережі – для рекомендації потенційних друзів тощо.

Розглянемо детальніше аналіз соціальних мереж таких, як наприклад, Facebook, Twitter, LinkedIn, Google+, GitHub тощо.

При аналізі даних соціальних мереж виникає ряд проблем:

1. Приватність персональних даних.

2. Обмеження доступу та кількості запитів.

3. Необхідність в структуризації даних.

Щодо напрямків аналізу соціальних мереж, то можна виділити наступні:

1. Аналіз текстової інформації.

2. Ідентифікація користувача у різних мережах.

3. Рекомендації контенту, товарів, послуг.

4. Прогнозування поведінки користувача.

5. Пошук спільнот, списків друзів користувача.

6. Визначення впливовості користувачів.

6. Встановлення прихованих атрибутів.

7. Визначення несправжніх профілів та таких, що несуть загрозу суспільству.

Соціальні мережі класифікують на два типи: соціоцентричні (повні) та егоцентричні (особисті).

Соціоцентричні являють собою зв'язки всіх елементів закритої популяції, сфокусовані на розгляді великих груп людей, а *егоцентричні* – зв'язки окремих особистостей.

Розглянемо основні метрики аналізу соціальних мереж.

Щільність – одна із метрик, що являє собою відношення числа наявних ребр графа до максимально можливої кількості ребр даного графа. Даний вимір часто використовується для порівняння графів одного розміру.

Коефіцієнт кластеризації – деяка оцінка фрагментованості мережі. Наприклад, в мережі друзів це ймовірність того, що двоє моїх друзів є друзями між собою.

Наближеність – це вимір, що показує швидкість передачі інформації від одного вузла до інших, пов'язаних з ним, вузлів.

Міст – вузол, що з'єднує окремі частини мережі.

Центральність – різниця між кількістю посилань для кожного вузла, поділена на максимально можливу суму різниць.

Сьогодні застосування аналізу та моделювання соціальних мереж набуло неабиякого поширення.

Моделювання соціальних мереж зосереджується на двох основних напрямках: *аналіз структури соціальних мереж* та *дослідження поширення інформації у соціальних мережах*.

Моделювання дозволяє дослідити, як формуються та розвиваються соціальні мережі, а також допомагає зрозуміти і спрогнозувати поширення інформації, вибір поведінки людьми, поведінку ринку тощо.

Важливе значення моделювання соціальних мереж також має у комп'ютерних рольових іграх, у яких передбачена можливість спілкування із неігровими персонажами. У цьому випадку моделювання спрямоване на відтворення персонажами, керованими комп'ютером, звичної соціальної поведінки.

Останні роки теорія моделювання соціальних мереж успішно розвивається. При цьому можна виділити різноманітність підходів вибору математичних моделей. Перерахуємо деякі з них:

– моделі випадкових графів;

– кооперативні теоретико-ігрові моделі;

– некооперативні теоретико-ігрові моделі.

Часто соціальні мережі представляють у вигляді моделей випадкового графа. Такі випадкові графи повинні володіти деякими властивостями, що вимірюються наступними величинами:

– Відстань між двома вершинами;

– Діаметр графа;

– Ступінь вершин (кількість контактів);

– Розподіл ступенів вершин;

– Центральності вузла;

– Коефіцієнт кластеризації;

– Коефіцієнт асортативності.

Своєрідною відмінністю соціальних мереж від традиційних графів є наявність вершин з «важким хвостом» – вершин із досить великим ступенем. При цьому видалення навіть 10% таких вершин найчастіше призводить до розпаду мережі на дрібні компоненти.

Розглянемо існуюче програмне забезпечення для аналізу віртуальних соціальних мереж.

Centrifuge – пропонує аналітикам і дослідникам інтегрований набір можливостей, які можуть допомогти їм швидко зрозуміти та отримати інформацію з нових джерел даних, візуалізувати графи соціальних мереж та інші дані, щоб робити висновки.

Рисунок 2.1 – Приклад роботи ПЗ Centrifuge

Cuttlefish – це програмне забезпечення, яке візуалізує мережі за допомогою деяких із найвідоміших алгоритмів макета. Це дозволяє детально візуалізувати мережеві дані, інтерактивно маніпулювати макетом, редагувати графіки та візуалізувати процеси, а також різні методи введення та виведення в TeX за допомогою Tikz і PSTricks. Його можна завантажити як архів jar і використовувати з популярним ПЗ Gephi.

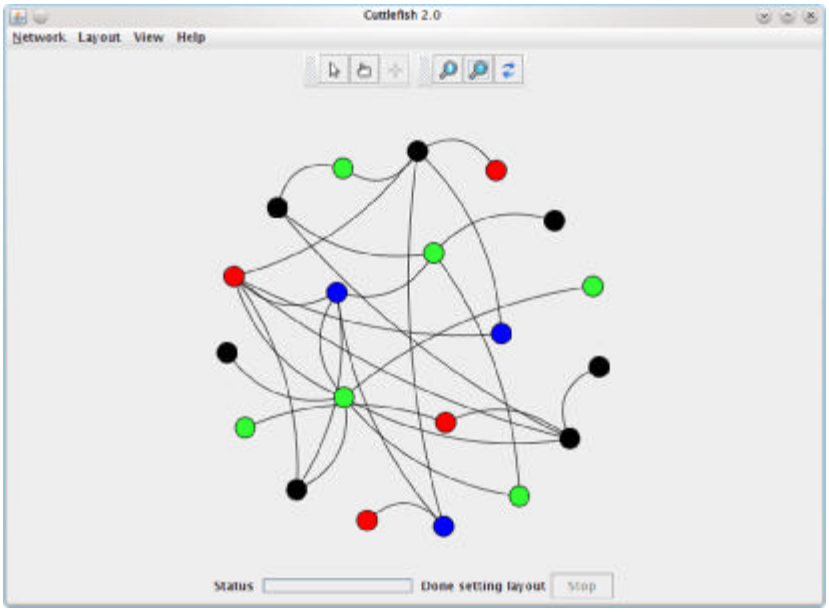


Рисунок 2.2 – Приклад роботи ПЗ Cuttlefish

Cytoscape – це програмна платформа з відкритим вихідним кодом для візуалізації мереж молекулярної взаємодії та біологічних шляхів та інтеграції цих мереж з анотаціями, профілями експресії генів та іншими даними про стан.

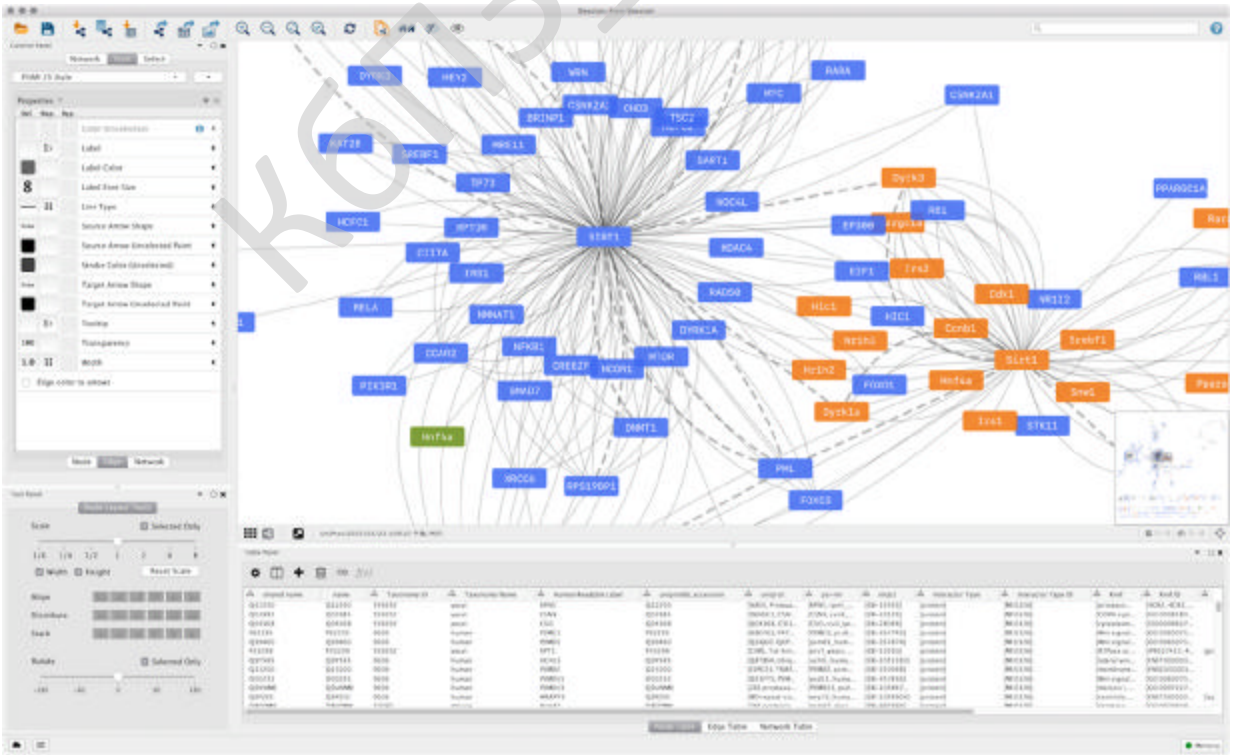


Рисунок 2.3 – Приклад роботи ПЗ Cytoscape

Хоча Cytoscape спочатку був розроблений для біологічних досліджень, тепер це загальна платформа для аналізу складних мереж, зокрема, і соціальних, та їх візуалізації. Базова збірка Cytoscape забезпечує набір основних функцій для інтеграції, аналізу та візуалізації даних. Додаткові функції доступні як плагіни, або додаткові програми. Доступні програми для аналізу мережевого та молекулярного профілювання, нових макетів, підтримки додаткових форматів файлів, створення сценаріїв і підключення до баз даних.

Gephi – це інтерактивна платформа візуалізації та дослідження для всіх видів мереж і складних систем, динамічних та ієрархічних графів. Працює в Windows, Linux і Mac OS X. Gephi є відкритим і безкоштовним. Gephi – це інструмент для людей, яким потрібно досліджувати та розуміти мережі. Для аналізу даних користувач взаємодіє з представленням, маніпулює структурами, формами та кольорами, щоб виявити приховані властивості.

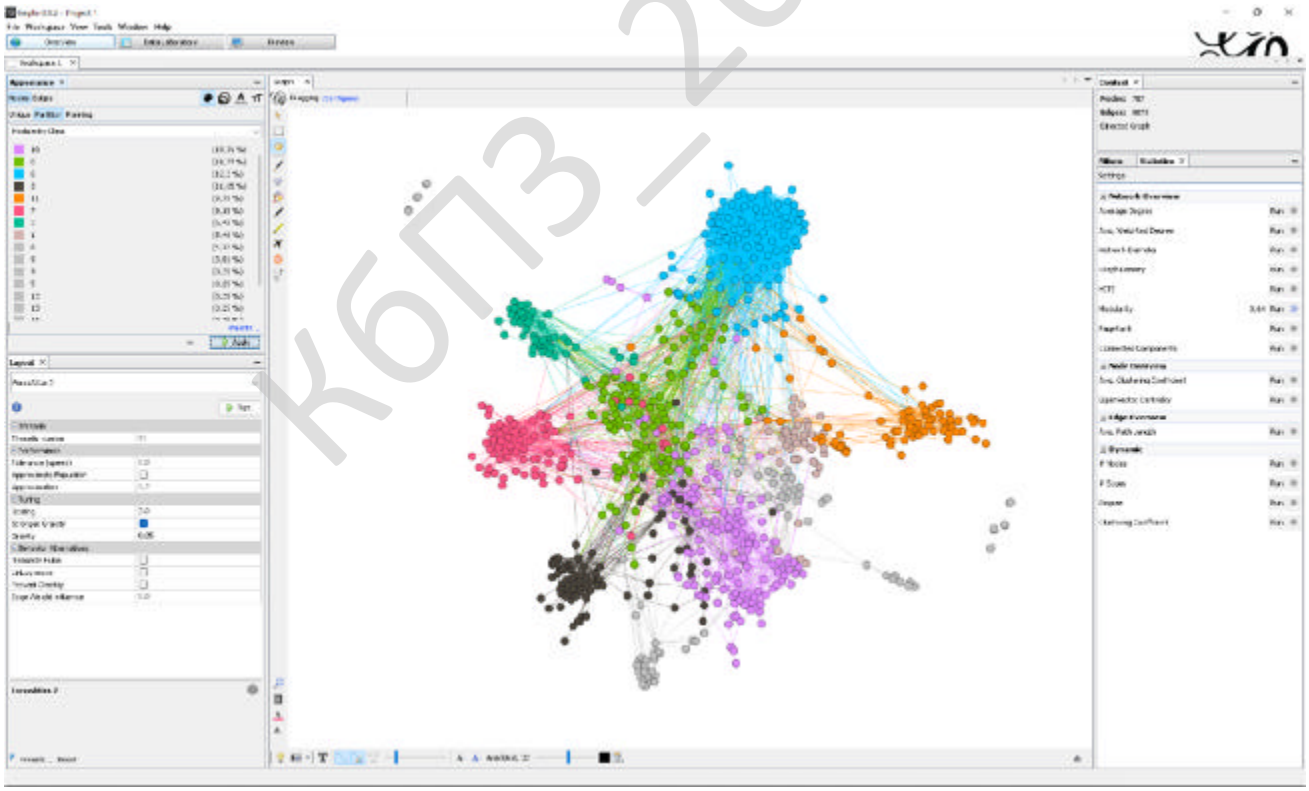


Рисунок 2.4 – Приклад роботи ПЗ Gephi

Social Network Visualizer (SocNetV) – це кросплатформна, зручна безкоштовна програма для аналізу та візуалізації соціальних мереж. За допомогою SocNetV можна:

– Створювати та редагувати структуру соціальної мережі кількома клацаннями на віртуальному полотні, завантажувати дані поля з файлу в підтримуваному форматі (GraphML, GraphViz, Adjacency, EdgeList, GML, Pajek, UCINET тощо) або сканувати Інтернет, щоб створити соціальну мережу підключених веб-сторінок.

– Редагувати акторів і зв'язки, аналізувати властивості графів і соціальних мереж, створювати чудові HTML-звіти та макети візуалізації мережі.

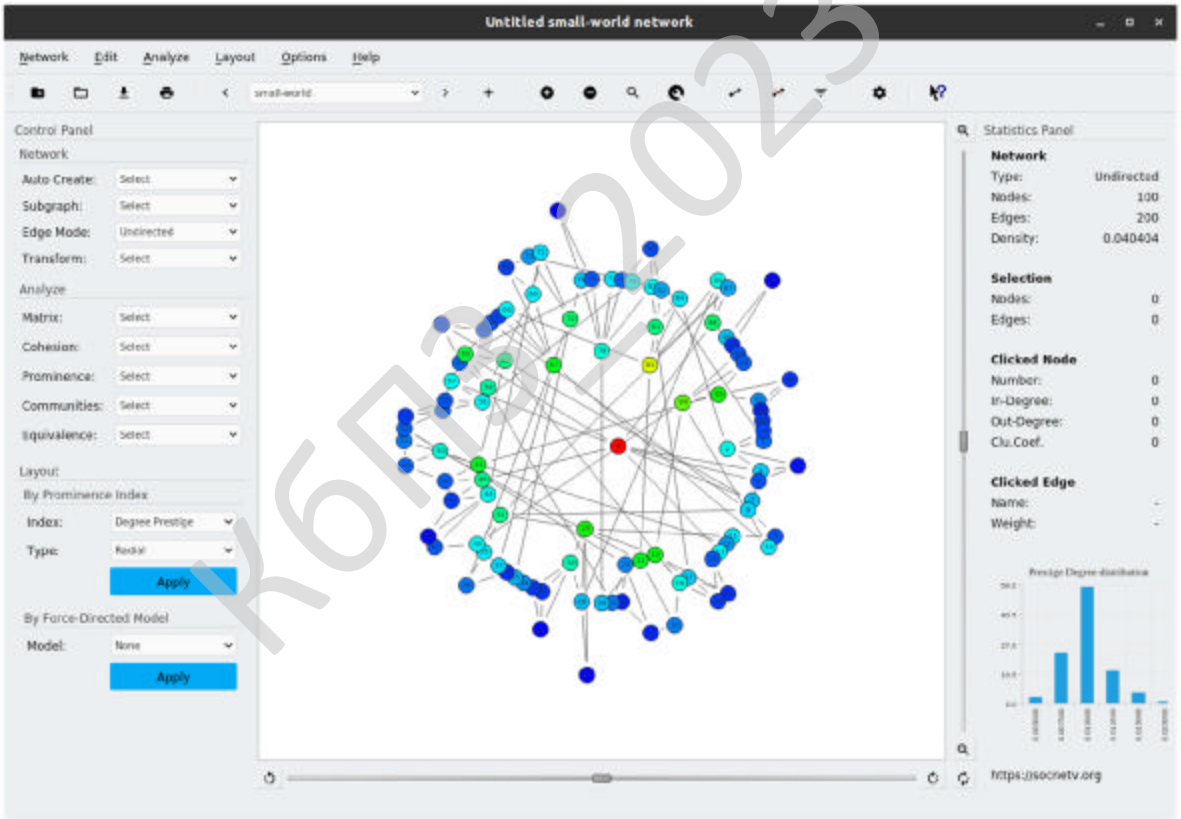


Рисунок 2.5 – Приклад роботи ПЗ Social Network Visualizer (SocNetV)

NetworkX – це пакет Python для створення, керування та вивчення структури, динаміки та функцій складних мереж.

NetworkX забезпечує:

– інструменти для дослідження структури та динаміки соціальних, біологічних та інфраструктурних мереж;

– стандартний інтерфейс програмування та реалізації графів, який підходить для багатьох програм;

– середовище швидкої розробки для спільних мультидисциплінарних проектів;

– інтерфейс до існуючих чисельних алгоритмів і коду, написаного мовами C, C++ і FORTRAN;

– можливість безболісно працювати з великими нестандартними наборами даних.

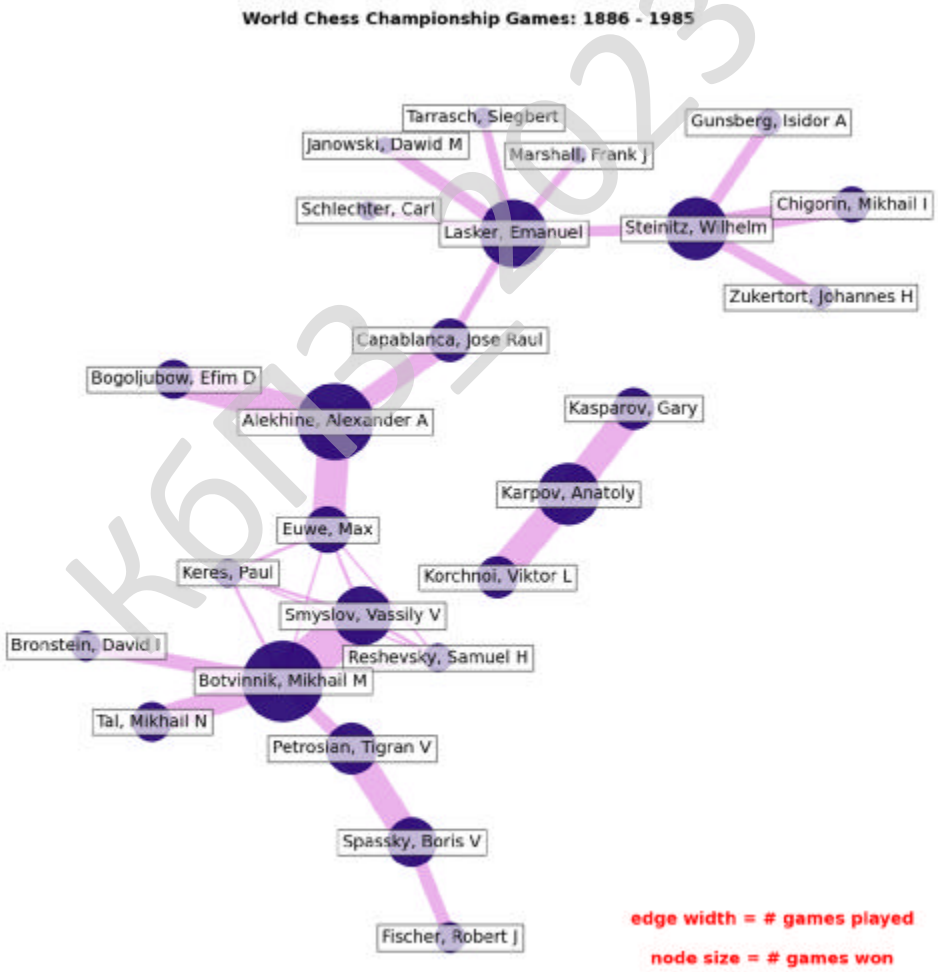


Рисунок 2.6 – Приклад №1 можливостей застосування бібліотеки NetworkX

На рис. 2.6 наведено результати читання колекції шахових матчів, що зберігаються у файлі PGN і містить усі 685 матчів чемпіонату світу з шахів 1886–1985 років. Кожен вузол – це прізвище шахового майстра. Кожен край спрямований від білого до чорного та містить вибрану інформацію про гру.

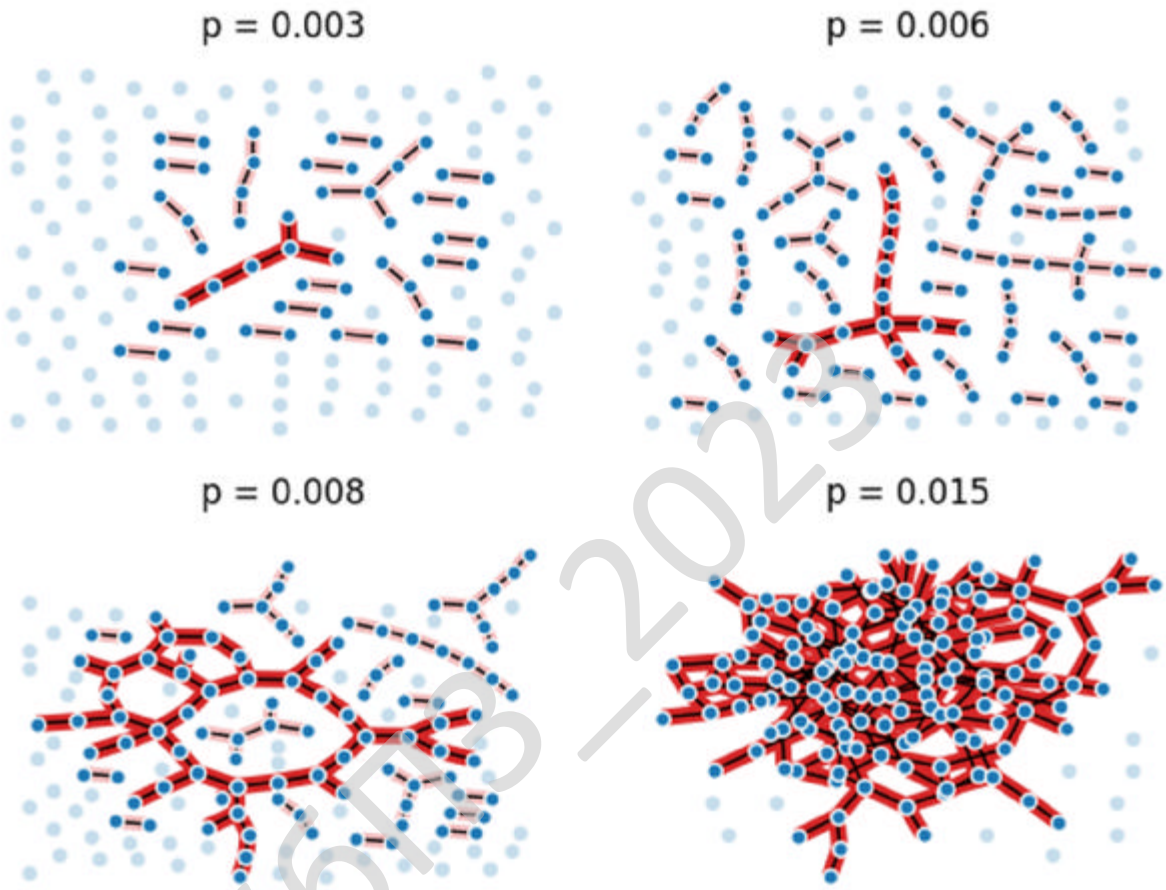


Рисунок 2.7 – Приклад №2 можливостей застосування бібліотеки NetworkX

На рис. 2.7 наведено результати пошуку гігантської зв’язної компоненти в біноміальному випадковому графі.

За допомогою NetworkX можна завантажувати та зберігати мережі в стандартних і нестандартних форматах даних, генерувати багато типів випадкових і класичних мереж, аналізувати мережеву структуру, будувати мережеві моделі, проектувати нові мережеві алгоритми, малювати мережі та багато іншого.

Усі досліджені приклади зосереджені більше на аналізі, ніж на моделюванні соціальних мереж, і переважно потребують збору вхідних даних для аналізу.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки системи моделювання та аналізу віртуальних соціальних мереж було обрано мову програмування Python та графову базу даних Neo4j.

Вибір мови програмування та бази даних є важливим етапом у процесі розробки. Нижче наведено обґрунтування вибору мови програмування та бази даних цієї системи.

Переваги вибору мови програмування Python:

– Зручність та продуктивність. Python є однією з найпопулярніших та найзручніших мов програмування в світі. Вона відома своєю простотою та легкістю вивчення, що робить її вдалим вибором для вирішення широкого кола задач.

– Багата екосистема бібліотек. Python має велику кількість бібліотек та модулів, які допомагають в розробці систем аналізу даних, обробки тексту, візуалізації тощо. Наприклад, бібліотеки NetworkX та Matplotlib можуть бути використані для аналізу та візуалізації віртуальних соціальних мереж.

– Підтримка графових баз даних. Python має декілька бібліотек, які дозволяють легко взаємодіяти з графовими базами даних та бібліотекою Neo4j, що досить важливо та зручно для аналізу віртуальних соціальних мереж.

– Ком'юніті та документація. Python має активне та дружнє ком'юніті розробників, що полегшує вирішення проблем та отримання допомоги. Додатково, є велика кількість документації та навчальних ресурсів для вивчення цієї мови та її бібліотек.

Переваги вибору бази даних Neo4j:

– Графова структура даних. Neo4j – це база даних з графовою структурою, що ідеально підходить для зберігання та операцій з даними віртуальних соціальних

мереж. Вона дозволяє ефективно моделювати та аналізувати зв'язки між користувачами, спільнотами, подіями тощо.

– **Можливості аналізу графів.** Neo4j надає потужні інструменти для аналізу графів, такі як виявлення ключових вузлів та шляхів, знаходження подібних графових структур тощо. Це важливо для вивчення відносин та впливу у віртуальних соціальних мережах.

– **Швидкодія.** Neo4j оптимізована для операцій з графами, що робить її ефективною для роботи з великими обсягами даних у віртуальних соціальних мережах.

– **Масштабованість.** Neo4j має можливості масштабування, що дозволяє розширювати базу даних для відповіді на зростаючі потреби в обсягу та продуктивності.

Вибір мови Python та БД Neo4j сприяє розробці ефективної та продуктивної системи моделювання та аналізу віртуальних соціальних мереж, забезпечуючи зручність програмістів та можливість ефективної роботи з графовими даними.

2.3 Розгорнута постановка завдання

Завдання дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж передбачає виконання наступних підзадач:

1. Огляд літератури та аналіз існуючих систем:

– Провести огляд наукових досліджень та існуючих програмних рішень з області моделювання та аналізу віртуальних соціальних мереж.

– Визначити основні вимоги до системи моделювання та аналізу.

2. Розробка концепції системи:

– Визначити цілі та завдання системи моделювання та аналізу віртуальних соціальних мереж.

– Розробити концепцію архітектури системи та обрати технології розробки.

3. Розробка моделі віртуальних соціальних мереж:

– Розробити моделі, які відображатимуть структуру та взаємодію користувачів у віртуальній соціальній мережі.

– Врахувати фактори, які впливають на поведінку користувачів.

4. Розробка алгоритмів аналізу та симуляції:

– Розробити алгоритми для симуляції подій та взаємодії користувачів у соціальних мережах.

– Розробити інструменти для аналізу структури мережі, поширення інформації, впливу користувачів тощо.

5. Розробка графічного інтерфейсу:

– Розробити графічний інтерфейс для взаємодії з системою моделювання та аналізу віртуальних соціальних мереж.

– Забезпечити зручну візуалізацію результатів аналізу.

6. Тестування та валідація:

– Провести тестування системи на різних видах віртуальних соціальних мереж та в різних сценаріях.

– Валідувати результати симуляцій та аналізу за допомогою наявних даних або імітаційних досліджень.

6. Документування та звітність:

– Створити документацію до системи, включаючи опис архітектури, алгоритмів, інструкцію користувача.

– Підготувати звіт з результатами дослідження та програмної реалізації.

Завдання з моделювання та аналізу віртуальних соціальних мереж вимагає інтердисциплінарного підходу, об'єднання знань з інформатики, соціології та математики. Реалізація такої системи може бути важливою для вивчення та розуміння поведінки користувачів у віртуальних середовищах, а також для аналізу впливу різних чинників на соціальні мережі.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Аналіз соціальних мереж, також відомий як наука про мережі, є загальним дослідженням соціальних мереж із використанням концепцій теорії графів та теорії складних мереж. Він досліджує поведінку індивідів на мікрорівні, їхні стосунки (соціальну структуру) на макрорівні та зв’язок між ними.

Аналіз соціальних мереж використовує декілька методів та інструментів для вивчення взаємозв'язків, взаємодії та зв'язку в мережі. Це дослідження є ключовим для процедур та ініціатив, пов’язаних із вирішенням проблем, адмініструванням та діяльністю цієї мережі.

Основними сутностями, необхідними для побудови мережі, є **вузли** та **ребра,** що з’єднують вузли. Аналіз соціальних мереж є широко використовуваним підходом для аналізу міжособистісних зв’язків в Інтернеті через бум соціальних мереж. Але ця концепція не обмежується соціальними мережами онлайн; його можна використовувати для будь-якого явища, який можна моделювати як мережу.

Як було встановлено раніше, **вузли** та **ребра** є будівельними блоками для аналізу соціальних мереж.

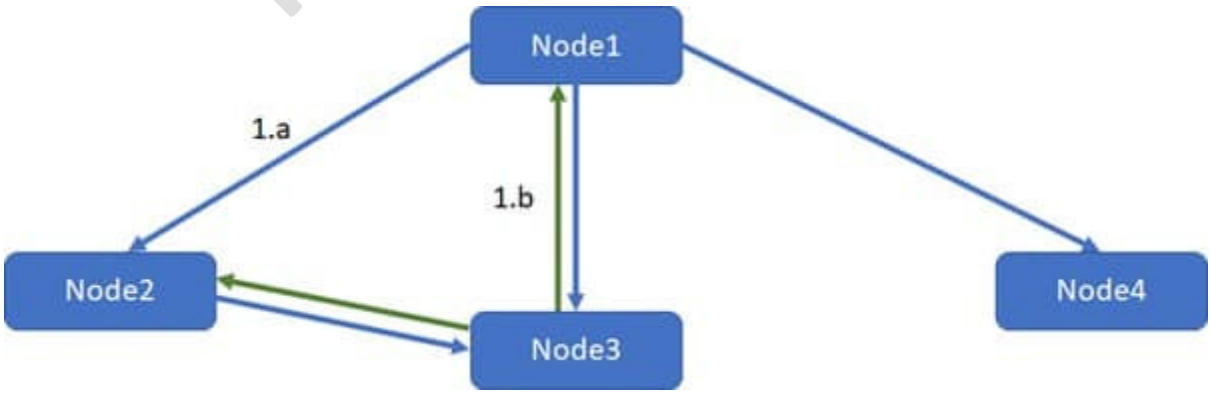


Рисунок 3.1 – Граф соціальної мережі

На рис. 3.1 наведено 1.a – орієнтоване ребро – вузли, з’єднані цим ребром, упорядковані, тобто з’єднання між вузлами одностороннє. Наприклад, Twitter, Instagram – це переважно орієнтовані периферійні мережі. Користувач може стежити за кимось без того, щоб той інший стежив за ним. 1.b Неорієнтоване ребро – зв’язок між вузлами, з’єднаними цим краєм, є взаємним, тобто зв’язок застосовний в обох напрямках. Наприклад, подружившись із особою на Facebook, LinkedIn тощо, користувач автоматично створює двостороннє з’єднання.

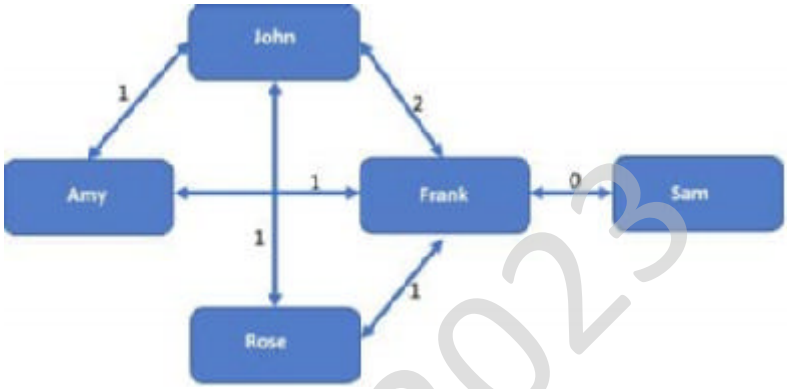


Рисунок 3.2 – Граф соціальної мережі для зав’язків типу «дружба»

Вага ребер – у зваженій мережі ребро несе мітку (вагу) між вузлами. В аналізі соціальних мереж вага може визначати кількість взаємних зв’язків між вузлами, з’єднаними цим ребром.

Важливим поняттям є щільність зв’язків у соціальній мережі (рис. 3.3).

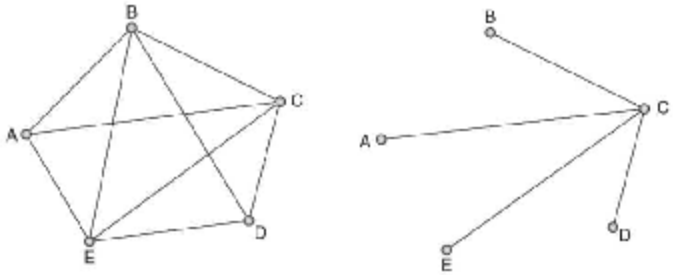


Рисунок 3.3 – Приклади соціальної мережі з: a) високою щільністю зв’язків, b) низькою щільністю зв’язків

Щільність зв'язків – відношення між кількістю існуючих з'єднань у мережі та всіма можливими з'єднаннями в мережі, обчислюється наступним чином:

$$Density = Actual\_Connections / Potential\_Connections$$

де $Potential\_Connections = n*(n-1)/2$, n = кількість вузлів у мережі.

На рис. 3.3 мережа з п'яти точок/вузлів. Загальна кількість можливих з'єднань у цій мережі дорівнює 10. Граф на рис. 3.a має дев'ять ребер; його щільність 90%. Отже, це мережа високої щільності. У той час як граф на рис. 3.b має лише чотири ребра, він має низьку щільність 40%.

Реальні мережі

Реальні мережі, і зокрема соціальні мережі, мають унікальну структуру, яка часто відрізняється від випадкових математичних мереж (рис. 3.4):

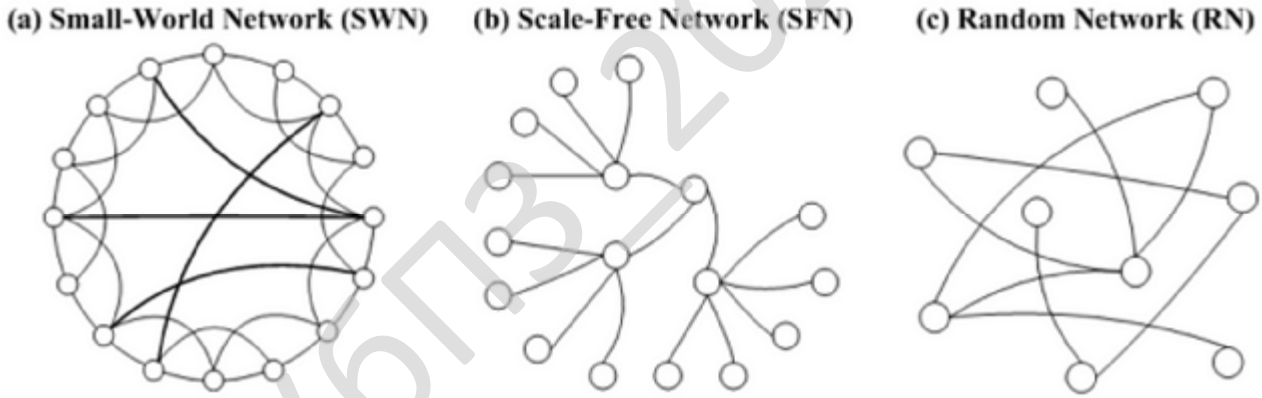


Рисунок 3.4 – Приклади складних мереж

Основні властивості складних мереж:

– **Феномен Small World** стверджує, що реальні мережі часто мають дуже короткі шляхи (щодо кількості переходів) між будь-якими підключеними членами мережі. Це стосується реальних і віртуальних соціальних мереж (теорія шести рукостискань) і фізичних мереж, таких як аеропорти або електрика маршрутів веб-трафіку.

– **Безмасштабні мережі** зі степеневим розподілом мають розшаровану популяцію з декількома високозв'язаними вузлами (наприклад, соціальні впливи) і великою кількістю слабко зв'язаних вузлів.

– **Гомофілія –** це схильність індивідів об'єднуватися та зв'язуватися з подібними іншими, що призводить до подібних властивостей серед сусідів.

Одною з найважливіших характеристик структурної позиції користувача соціальної мережі є впливовість.

Впливовість (влада) – це характеристика місця розташування користувача у віртуальній соціальній мережі, що дає йому здатність впливати або напряму контролювати поведінку інших, а також вказує на володіння ним певними соціальними ресурсами, які дозволяють контролювати потоки інформації в мережі, та (або) уникати такого контролю з боку інших користувачів.

Мірою влади є центральність вершини графу соціальної мережі, яку представляє користувач.

Існують різні центральності. Розглянемо найбільш відомі та вживані.

Центральність за степенем дозволяє виділити користувачів, які пов'язані з максимальною кількістю інших учасників мережі. Користувач, який має більше зв'язків з іншими, знаходиться в більш вигідному становищі. Велика кількість зв'язків дає йому більше альтернатив для знаходження ресурсів і розповсюдження свого інформаційного впливу і, водночас, меншу залежність від кожного суб'єкта мережі.

Індекс центральності за степенем для мереж ненаправлених відношень обчислюють за формулою:

$$C_D(n_i) = d(n_i), \tag{3.1}$$

де $d(n_i)$ – степінь вузла n_i.

На практиці частіше використовують стандартизовану міру центральності користувача за степенем:

$$C_D'(n_i) = \frac{d(n_i)}{N}, \tag{3.2}$$

де N – загальна кількість акторів у мережі.

Можна враховувати центральність користувачів, з якими пов'язаний поточний досліджуваний користувач. Це називається центральністю за престижністю:

$$C_P(n_i) = \sum_{i=1}^{N} d(n_i) \cdot C_C(n_i). \quad (3.3)$$

Центральність за степенем є ефективною для мереж із «зіркоподібним» розташуванням акторів. Але для лінійних структур її використання може призвести до помилкових висновків щодо впливовості користувачів. В таких мережах вимірювання центральності доцільно здійснювати на основі близькості.

Дослідження *центральності за близькістю* дозволяє виділити коритсувачів, через яких проходить максимальна кількість найкоротших шляхів, які сполучають між собою інших учасників мережі. Визначення центральності за близькістю базується на понятті *найменшої відстані* між користувачами – $d(n_i, n_j)$. Чим меншими є відстані від даного користувача до всіх інших, тим більш владним він є. Найпростіший показник центральності за близькістю:

$$C_c(n_i) = \sum_{i=1}^{N} d(n_i, n_j). \quad (3.4)$$

Центральність за близькістю дозволяє виявити користувачів мережі, що мають максимальну незалежність від інформаційних впливів інших учасників мережі. Стандартизований індекс центральності за близькістю підраховується за формулою:

$$C_C'(n_i) = \sum_{i=1}^{N} \frac{1}{d(n_i, n_j)}. \quad (3.5)$$

Даний індекс підраховується тільки у зв'язних графах. Для ізольованих акторів $d(n_i,n_j) \to \infty$, що робить неможливим обчислення даного індексу центральності. Як можливий спосіб подолання вказаного недоліку можна підрахувати індекс центральності за близькістю з поправкою, що враховує *сферу впливу* кожного користувачаа. Сфера впливу користувача включає всіх інших користувачів мережі, які є досяжними для даного. Індекс підраховується за формулою:

$$C_C^*(n_i) = \frac{J_i/(N-1)}{\sum d(n_i,n_j)/J_i} \quad (3.6)$$

де J_i – кількість акторів у сфері впливу актора i.

Даний індекс може бути корисним для вивчення феномену соціального кола, тобто осіб, що є досяжними для даногокористувача. При цьому є можливість визначення кіл різного порядку: 1-го – особи, що мають відстань від даної довжиною 1, 2-го – довжиною 2 і т. д.

Індекс центральності за близькістю вказує наскільки користувач концентрує на собі прямі та непрямі зв'язки в мережі та може розглядатись як міра незалежності актора від інформаційних впливів з боку інших учасників мережі.

Центральність власного вектора відносна міра важливості вузла в мережі. Кожному вузлу присвоюється значення або бал залежно від кількості інших відомих вузлів/вузлів з високим балом, до яких він підключений.

Реальні приклади використання аналізу соціальних мереж:

1. Управління ланцюгом постачання. Ланцюг постачання можна моделювати як мережу відносин постачальник/споживач. Аналіз мережі в ланцюзі поставок допомагає підвищити ефективність роботи шляхом виявлення та усунення менш важливих вузлів (постачальників/складів). Це може допомогти визначити ключові вузли в мережі та створити резерв у кризових чи надзвичайних ситуаціях. До вузлів належать роздрібні торговці, постачальники, склади, транспортери, регуляторні органи. Програми аналізу соціальних мереж можуть допомогти виробникам ідентифікувати більш критичні з точки зору роботи вузли та визначити потенційні джерела для збільшення кількості підключень до постачальників. Це також може допомогти виявити будь-які вузькі місця в процесі постачання та управлінні запасами.

2. Людські ресурси. HRM часто прагне визначити критичні ресурси та зрозуміти їхній внесок в організаційний потік, співпрацю, участь та потік інформації. Дотримуючись аналізу організаційної мережі (ONA), організація оптимізує зв'язки талантів, продуктивність і використання. Це також допоможе визначити охоплення особи, ідентифікувати прискорювачі зростання та погано підключені ресурси, а також вирішити, кому надати більше можливостей.

3. Передача інфекційних захворювань. Аналіз соціальних мереж може допомогти ідентифікувати та ізолювати осіб і групи з високим ступенем зв'язку та центральності (передувачі хвороби) і впровадити обґрунтовані дії з відстеження контактів, щоб пом'якшити вплив. Окрім відстеження контактів, аналіз соціальних мереж також може ідентифікувати домінуючі теми та зв'язки між ключовими словами та визначити почуття.

4. Фінанси, виявлення шахрайства. Фінансові організації можуть використовувати аналіз соціальних мереж для виявлення шахрайства. Шахрайство часто організовується групами людей, слабко пов'язаних між собою. Таке відображення мережі дозволить фінансовим установам ідентифікувати клієнтів, які можуть мати стосунки з окремими особами чи організаціями в їхньому списку спостереження (мережі), і вживати запобіжних заходів. Аналіз соціальних мереж також можна використовувати для заборони доступу до потенційних хакерських мереж, виявлення групи шахраїв і серії грошових операцій, які можуть бути пов'язані з діяльністю відмивання грошей.

Для генерації структури соціальної мережі було використано **модель Воттса-Строгаца** та удосконалено її шляхом додавання симуляції характеристик користувачів і перепідключенням ребер графу з урахуванням схожості вузлів мережі.

Модель Ваттса-Строгатца (Watts-Strogatz model) використовується для генерації малого світу (small-world) графів, які характеризуються короткими середніми шляхами між вершинами та високою класетризацією (кількість зв'язків між сусідніми вершинами). Алгоритм на основі цієї моделі дозволяє моделювати графи, які відображають деякі властивості реальних соціальних та інформаційних мереж.

Принцип роботи алгоритму Ваттса-Строгатца наступний:

1. Початково створюється кілька вершин (вузлів) із заданою кількістю зв'язків між ними. Ця початкова структура може бути створена за допомогою

регулярного графа, де кожна вершина з'єднана з певною кількістю сусідніх вершин.

2. Потім алгоритм проходить кожен зв'язок у графі і має можливість "перепідключити" його до іншої вершини з певною ймовірністю **p**. Це робить граф більш "малосвітовим", оскільки додає короткі випадкові зв'язки між вершинами.

3. Результатом роботи алгоритму є граф, де збережена початкова структура регулярного графа з випадковими короткими зв'язками. Такий граф має своєрідну "малосвітову" структуру, де більшість вершин все ще знаходиться близько одна до одної через короткі шляхи, але є деякі випадкові зв'язки, які роблять його більш "малосвітовим".

Алгоритм Ваттса-Строгатца дозволяє досліджувати властивості графів, які спостерігаються у реальних мережах, таких як соціальні мережі та мережі зв'язку. Він допомагає розуміти, як властивості мережі впливають на поширення інформації, досліджувати шляхи та відстані між вершинами, а також моделювати різні сценарії розвитку мережі.

Для візуалізації графу соціальної мережі було використано різні алгоритми, зокрема, **алгоритмом Fruchterman-Reingold** та візуалізацію у вигляді **графа Лапласа**.

Алгоритм візуалізації графа Fruchterman-Reingold (або просто FR-алгоритм) є одним із найпопулярніших алгоритмів для розкладання графа у двовимірному просторі. Він використовується для візуалізації графів таким чином, щоб близькі вершини за графовою структурою знаходилися близько одна до одної на візуалізованому графі, та змінює розташування вершин у просторі так, щоб уникнути їх пересічень.

Основний принцип роботи алгоритму FR можна описати так:

1. Кожна вершина графа представляється як точка у двовимірному просторі. Початкове розташування вершин може бути випадковим або заданим за певними правилами.

2. В алгоритмі існують дві основні сили: сила відштовхування і сила притягування.

– *Сила відштовхування* відповідає за уникнення пересічень між вершинами. Вона обчислюється на основі закону Кулона-Кулона: вершини, які знаходяться ближче одна до одної, відштовхуються.

– *Сила притягування* між вершинами, які повинні бути близькими у візуалізації (наприклад, зв'язані вершини), спрямована на зведення їх разом. Вона обчислюється на основі закону Гука: вершини, які мають зв'язок, притягуються одна до одної.

3. На кожному кроці алгоритму обчислюються всі сили відштовхування та притягування для кожної вершини. Потім вершини переміщуються на певну відстань у напрямку вектора сил, з урахуванням параметра "температури" (параметр, що поступово зменшується з кожним кроком для зменшення рухливості вершин).

4. Процес переміщення вершин повторюється доки система не досягне стану рівноваги, коли всі сили відштовхування і притягування збалансовані, і вершини знаходяться в стабільному розташуванні.

Основна ідея FR-алгоритму – розмістити вершини графа у такий спосіб, щоб близькі вершини за структурою графа були близько одна до одної, а далекі вершини – далеко одна від одної. Цей алгоритм дозволяє створювати чітко читабельні візуалізації графів для подальшого аналізу та візуалізації.

Алгоритм візуалізації графа у вигляді графа Лапласа є досить складним, і основна його ідея полягає в тому, щоб відобразити граф у просторі так, щоб властивості графа, які відображаються в його матриці Лапласа, були відображені у просторі візуалізації. Матриця Лапласа графа має важливу структурну інформацію, і вона може бути використана для створення подібної структури в просторі візуалізації.

Основні кроки алгоритму візуалізації графа у вигляді графа Лапласа:

1. Обчислення матриці Лапласа для заданого графа. Матриця Лапласа L графа зазвичай обчислюється як різниця матриці ступенів D та матриці суміжності A:

$$L = D - A$$

де D – діагональна матриця ступенів вершин, а A – матриця суміжності графа.

2. Обчислення власних векторів та власних значень матриці Лапласа. Це може бути зроблено за допомогою чисельних методів, таких як метод степенів або метод QR-розкладу. Власні вектори та власні значення матриці Лапласа дозволяють отримати структуру простору для візуалізації.

3. Відображення вершин у просторі на основі власних векторів. Кожен власний вектор може бути розглянути як нова координата вершини графа в просторі візуалізації. Для візуалізації можна використовувати перші кілька власних векторів, які відповідають найбільшим власним значенням.

4. Візуалізація зв'язків між вершинами на основі матриці суміжності графа. Матриця суміжності A графа може бути використана для розташування зв'язків між вершинами в просторі візуалізації. Зв'язки можна представити лініями або кривими, які з'єднують відповідні вершини.

5. Візуалізація та налаштування графічного відображення. Після обчислення координат вершин та розташування зв'язків, можна створити графічне відображення графа Лапласа з використанням бібліотек для візуалізації.

Цей алгоритм може бути використаний для створення візуалізацій графів, щоб відобразити структурну інформацію графа, таку як розподіл вершин, спектральні властивості графа та зв'язки між вершинами.

3.2 Розробка структурної схеми

Структурна схема розробленого програмного забезпечення зображена на рисунку 4.4.

Рисунок 4.4 – Структурна схема системи

З рисунку видно, що система складається з наступних частин:

- Генерація структури соціальної мережі.
- База даних структур різних соціальних мереж.
- Завантаження структури соціальної мережі з відкритих датасетів.
- Структури даних, що зберігають граф соціальної мережі та характеристики користувачів.
- Моделювання інформаційних процесів в соціальній мережі.
- Аналіз соціальної мережі.
- Аналіз структури мережі.
- Визначення співтовариств.
- Характеристики вузлів.
- Характеристики мережі.
- Візуалізація структури соціальної мережі.

- Розфарбовування графу.
- Розташування вершин графу.
- Аналіз поширення вірусної інформації у мережі.
- База даних звітів.

3.3 Розробка функціональної схеми

Функціональна схема системи наведена на рис. 3.4.

Як видно з рисунку, у розробленій системі є наступні модулі:

– Модуль моделювання соціальних мереж.

– Модуль візуалізації соціальних мереж.

– Модуль аналізу соціальних мереж.

Модуль моделювання соціальних мереж містить функції для:

– Моделювання структури соціальної мережі.

– Моделювання користувачів соціальної мережі та їх властивостей і станів.

– Моделювання обміну інформацією між користувачами.

– Моделювання зміни станів користувачів соціальної мережі.

Модуль візуалізації соціальних мереж містить функції для:

– Візуалізації графу соціальної мережі з випадковим розташуванням вершин.

– Візуалізації графу соціальної мережі у вигляді графа Лапласа.

– Візуалізації графу соціальної мережі алгоритмом Fruchterman-Reingold.

– Розфарбовування графу соціальної мережі для її візуального аналізу.

Модуль аналізу соціальних мереж містить функції для:

– Визначення центральностей вузлів соціальної мережі.

– Визначення подоби користувачів соціальної мережі.

– Визначення кластерів соціальної мережі.

– Визначення станів користувачів соціальної мережі при поширенні вірусної інформації.

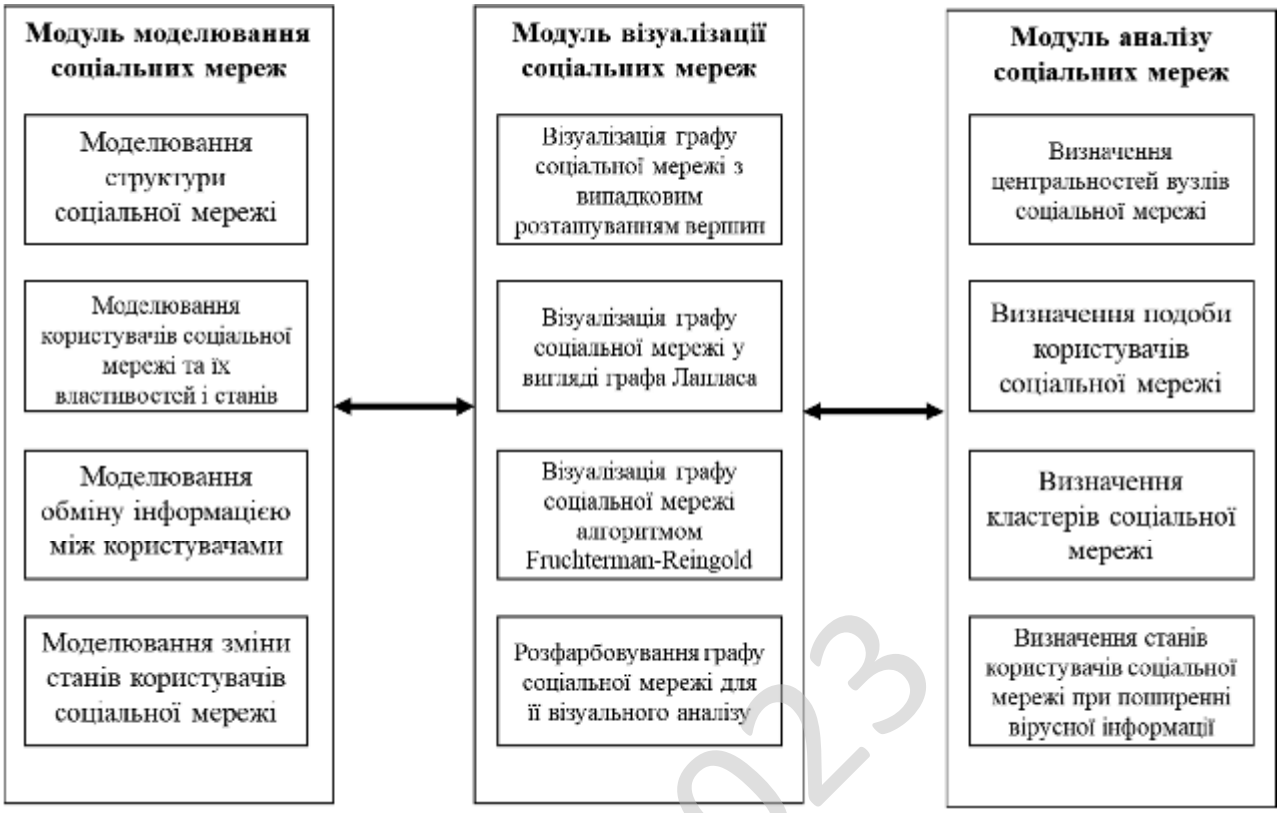


Рисунок 3.4 – Функціональна схема системи

3.4 Розробка діаграми процесів

На рисунку 3.5 зображена діаграма процесів, що описує наявні у системі процеси та їх взаємодію.

У розробленій системі присутні наступні процеси:

- Головне вікно програми;
- Моделювання соціальної мережі;
- Граф соціальної мережі;
- Генерація структури соціальної мережі;
- Завантаження структури соціальної мережі;
- Візуалізація соціальної мережі;
- Розфарбовування графу;
- Розташування вершин графу;

- Моделювання поширення вірусної інформації;
- Аналіз соціальної мережі;
- Визначення центральності вузлів;
- Визначення подоби користувачів;
- Визначення кластерів (співтовариств) ;
- Визначення параметрів мережі;
- Звіт про аналіз соціальної мережі.

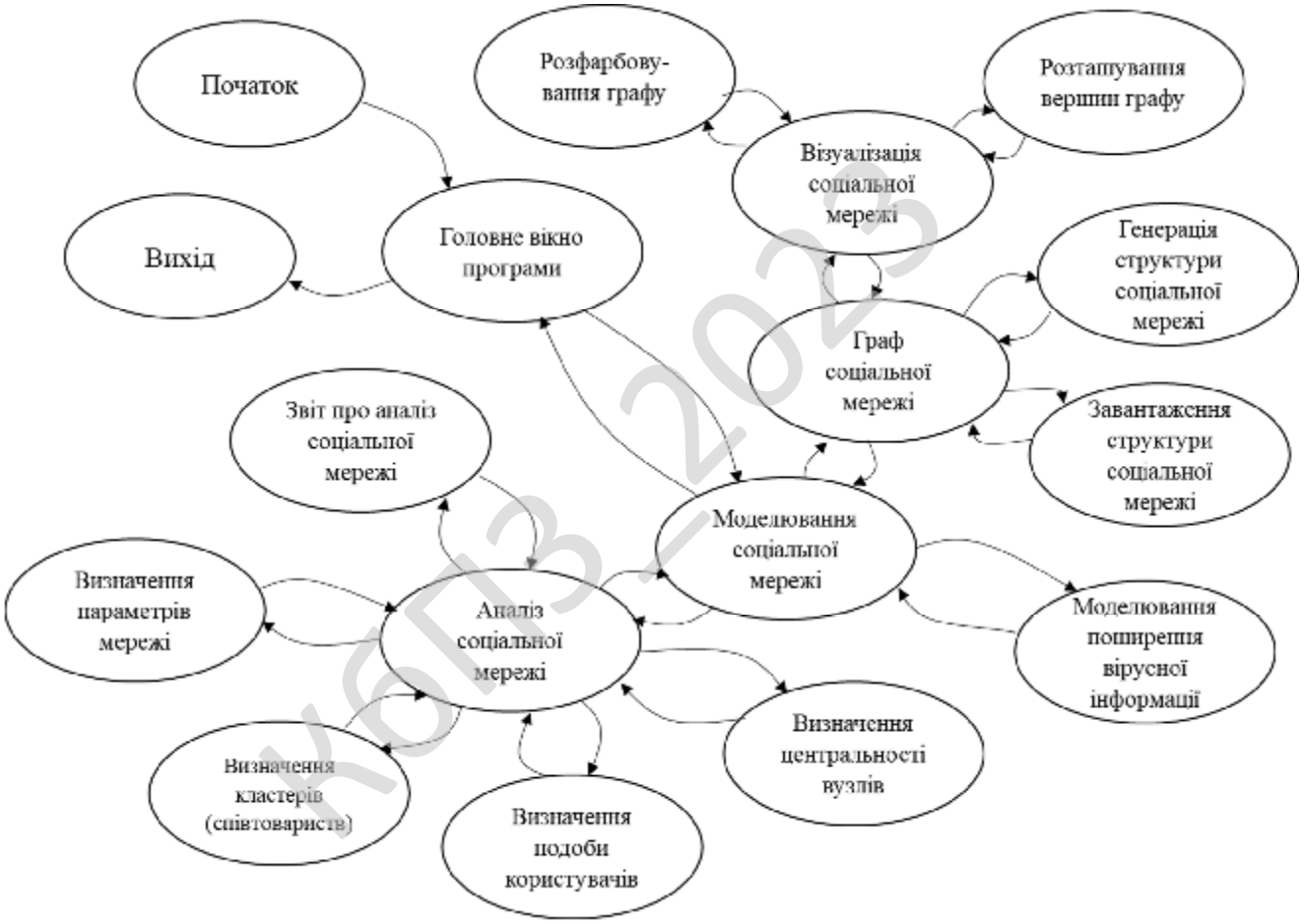


Рисунок 3.5 – Діаграма процесів системи

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рис. 4.1 показана блок-схема основної програми.

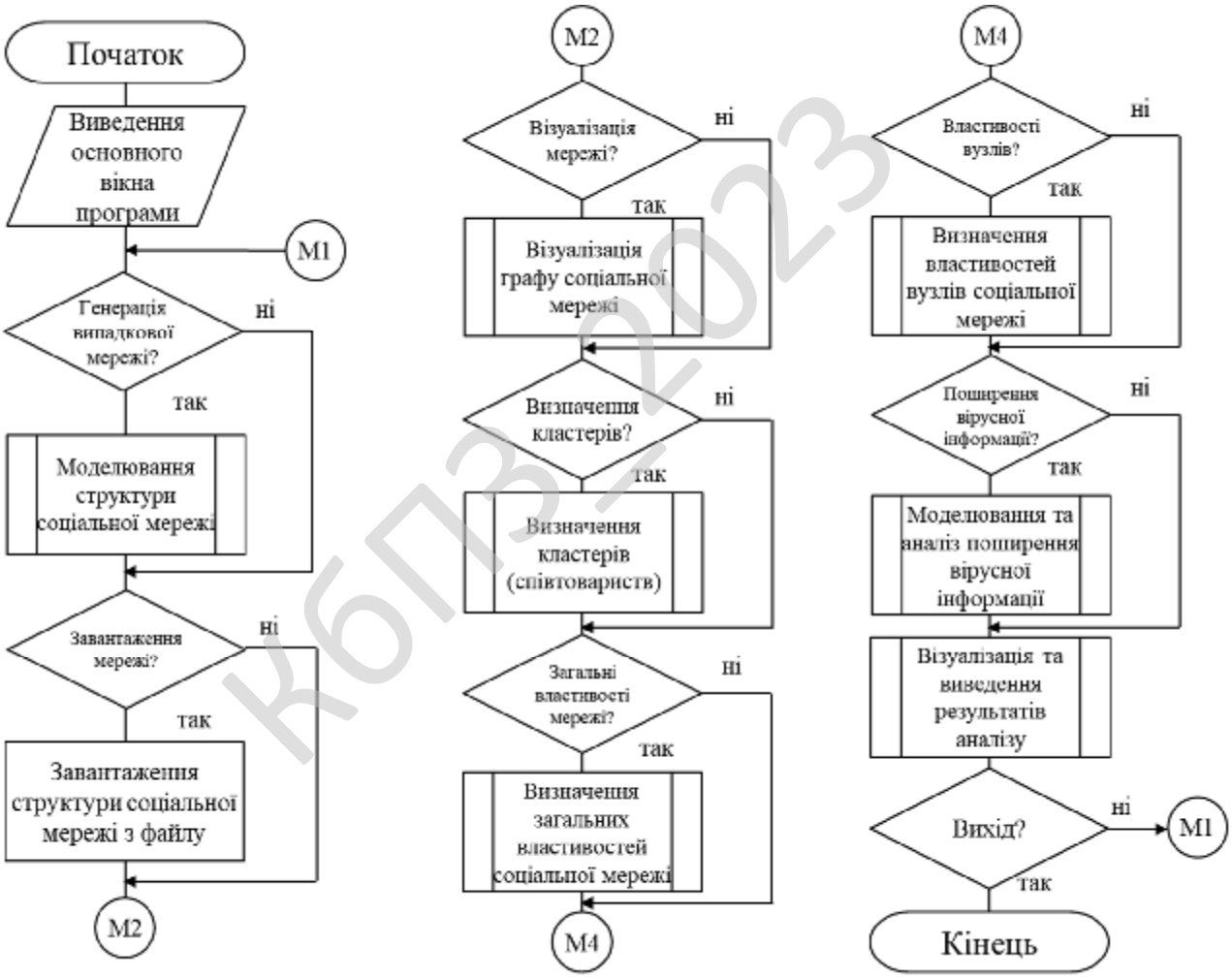


Рисунок 4.1 – Блок-схема основної програми

Основними функціями розробленого програмного забезпечення є моделювання віртуальної соціальної мережі та аналіз її властивостей і процесів.

Як видно з рисунку, алгоритм роботи основної програми складається з наступних кроків:

Крок 1. Виведення основного вікна програми.

Крок 2. Якщо користувач обирає генерацію випадкової структури соціальної мережі, то виконується крок 3 інакше відбувається перехід на крок 4.

Крок 3. Запуск підпрограми моделювання структури соціальної мережі.

Крок 4. Якщо користувач обирає завантаження структури соціальної мережі, то виконується крок 5 інакше відбувається перехід на крок 6.

Крок 5. Завантаження структурної соціальної мережі з файлу.

Крок 6. Якщо користувач обирає візуалізацію структури соціальної мережі, то виконується крок 7 інакше відбувається перехід на крок 8.

Крок 7. Запуск підпрограми візуалізації структури соціальної мережі.

Крок 8. Якщо користувач обирає визначення кластерів соціальної мережі, то виконується крок 9 інакше відбувається перехід на крок 10.

Крок 9. Запуск підпрограми визначення кластерів (співтовариств).

Крок 10. Якщо користувач обирає визначення загальних властивостей структури соціальної мережі, то виконується крок 11 інакше відбувається перехід на крок 12.

Крок 11. Запуск підпрограми визначення загальних властивостей структури соціальної мережі.

Крок 12. Якщо користувач обирає визначення властивостей вузлів соціальної мережі, то виконується крок 13 інакше відбувається перехід на крок 14.

Крок 13. Запуск підпрограми визначення властивостей вузлів соціальної мережі.

Крок 14. Якщо користувач обирає моделювання та аналіз поширення вірусної інформації, то виконується крок 15 інакше відбувається перехід на крок 16.

Крок 15. Запуск підпрограми моделювання та аналіз поширення вірусної інформації.

Крок 16. Візуалізація та виведення результатів аналізу.

Крок 17. Якщо користувач обирає вихід з програми, то відбувається завершення роботи програмного забезпечення, інакше відбувається перехід на крок 2.

Моделювання та аналіз поширення вірусної інформації у віртуальній соціальній мережі у розробленому програмному забезпеченні реалізоване на основі моделей SIR та SIRS.

Модель SIR (Susceptible-Infectious-Recovered) є однією з класичних моделей для моделювання поширення вірусної інформації, і вона також може бути використана для моделювання поширення інфекційних хвороб або вірусної інформації в соціальних мережах. Ця модель розглядає індивідуальні члени спільноти як одну з трьох категорій: схильних до інфекції (Susceptible), інфікованих (Infectious) та відновлених (Recovered).

Модель SIRS (Susceptible-Infectious-Recovered-Susceptible) подібна до моделі SIR, проте має додатковий стан, який відображає особливість рецидивів чи повторного стану схильності до інфекції після одужання. Основна відмінність між моделями SIR і SIRS полягає в тому, що в моделі SIR відновлені особи стають імунними до інфекції назавжди, тоді як у моделі SIRS вони можуть знову стати схильними до інфекції після певного часу.

Схильні до інфекції (Susceptible) – це особи, які ще не інфіковані і можуть заразитися, якщо вони зустрінуться з інфікованими.

Види користувачів нейронної мережі у моделі поширення вірусної інформації (або вірусів) SIRS:

Інфіковані (Infectious) – це особи, які інфіковані і можуть поширювати інфекцію (вірусну інформацію) на схильних осіб.

Відновлені (Recovered) – це особи, які одужали від інфекції і стали імунними до неї на певний час.

Знову схильні до інфекції (Susceptible) – після певного часу відновлені особи можуть знову стати схильними до інфекції (повернутися до стану схильності

до інфекції).

На рис. 4.2 показана блок-схема алгоритму роботи підпрограми поширення вірусної інформації.

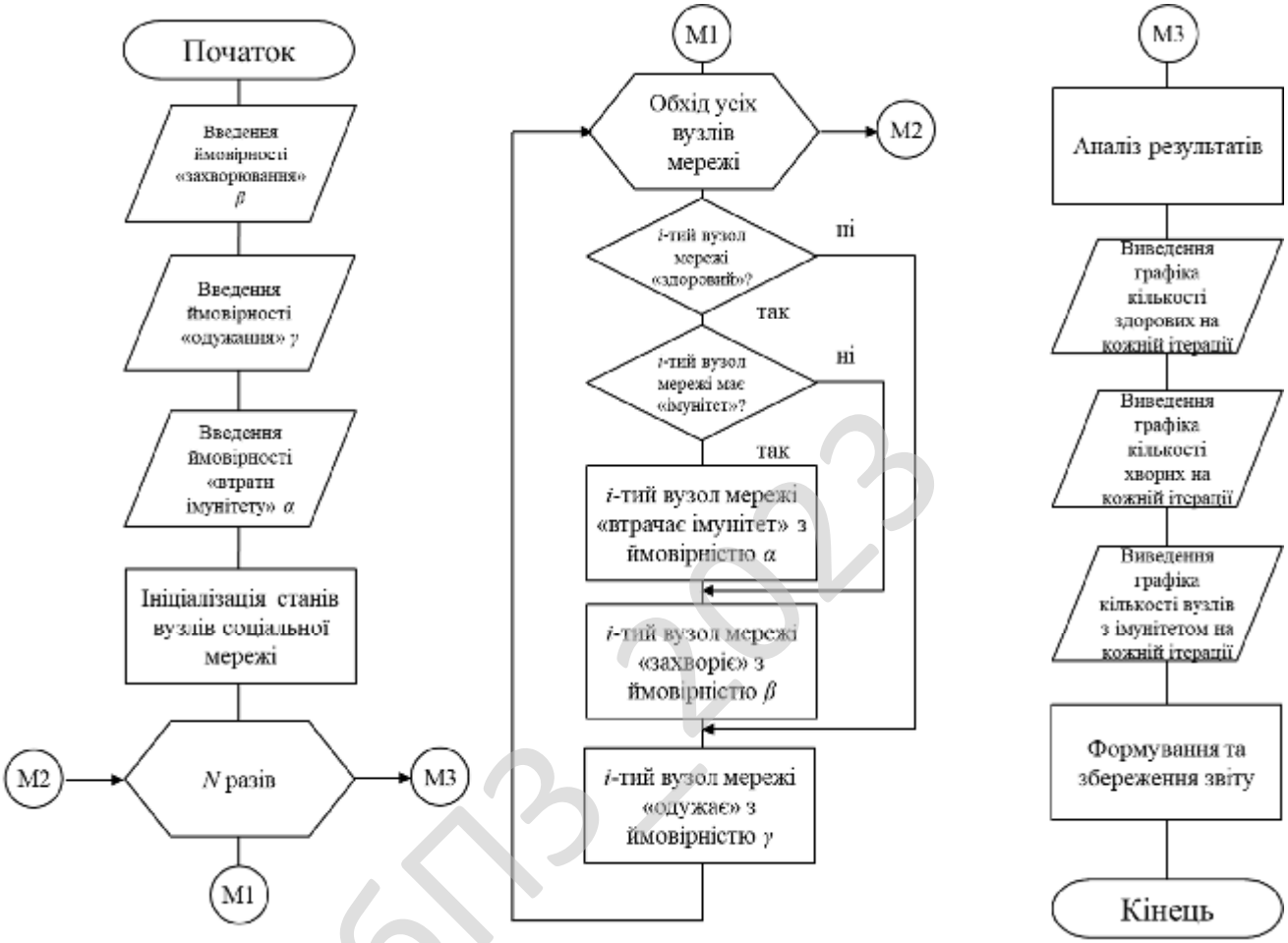


Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми поширення вірусної інформації

Алгоритм роботи моделі SIRS:

1. Ініціалізація. Спочатку визначається кількість осіб в соціальній мережі та початкова кількість інфікованих осіб. Вказуються ймовірності, які впливають на модель: імовірність захворювання (β), імовірність одужання (γ) та імовірність втрати імунітету (α).

2. Ітерації. Модель працює за допомогою дискретних часових ітерацій. На кожній ітерації робляться наступні дії:

– Кожна особа в соціальній мережі може захворіти від інфікованих осіб, з якими пов’язана, з імовірністю β. Інфіковані особи залишаються інфікованими.

– Кожна інфікована особа може одужати з імовірністю γ.

– Кожна одужала особа може втратити імунітет і стати вразливою до інфекції з імовірністю α.

3. Запуск ітерацій. Ітерації повторюються протягом певної кількості часу або до тих пір, поки кількість інфікованих осіб не стабілізується.

4. Аналіз результатів. Після закінчення ітерацій відбувається аналіз результатів, включаючи кількість інфікованих, одужавших і вразливих осіб на кожній ітерації. Вивчається динаміка імунітету та поширення інфекції в соціальній мережі.

Для генерації структури віртуальної соціальної мережі розроблено наступний код:

```
import networkx as nx
import random

import numpy as np
import matplotlib.pyplot as plt

import community  # Бібліотека для використання алгоритму Лувейна
# Параметри графа
n = 70  # Кількість користувачів (вузлів)
k = 4   # Початковий ступінь вузла
p = 0.7 # Ймовірність перепідключення замість додавання ребра


# Генерація рис користувачів (10 випадкових чисел для кожного)
user_attributes = {i: np.random.rand(10) for i in range(n)}


# Функція для обчислення коефіцієнта кореляції Пірсона між рисами
користувачів
def compute_similarity(user1, user2):
    return np.corrcoef(user1, user2)[0, 1]


# Створення початкового регулярного графа
G = nx.Graph()
for i in range(n):
    for j in range(1, k // 2 + 1):
        G.add_edge(i, (i + j) % n)
```

```
# Перепідключення ребер з урахуванням схожості користувачів
for i in range(n):
    for j in range(1, k // 2 + 1):
        if random.random() < p:
            new_neighbor = random.randint(0, n - 1)
            while new_neighbor == i or G.has_edge(i, new_neighbor):
                new_neighbor = random.randint(0, n - 1)
            similarity = compute_similarity(user_attributes[i],
user_attributes[new_neighbor])
            connection_probability = similarity
            if random.random() < connection_probability:
                G.remove_edge(i, (i + j) % n)
                G.add_edge(i, new_neighbor)
```

Для моделювання поширення вірусної інформації у віртуальній соціальній мережі на основі моделі SIR розроблено наступний код:

```
# Ініціалізуємо стани SIR моделі
infected_nodes = random.sample(list(G.nodes()), k=2)# Початкові інфіковані
infected_nodes = random.sample(list(G.nodes()), k=2)
susceptible_nodes = set(G.nodes()) - set(infected_nodes)
recovered_nodes = set()


# Параметри моделі
beta = 0.2  # Імовірність передачі вірусу
gamma = 0.3  # Імовірність одужання


# Функція для виводу графа з кольорами відповідно до стану
def plot_graph(G, infected_nodes, susceptible_nodes, recovered_nodes):
    node_colors = []
    for node in G.nodes():
        if node in infected_nodes:
            node_colors.append('red')
        elif node in susceptible_nodes:
            node_colors.append('blue')
        else:
            node_colors.append('green')

    nx.draw(G, with_labels=True, node_color=node_colors)
    plt.show()


# Візуалізуємо початковий стан графа
plot_graph(G, infected_nodes, susceptible_nodes, recovered_nodes)



# Моделюємо поширення вірусу
iterations = 10
for t in range(iterations):
    new_infections = set()
    new_recoveries = set()
```

```
    for node in infected_nodes:
        for neighbor in G.neighbors(node):
            if neighbor in susceptible_nodes and random.random() < beta:
                new_infections.add(neighbor)
        if random.random() < gamma:
            new_recoveries.add(node)


    # infected_nodes |= new_infections
    infected_nodes = set(infected_nodes) | new_infections
    recovered_nodes |= new_recoveries
    susceptible_nodes -= new_infections


    # Візуалізуємо граф після кожної ітерації
    plot_graph(G, infected_nodes, susceptible_nodes, recovered_nodes)
```

Для моделювання поширення вірусної інформації у віртуальній соціальній мережі на основі моделі SIRS розроблено наступний код:

```
# Параметри моделі
initial_infections = random.sample(range(100), k=5)  # Початкові інфіковані
recovery_probability = 0.3  # Імовірність одужання
loss_immunity_probability = 0.1  # Імовірність втрати імунітету
vulnerability = [random.uniform(0, 1) for _ in range(n)]  # Вразливість
користувачів до вірусу


# Початковий стан графа
node_states = {node: "S" for node in G.nodes()}
for node in initial_infections:
    node_states[node] = "I"

# Ініціалізація даних для графіків
infection_counts = []
immunity_counts = []




# Функція для візуалізації графа
def plot_graph(G, node_states):
    node_colors = {"S": "blue", "I": "red", "R": "green"}
    colors = [node_colors[state] for state in node_states.values()]
    pos = nx.spring_layout(G)
    nx.draw(G, pos, node_color=colors, with_labels=True)
    plt.show()


# Проведення ітерацій
num_iterations = 50
for t in range(num_iterations):
    new_infections = set()
    new_recoveries = set()

    for node in G.nodes():
        if node_states[node] == "I":
            # Інфікований може одужати з імовірністю recovery_probability
```

```
                if random.random() < recovery_probability:
                    new_recoveries.add(node)

            if node_states[node] == "R":
                # "Відновлений" може втратити імунітет з імовірністю
loss_immunity_probability
                if random.random() < loss_immunity_probability:
                    node_states[node] = "S"

            if node_states[node] == "S":
                # Сприйнятливий може заразитися від інфікованих сусідів
                for neighbor in G.neighbors(node):
                    if node_states[neighbor] == "I":
                        if random.random() < vulnerability[node]:  # Імовірність
заразитися від сусіда
                            new_infections.add(node)
                            break

        for node in new_infections:
            node_states[node] = "I"
        for node in new_recoveries:
            node_states[node] = "R"


     # Візуалізація графа на кожній ітерації
        plot_graph(G, node_states)


      # Рахунок кількості інфікованих та осіб з імунітетом
        infection_counts.append(list(node_states.values()).count("I"))
        immunity_counts.append(list(node_states.values()).count("R"))



    # Графіки поширення інфекції та імунітету у часі
    plt.plot(range(num_iterations), infection_counts, label="Infections")
    plt.plot(range(num_iterations), immunity_counts, label="Immunity")
    plt.xlabel("Time")
    plt.ylabel("Count")
    plt.legend()
    plt.show()
```

Для визначення кластерів (співтовариств) у віртуальній соціальній мережі розроблено наступний код:

```
    # Знаходження співтовариств у графі за допомогою алгоритму Лувейна
    partition = community.best_partition(G)


    # Визначення позицій вершин з одного співтовариства поруч, але не
    ідентичними
    pos = nx.spring_layout(G, seed=42)

    for comm in set(partition.values()):
        nodes_in_comm = [node for node, comm_id in partition.items() if comm_id
    == comm]
        x_positions = [pos[node][0] for node in nodes_in_comm]
        y_positions = [pos[node][1] for node in nodes_in_comm]
        avg_x = sum(x_positions) / len(x_positions)
```

```
        avg_y = sum(y_positions) / len(y_positions)
        for node in nodes_in_comm:
            pos[node] = (avg_x + random.uniform(-0.1, 0.1), avg_y +
    random.uniform(-0.1, 0.1))


    plt.figure(figsize=(8, 6))
    cmap = plt.get_cmap("viridis", max(partition.values()) + 1)
    node_colors = [cmap(partition[node]) for node in G.nodes()]
    node_sizes = [10 * G.degree[node] for node in G.nodes()]


    # Візуалізація графа з вершинами з одного співтовариства поруч
    nx.draw(G, pos, with_labels=False, node_size=node_sizes,
    node_color=node_colors, edge_color='gray', width=0.5, alpha=0.7)
    plt.show()

    # Виведення співтовариств і списків вершин у них текстом
    print("Співтовариства:")
    communities = {}
    for node, comm_id in partition.items():
        if comm_id not in communities:
            communities[comm_id] = []
        communities[comm_id].append(node)

    for comm_id, nodes in communities.items():
        print(f"Співтовариство {comm_id}: {nodes}")
```

Для визначення центральностей вузлів віртуальної соціальної мережі розроблено наступний код:

```
    def degree_centrality_size(G):
        # Визначення центральності за степенем
        degree_centrality = nx.degree_centrality(G)

        # Визначення розмірів вершин на основі центральності
        node_sizes = [2000 * degree_centrality[node] for node in G.nodes()]

        return degree_centrality, node_sizes


    def closeness_centrality_size(G):
        # Визначення центральності за близькістю
        closeness_centrality = nx.closeness_centrality(G)


      # Визначення розмірів вершин на основі центральності
        node_sizes = [2000 * closeness_centrality[node] for node in G.nodes()]

        return closeness_centrality, node_sizes


    def eigenvector_centrality_size(G):
        # Визначення центральності за власним вектором
        eigenvector_centrality = nx.eigenvector_centrality(G)
```

```
    # Визначення розмірів вершин на основі центральності
    node_sizes = [2000 * eigenvector_centrality[node] for node in
G.nodes()]

    return eigenvector_centrality, node_sizes


# Визначення позицій вершин
pos = nx.spring_layout(G, seed=42)


# Визначення центральності за степенем та розмірів вершин
degree_centrality, node_sizes = degree_centrality_size(G)

plt.figure(figsize=(8, 6))


# Візуалізація графа з вершинами різних розмірів
nx.draw(G, pos, with_labels=False, node_size=node_sizes,
node_color='skyblue', edge_color='gray', width=0.5, alpha=0.7)


# Виведення номерів вершин та їх центральностей текстом
print("Центральність вершин:")
for node, centrality in degree_centrality.items():
    print(f"Вершина {node}: Центральність = {centrality:.2f}")

plt.show()
```

Для визначення загальних властивостей графу віртуальної соціальної мережі розроблено наступний код:

```
# Визначення параметрів графу
num_nodes = len(G.nodes())
num_edges = len(G.edges())
average_degree = 2 * num_edges / num_nodes  # Середній ступінь вузла
clustering_coefficient = nx.average_clustering(G)  # Коефіцієнт
кластеризації
diameter = nx.diameter(G)  # Діаметр графу
density = nx.density(G)  # Густина зв'язків

# Виведення результатів
print(f"Кількість вузлів: {num_nodes}")
print(f"Кількість ребер: {num_edges}")
print(f"Середній ступінь вузла: {average_degree}")
print(f"Коефіцієнт кластеризації: {clustering_coefficient}")
print(f"Діаметр графу: {diameter}")
print(f"Густина зв'язків: {density}")
```

Для візуалізації графу соціальної мережі алгоритмом Fruchterman-Reingold розроблено наступний код:

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
```

```
# Параметри графа
n = 20  # Кількість вузлів
k = 4   # Початковий ступінь вузла
p = 0.3 # Ймовірність перепідключення замість додавання ребра


# Генерація графа за моделлю Ваттса-Строгатца
G = nx.watts_strogatz_graph(n, k, p)


# Візуалізація графу за допомогою алгоритму Fruchterman-Reingold
pos = nx.spring_layout(G, seed=42)  # seed для відтворюваності розташування
вузлів
nx.draw(G, pos, with_labels=True, node_size=500, node_color='skyblue',
font_size=12, font_color='black', font_weight='bold', edge_color='gray',
width=2)


# Налаштування відображення
plt.axis('off')  # Вимкнення відображення координатних вісей
plt.show()
```

Для запису структури графу віртуальної соціальної мережі в базу даних Neo4j розроблено наступний код:

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

from py2neo import Graph # бібліотека для роботи з базою даних Neo4j


# Параметри графа
n = 20  # Кількість вузлів
k = 4   # Початковий ступінь вузла
p = 0.3 # Ймовірність перепідключення замість додавання ребра


# Генерація графа за моделлю Ваттса-Строгатца
G = nx.watts_strogatz_graph(n, k, p)


# Підключення до бази даних Neo4j
url = "bolt://localhost:7687"
username = "user1"
password = "123"

graph = Graph(url, auth=(username, password))


# Збереження графа у базу даних Neo4j
for node in G.nodes():
    node_properties = {"name": f"Node_{node}", "degree": G.degree[node]}
    graph.merge_one("Node", "name", node_properties)


for edge in G.edges():
    start_node = G.nodes[edge[0]]
    end_node = G.nodes[edge[1]]
```

```
    relationship_properties = {"type": "CONNECTED"}
    graph.merge_one("Node", "name", start_node)
    graph.merge_one("Node", "name", end_node)
    graph.merge_one(start_node, "CONNECTED", end_node,
properties=relationship_properties)
```

Для читання структури графу віртуальної соціальної мережі з бази даних Neo4j розроблено наступний код:

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

from py2neo import Graph # бібліотека для роботи з базою даних Neo4j


# Параметри підключення до бази даних Neo4j
url = "bolt://localhost:7687"
username = "user1"
password = "123"


# Створення об'єкту для підключення до бази даних
graph = Graph(url, auth=(username, password))


# Запит для отримання графа з бази даних
query = """
MATCH (n:Node)-[:CONNECTED]->(m:Node)
RETURN n, m
"""


# Виконання запиту та створення графа
result = graph.run(query)
G = nx.Graph()

for record in result:
    node1 = record["n"]
    node2 = record["m"]
    G.add_node(node1["name"])
    G.add_node(node2["name"])
    G.add_edge(node1["name"], node2["name"])


# Вивід інформації про граф
print("Вершини графа:", G.nodes())
print("Ребра графа:", G.edges())

# Візуалізація графу за допомогою алгоритму Fruchterman-Reingold
pos = nx.spring_layout(G, seed=42)  # seed для відтворюваності розташування
вузлів
nx.draw(G, pos, with_labels=True, node_size=500, node_color='skyblue',
font_size=12, font_color='black', font_weight='bold', edge_color='gray',
width=2)

# Налаштування відображення
plt.axis('off')  # Вимкнення відображення координатних вісей
plt.show()
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення було використано метод шифрування даних AES. Пропонується шифрувати файли бази даних, що зберігають інформацію про аналіз віртуальних соціальних мереж.

Щоб зашифрувати файл бази даних Neo4j, спершу потрібно зробити резервну копію файлу бази даних. Потім можна використати бібліотеку PyCryptodome для шифрування та розшифрування цього файлу. Ось приклад:

1. Встановити бібліотеку PyCryptodome, якщо вона ще не встановлена. Наприклад, за допомогою pip:

```
pip install pycryptodome
```

2. Шифрування файлу бази даних. Створення ключа та ініціалізаційного вектору, які будуть використовуватися для шифрування та розшифрування файлу бази даних. Приклад коду:

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import os

# Функція для шифрування файлу
def encrypt_file(input_filename, output_filename, key):
    iv = get_random_bytes(16)  # Генеруємо ініціалізаційний вектор
    cipher = AES.new(key, AES.MODE_CBC, iv)

    with open(input_filename, 'rb') as f_input, open(output_filename, 'wb')
as f_output:
        f_output.write(iv)  # Записуємо ініціалізаційний вектор в початок
файлу

        while True:
            chunk = f_input.read(16)  # Зчитуємо дані блоками по 16 байтів
            if not chunk:
                break
            elif len(chunk) % 16 != 0:
                chunk += b' ' * (16 - len(chunk) % 16)  # Доповнюємо
останній блок до 16 байтів
            ciphertext = cipher.encrypt(chunk)
            f_output.write(ciphertext)

# Генеруємо ключ
key = get_random_bytes(16)  # 128 біт (16 байт) ключ

# Шифруємо файл бази даних
input_filename = 'база_даних.db'  # Встановіть ім'я файлу бази даних
```

```
encrypted_filename = 'зашифрована_база_даних.db'
encrypt_file(input_filename, encrypted_filename, key)
```

3. Розшифрування файлу бази даних. Для розшифрування файлу бази даних використовується той самий ключ і ініціалізаційний вектор:

```
# Функція для розшифрування файлу
def decrypt_file(input_filename, output_filename, key):
    with open(input_filename, 'rb') as f_input, open(output_filename, 'wb')
as f_output:
        iv = f_input.read(16)  # Зчитуємо ініціалізаційний вектор

        cipher = AES.new(key, AES.MODE_CBC, iv)

        while True:
            chunk = f_input.read(16)  # Зчитуємо дані блоками по 16 байтів
            if not chunk:
                break
            decrypted_chunk = cipher.decrypt(chunk)
            f_output.write(decrypted_chunk)

# Розшифровуємо файл бази даних
decrypted_filename = 'розшифрована_база_даних.db'
decrypt_file(encrypted_filename, decrypted_filename, key)
```

Важливо зберігати ключ конфіденційно, оскільки він відповідає за розшифрування файлу бази даних.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

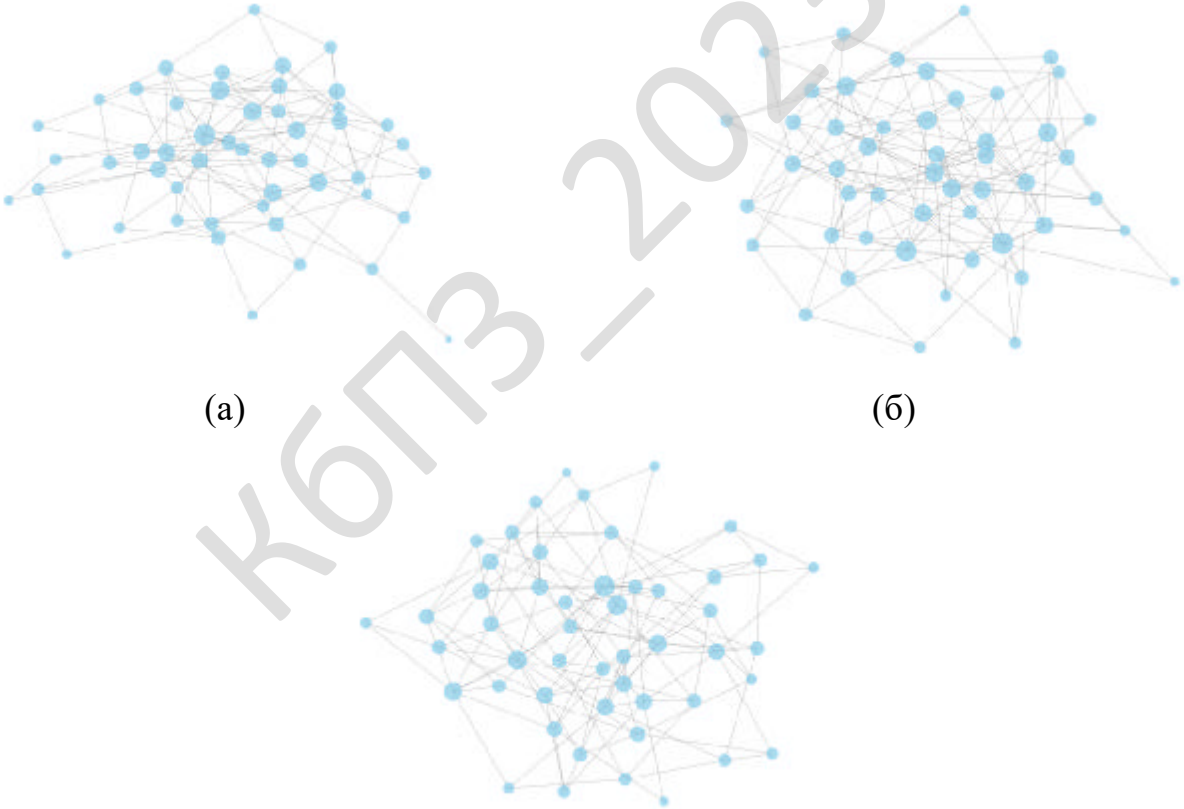
Для використання розробленого програмного забезпечення необхідно встановити наступні бібліотеки для мови Python:

– networkx – це бібліотека для візуалізації та дослідження графів і мереж.

– py2neo – це бібліотека для роботи з базою даних Neo4j.

– community – це бібліотека для визначення співтовариств у соціальній мережі.

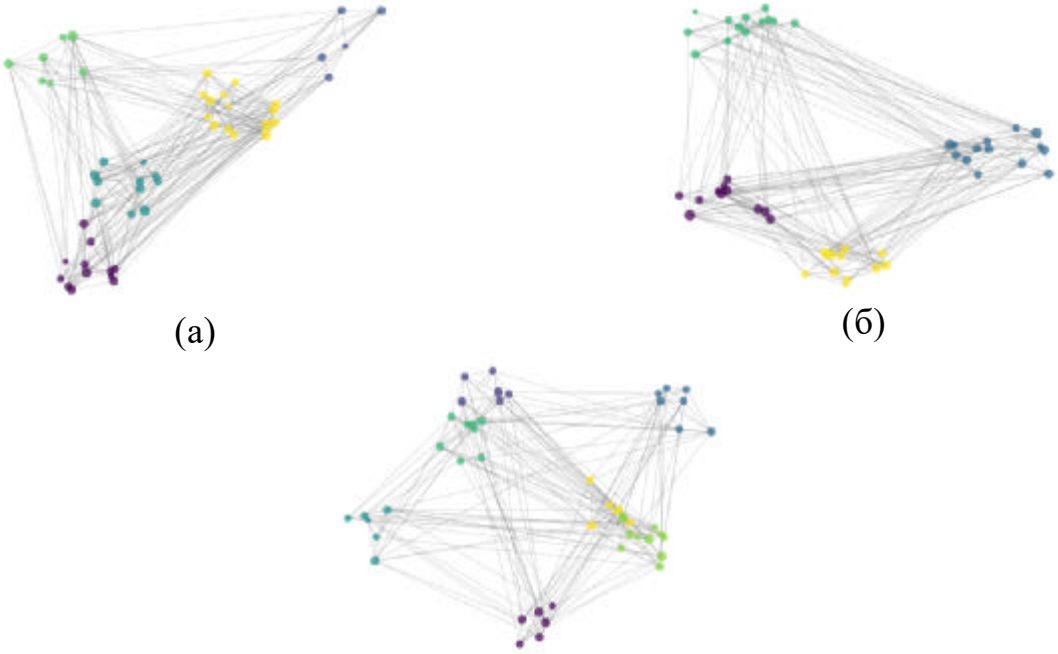
Приклади роботи розробленого програмного забезпечення наведені на наступних рис. 5.1-5.8.



(а) (б)

(в)

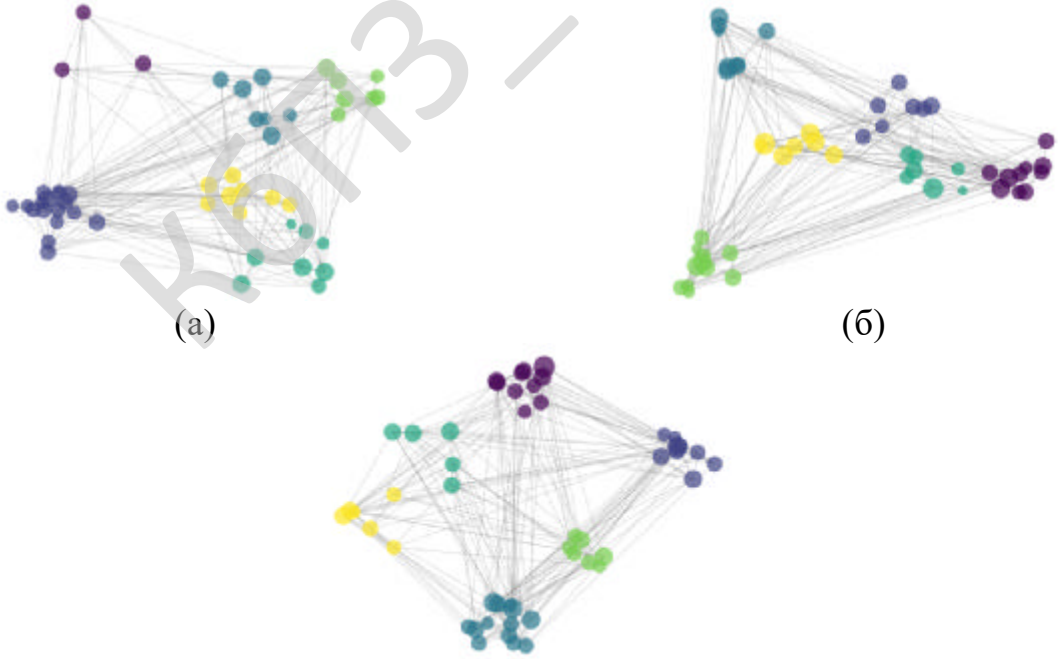
Рисунок 5.1 – Визначення центральностей вершин за степенями – чим більша центральність вершини за степенем, тим більшим колом вона зображена



(а) (б)

(в)

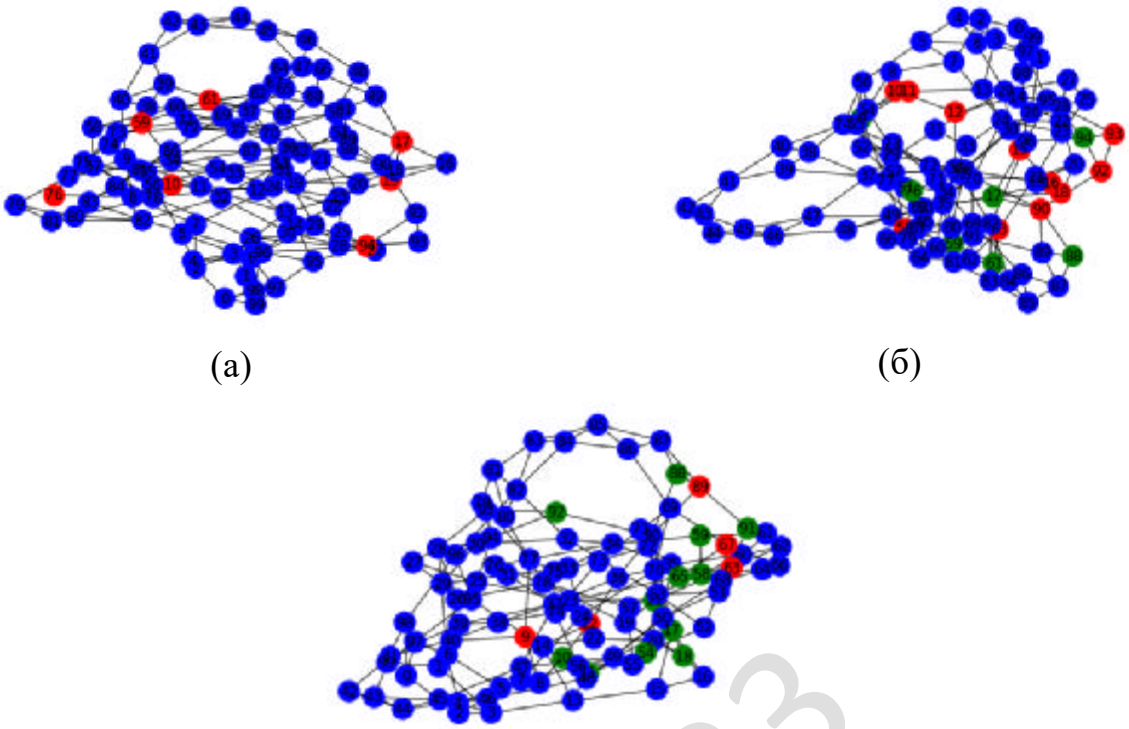
Рисунок 5.2 – Визначення співтовариств – вершини різних співтовариств розфарбовані різним кольором, вершини одного і того ж співтовариства розташовані поряд



(а) (б)

(в)

Рисунок 5.3 – Визначення співтовариств та центральностей вершин за степенями



(а) (б)

(в)

Рисунок 5.4 – Симуляція поширення вірусної інформації у віртуальній соціальній мережі на основі моделі SIRS – візуалізація мережі (сині вершини – здорові, червоні вершини – хворі, зелені вершини – з імунітетом): а) ітерація №1, б) ітерація №5, в) ітерація №10

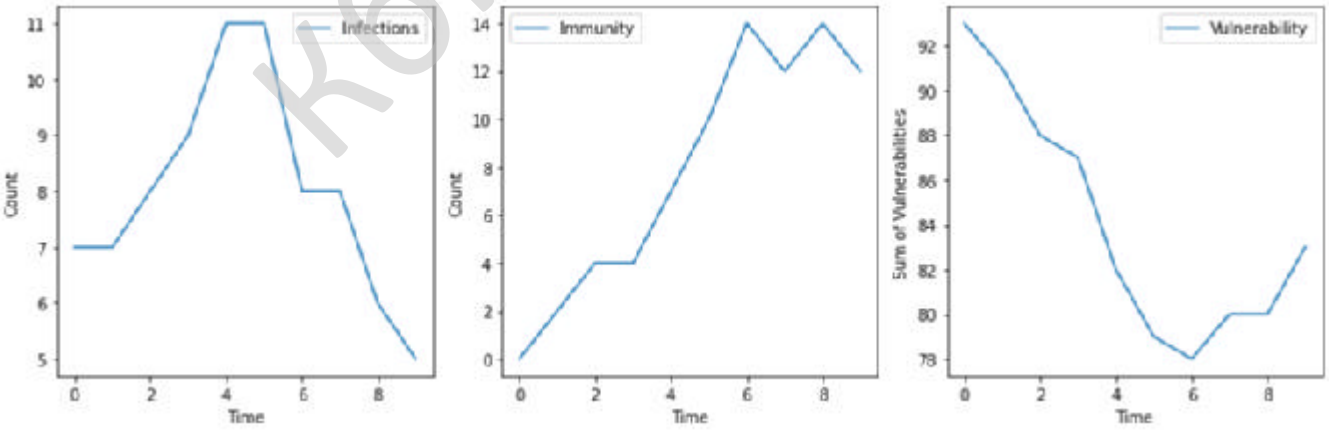


Рисунок 5.5 – Симуляція поширення вірусної інформації у віртуальній соціальній мережі на основі моделі SIRS – звіт після 10-ти перших ітерацій

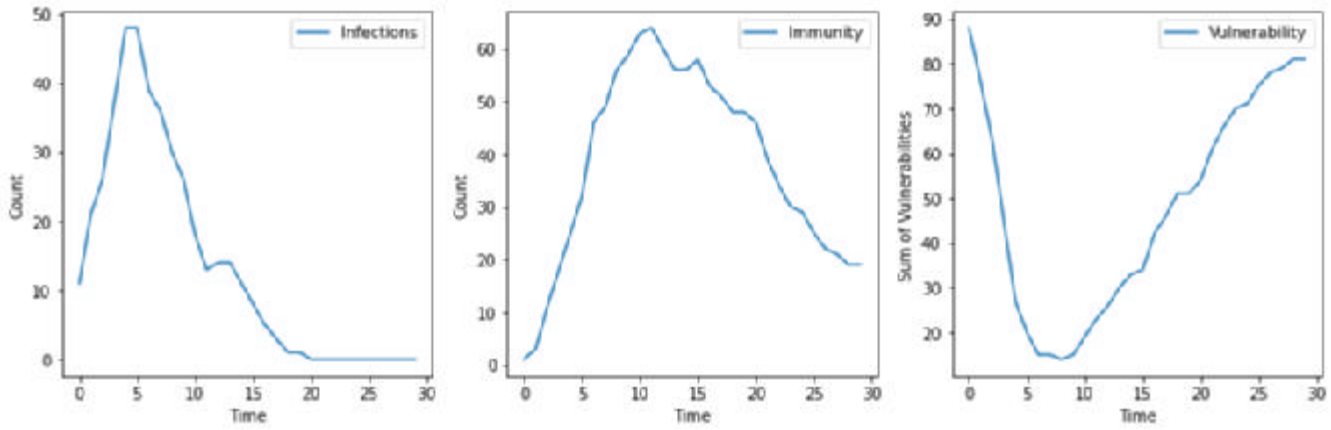


Рисунок 5.6 – Симуляція поширення вірусної інформації у віртуальній соціальній мережі на основі моделі SIRS – звіт після 30 перших ітерацій (приклад №2)

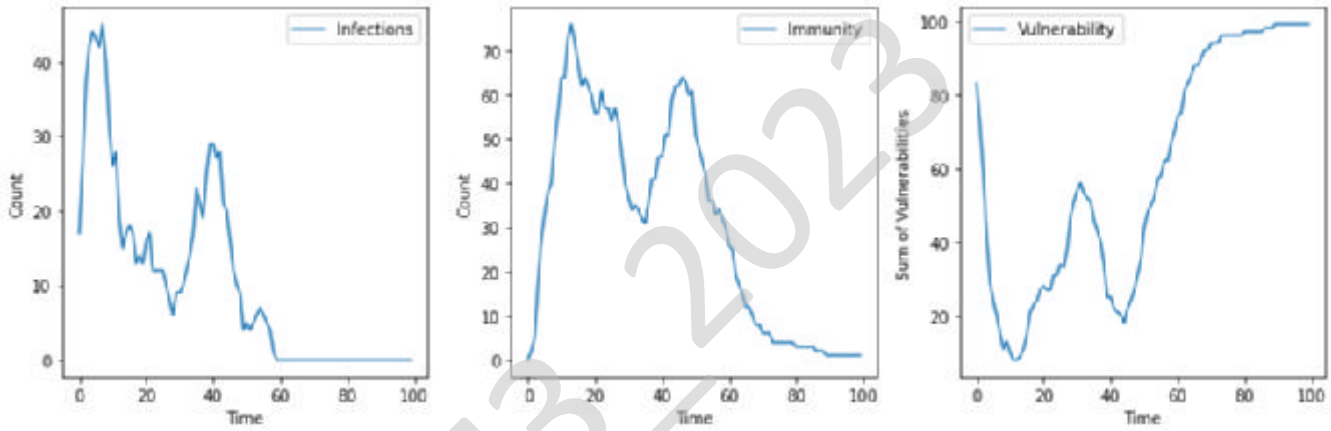


Рисунок 5.7 – Симуляція поширення вірусної інформації у віртуальній соціальній мережі на основі моделі SIRS – звіт після 100 перших ітерацій (приклад №3)

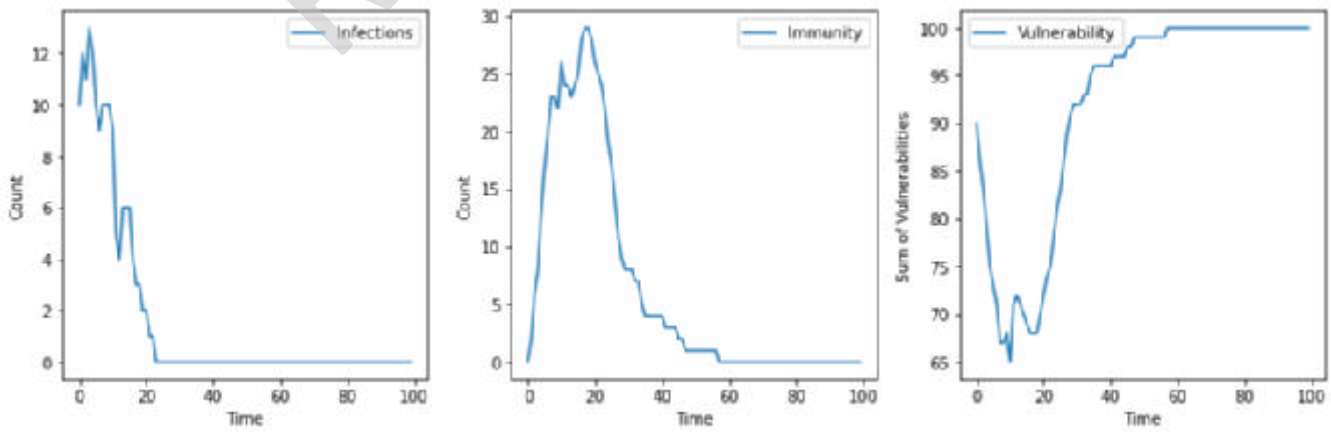


Рисунок 5.8 – Симуляція поширення вірусної інформації у віртуальній соціальній мережі на основі моделі SIRS – звіт після 100 перших ітерацій (приклад №4)

6 НАУКОВА НОВИЗНА

Метою роботи є дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж.

Об'єктом дослідження є процес автоматизованого дослідження віртуальних соціальних мереж.

Предметом дослідження є методи моделювання та аналізу віртуальних соціальних мереж.

Методи дослідження базуються на теорії графів, теорії складних мереж, теорії алгоритмів та структур даних, теорії статистики, теорії розробки програмного забезпечення та теорії імітаційного моделювання.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

1. Запропоновано метод імітаційного моделювання структури віртуальних соціальних мереж на основі моделі Воттса-Строгаца, який відрізняється від відомих симуляцією характеристик користувачів та перепідключенням ребер графу з урахуванням схожості вузлів мережі.

2. Розроблено вітчизняний продукт імітаційного моделювання та аналізу віртуальних соціальних мереж, який має більш широкі можливості, на відміну від існуючих аналогів та може бути використаний при дослідженні та прогнозуванні поведінки користувачів веб-ресурсів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моделювання та аналізу віртуальних соціальних мереж.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми магістерської роботи

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові данні

Показники		Позначення	Характеристика або величина
1		2	3
1.	Кількість розроблених програм період, шт	N	1
2.	Кількість екземплярів програм, шт	Ne	40
3.	Запланований термін розробки, днів	Fpq	60 (3 місяці)
4.	Група задачі підсистеми управління (1-6)	–	1
5.	Ступінь новизни задачі (А, Б, В, Г)	–	Б
6.	Складність алгоритму (1, 2, 3)	–	2
7.	Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	3
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження табл. 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	40000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B , \qquad (7.1)$$

де A – коефіцієнт Боема, A=2,45; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B - показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \qquad (7.2)$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПВ_j, \qquad (7.3)$$

де $ПВj$ - добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S, \qquad (7.4)$$

де С – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot2{,}85\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot42=76 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	76	Ф 7.1-7.4
Впровадження	13	Д13
Всього	117	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{117 \cdot 1}{60\text{-}5} = 2{,}1 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	5	450	7,5
Монітор	60	5	300	5
Клавіатура	30	5	150	2,5
Маніпулятор «мишка»	30	5	150	2,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор–маршрутизатор	30	1	30	0,5
Кабельні господарства ЛВС на 1 м. п.	2,5	180	450	7,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	31,16

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2} \quad .6)$$

$$Ф_{др}^{с} = \frac{31,16 \cdot 3}{1,2} = 78 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=78/(60·8)=0,15 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів – електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, серверу доступу АДСЛ (ОС Linux), Wi-Fi налаштування ADSL, VPN, PPPoE, Frame Relay	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців:

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (IT-менеджер)	0,5	18200	27300
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	2,1	16000	100800
Інженер-електронщик	0,15	12000	5400
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	12000	9000
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	12000	9000
Бухгалтер-економіст	0,5	12000	18000
Всього за період розробки	$R_{сп}$=5	-	$Ф_{роб}$=214500

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \quad (7.8)$$

де $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{214500}{5 \cdot 60} = 715 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць. Sу – питома площа на одне робоче місце, $м^2$; Цпл – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає $500…1600$ $у.о./м^2$.

Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно $8\,м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались за прайсом фірми Компбест за 03.11.23 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	INTEL Pentium G6405 (BX80701G6405) 1200, 2 ядра, 4 потоки, 4.1 GHz, TDP - 58 Вт	-
Системна плата	GIGABYTE H470M H сокет - 1200, DDR4, - 3200 MHz, LAN - 1 Гбит/с, D-Sub (VGA), HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	вбудована	-
Жорсткий диск	SSD 2.5" 256GB Mibrand (MI2.5SSD/CA256GBST) 256 GB, 3D TLC NAND, 2.5", SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 2666 MHz eXceleram (E408266A)	-
DVD-привод	-	-
Корпус	Vinga CS105B-450W Класіка, Miditower, ATX, Micro - ATX, Mini - ITX, PSU - 450 Вт	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	-
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	LG W2363V-WF Wide LCD 2ms, 70 000:1, 300кд/м2, 170/160, D-Sub / Glossy White	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	40000	10	4000
Разом	$К_р$= 1596075		$А_р$= 140140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де Ne – Кількість екземплярів програм, шт.

$$З_о = 715 \cdot 117/40 = 2090 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 2090 \cdot 10 \cdot 0{,}01 = 209 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(2090+209) = 506 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом Нг=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 2090 \cdot 15 \cdot 0{,}01 = 314 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 210 \cdot 0{,}5 = 105 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 10):

$$З_{М2} = \sum Ц_д \,, \tag{7.17}$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 33,6 грн./шт., DVD-R box – 49,2 грн./шт.

$$З_{М2} = 49{,}2 \cdot 10 = 492 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \tag{7.18}$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570=1702 \text{ грн.}$$

$$З_М = (105 + 492 + 1702)/40=57 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом (Нп = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \tag{7.19}$$

де Нп - норматив витрат на освоєння нових мов програмування, %.

$$О_п = 2090 \cdot 15 \cdot 0{,}01=314 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 40 прим.).

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \tag{7.20}$$

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 140140 \cdot 3/(40 \cdot 12)=876 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м . \quad (7.21)$$

$$C_п = 2090+209+506+314+57+314+876 = 4366 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 55%.

$$П_р = 0{,}01 \cdot Р_п \cdot С_п , \quad (7.22)$$

де Рс – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 55 \cdot 4366 = 2401 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позна-чення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	2090
2. Додаткова зарплата виконавців	$З_д$	209
3. Відрахування на соціальні потреби	$С_{оц}$	506
4. Загальногосподарські витрати	$Г_{осп}$	314
5. Витрати на матеріали	$З_М$	57
6. Освоєння нових операційних систем, мов програмування	$О_п$	314
7. Амортизація основних фондів	$А_м$	876
8. Повна собівартість програмного забезпечення	$С_п$	4366
9. Плановий прибуток	$П_р$	2401
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	6767
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1353,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	9168

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.9.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	9168
Всього капітальних витрат	–	9168

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на оплату праці	$З_р$	96624	24156
2. Витрати на електроенергію	$З_{ел}$	2099	1049
3. Витрати на амортизацію	$З_{ам}$	0	2292
Всього витрат за рік	$І$	98723	27497

Витрати на оплату праці:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \quad (7.23)$$

де Тр – кількість годин обслуговування системи за рік, *год.*; Зг – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 600 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 600 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 96624\ грн.$$

до:

$$З_{р\,нов} = 150 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 24156\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності (Пел) в кіловатах, часу експлуатації технічних засобів (Тр) в годинах та ціни однієї кіловат-години (Цел).

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Зел баз= $\cdot 0{,}475 \cdot 2455 \cdot 1{,}8 = 2099$ грн.

Зел нов= $\cdot 0{,}475 \cdot 1227 \cdot 1{,}8 = 1049$ грн.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	9168	–	2292
Всього відрахувань	-	–	9168	–	2292

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (6767 - 4366) \cdot 40 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 40000) \cdot 3/12 = 61005 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{188075}{(6767 - 4366) \cdot 40 \cdot 12 / 3} = 0,5 \text{ років}.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (98723 - 27497) - 0,25 \cdot 9168 = 68934 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{9168}{98723 - 27497} = 0{,}13 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	40
2. Повна собівартість розробленої програми	Грн	4366
3. Ціна розробленої програми	Грн.	6767
4. Плановий прибуток від реалізації розробленої програми	Грн.	2401
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1596075
7. Загальний прибуток від реалізації програмної продукції	Грн.	96040
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	61005
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Рік	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	9168
11.Величина економічного ефекту у користувача програмної продукції	Грн.	68934
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	0,13

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Характерною ознакою сучасного науково-технічного прогресу практично у всіх сферах діяльності людини є широке застосування комп'ютерних технологій, заснованих на використанні електронно-обчислювальних машин (ЕОМ). Сьогодення, а тим більше, майбутнє, вже важко уявити без комп'ютерів та іншої електронної техніки. Адже саме завдяки їм стала можливою швидка переробка величезних обсягів інформації, проведення необхідних розрахунків, виконання різних видів робіт, пов'язаних обробкою текстових та ілюстраційних зображень, організація оперативного отримання та передачі інформації, збереження її значних обсягів електронним способом.

Стрімке впровадження комп'ютерів не тільки в сфері управління виробництвом, в банківській систему, бізнесі, системі освіти, але також на транспорті, сфері обслуговування призвело до того, що десятки мільйонів людей у всьому світі виявились втягнутими у взаємодію людини з комп'ютером. Природно виникає запитання: настільки безпечною є ця взаємодія для людини? Адже відома аксіома про те, що будь-яка взаємодія людини та засобів праці двостороння.

Людина впливає на удосконалення засобів праці, а останні –на працюючу людину. Отже, навіть сучасні технології та техніка, до яких безперечно, залежать комп'ютерні технології та ЕОМ несуть у собі певні потенційні небезпеки. У зв'язку з цим набуває актуальності адекватна оцінка конкретних умов і характеру праці, яка сприяє обґрунтованому розробленню та впровадженню комплексу заходів і засобів, спрямованих на збереження здоров’я і працездатності людини в процесі праці за рахунок поліпшення параметрів виробничого середовища, зменшення важкості, напруженості трудового процесу та збереження здоров'я працівників на комп'ютеризованих робочих місцях.

Законодавством України чітко врегульовано норми та вимоги до використання комп’ютерної техніки на підприємстві, безпосередньо й охорона праці на підприємстві при роботі за комп’ютером., зокрема «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», затверджені наказом Мінсоцполітики від 14.02.2018 № 207 [47], «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98. [48].

Загальні вимоги пожежної безпеки під час експлуатації комп’ютерної техніки визначають «Правила пожежної безпеки в Україні» (затверджені наказом МВС від 30.12.2014 № 1417) [49], комп’ютерних класів — пункт 3 розділу VIII «Правил пожежної безпеки для навчальних закладів та установ системи освіти України» (затверджені наказом МОН від 15.08.2016 № 974). [50] та інші державні стандарти, що регламентують експлуатування комп’ютерної техніки як радіоелектронної апаратури.

8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером

Можна виділити наступні основні фактори, що впливають на стан здоров'я людей, які працюють за комп'ютером:

- сидяче положення на протязі тривалого періоду;
- вплив електромагнітного випромінювання монітора;
- втома очей, навантаження на зір;
- перевантаження суглобів кистей;
- стрес при втраті інформації.

У кожному з цих випадків ступінь ризику прямо пропорційний часу, що проводиться за комп’ютером і поблизу нього.В сучасних умовах взаємодія людини з технікою значно ускладнилась, що вимагає комплексного підходу, який передбачає розгляд людини, технічних засобів праці та виробничого середовища, як взаємозв'язаних елементів єдиної системи. Все вищесказане в повній мірі

відноситься й до системи «людина–комп'ютер–середовище».

Вагомий вплив на працездатність та здоров'я користувачів комп'ютерів здійснює виробниче середовище. Це середовище у виробничих приміщеннях (офісах), в основному, визначається мікрокліматом, освітленням, наявністю шкідливих речовин у повітрі, рівнем шуму, випромінювання.

Для того, щоб об'єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам та запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини необхідно здійснити санітарно-гігієнічну характеристику умов працівника, який працює з програмним продуктом.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці користувача ПК

Розглянемо приміщення в якому працює користувач ПК з даним програмним продуктом.

Приміщення має одностороннє природне освітлення і загальне штучне освітлення. Стіни і стеля обклеєні світлими шпалерами, підлога вкрита темним ламінатом. У приміщенні відсутні сильні вібрації та шкідливі речовини. Склад повітря в нормі. У кімнаті знаходиться ПК з 4-ядерним процесором і 23-дюймовим IPS монітором, а також меблі.

Приміщення має довжину 4м, ширину 3,5м, висоту стелі 2,7м. Кількість робочих місць–одне. Площа–14м^2, об'єм–37,8м^3. Виходячи з цього, отримано дані, наведені в таблиці 8.1.

Таблиця 8.1 – Фактичні та нормативні значення параметрів приміщення

Параметр	Норма *	Реальні параметри
Площа, S	не менше 6 м^2	14 м^2
Об'єм, V	не менше 20 м^3	37,8 м^3

*Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

За даними, які наведено у табл. 8.1, можна зробити висновок, що отримані показники, площа та об’єм приміщення у розрахунку на одно робоче місце користувача ПК відповідає чинним нормам і вимогам.

Щодо мікроклімату, то згідно з ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [51] роботу з ПК можна віднести до категорії легка 1а. Джерелами тепла в цьому приміщенні є люди, електроустаткування, освітлювальні прилади в темний час доби і система опалювання взимку. Оператором виділяється до 120ккал теплової енергії за годину. Оптимальні та фактичні значення параметрів мікроклімату приведені в таблиці 8.2.

Таблиця 8.2 – Значення параметрів мікроклімату

Період року	Параметр	Оптимальний*	Фактичний
Теплий	Температура	23 – 25 ^{0} С	24^0 С
	Вологість	40 – 60%	50%
	Швидкість повітря	< 0,1м/с	
Холодний	Температура	22 - 24^0 С	23^0 С
	Вологість	40 – 60%	55%
	Швидкість повітря	< 0,1м/с	

*ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень»

По отриманим замірам параметрів мікроклімату можна зробити висновок, що усі показники задовольняють вимогам зазначеним для робіт категорії легка 1а і є задовільними для здоров'я людини.

Щодо освітлення, то згідно з ДБН В.2.5-28:2006 «Природне і штучне освітлення» [52] ця робота відноситься до Va розряду зорових робіт. Передбачається використання природного, штучного і змішаного освітлення.

Природне освітлення здійснюється за допомогою вікна, площа якого складає S' = 1,8*1,5 = 2,7м^2 і є боковим освітленням. У світильниках місцевого і загального освітлення використовуються світлодіодні лампи потужністю 20Вт із світловим потоком лампи 900 лм. Згідно замірів рівень освітлення в даному приміщенні і на робочому місці складає в межах 350 -500 лк, що відповідає

нормованому значенню

Джерелом шуму в приміщенні є комп'ютер. Вентилятори (кулери) системного блоку, процесора, відеокарти і блоку живлення є сучасними і мають низький рівень шуму. Згідно з технічною документацією шум, зумовлений кулером в блоці живлення складає 25 дБ, кулером процесора - 30 дБ, загальний -34 дБ. Враховуючи незначний рівень шуму від персонального комп'ютера і незначний рівень фонового шуму від іншого устаткування, можна стверджувати, що сумарний рівень шумового забруднення приміщення не перевищує максимально допустимий рівень коригованої звукової потужності і складає не більше 50 дБА, що відповідає рівню шуму для приміщень з комп'ютерною технікою згідно Державних санітарних правил і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

У приміщенні відсутні джерела інфрачервоного, ультрафіолетового і електромагнітного випромінювання, бо монітор ПК вироблений на основі рідкокристалічної матриці, підсвітка якої здійснюється неоновою лампою, що не має сильного електромагнітного випромінювання і сертифіковані в Україні.

Блок живлення є екранованим і не випускає вищезазначених видів випромінювання.

8.4 Розробка заходів з умов поліпшення охорони праці

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці користувача ПК.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: устаткування розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату, рівня звуку і освітленості до цих заходів можна віднести: організацію природної вентиляції, за

допомогою дефлектора, для забезпечення необхідного повітрообміну в приміщенні вузла; організацію системи центрального опалювання, для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення, для забезпечення необхідних умов зорової роботи, що відповідають, оформлення паспорта на приміщення вузла, з занесенням в нього вимірювань освітленості і рівня звуку, проведених відділом охорони праці.

Крім рекомендацій щодо конкретного приміщення, де було проведено дослідження умов праці, існують загальні вимоги, які зарекомендовані відповідними нормативними документами.

Правильна організація робочих місць запобігає передчасній втомлюваності користувача і сприяє збереженню здоров'я. Організація робочого місця передбачає:

- правильне розміщення робочого місця у виробничому приміщенні;
- вибір ергономічного обґрунтованого робочого положення, виробничих меблів з урахуванням характеристик людини;.
- раціональне компонування обладнання на робочих місцях;
- урахування характеру й особливостей трудової діяльності. Стосовно робочих місць користувача ВДТ, то організація робочого має забезпечуватися відповідно до ДСанПіН 3.3.2-007-98. Для запобігання перевтомленню необхідно виконувати вправи для очей та дотримуватись розпорядку роботи та відпочинку. На робочому місці реалізовувався режим відпочинку: кожні дві години – перерва для виконання фізичних вправ для м‘язів очей.

8.5 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток *63·63·6 мм*., (згідно з ДСТУ 2251-93 «Кутики сталеві гарячекатані рівнополичні. Сортамент») довжиною *L=2 м.,* та горизонтальний електрод – металева полоса з перетином *60·5 мм*. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проведем за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта — чорнозем, нижнього шару грунта — глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: $H=0,5$ *м*. Відстань між вертикальними заземлювачами (електродами) $A=2$ *м*. Глибина закладення горизонтального контура заземлення $t=0,6$ *м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та довжину полоси (горизонтального заземлювача).

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,6+2/2=1,6 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi\rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де $\psi = 1,36$ - табличне значення коефіцієнта сезонності для відповідної кліматичної зони у багатошаровому грунті [58];

$\rho_2 = 40$ *Ом·м*. - табличне значення питомого опору нижнього шару грунта (глина) [58].

Еквівалентний діаметр вертикального електрода (кутка) [58]:

$$Dв=0,95 \cdot K = 0,95 \cdot 63 = 59,85 \text{ мм.} = 0,0598 \text{ м.}$$

де $K = 63$ *мм.* – розмір металевого кутка (задан).

Відношення $A/L=2/2=1$

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [58]:

$$R_O=0,366(\rho/L)[\lg(2L/Dв)+(1/2)\lg((4T+L)/(4T-L))] =$$

$$=0,366(54,5/2)[(\lg(2 \cdot 2/0,0598)+(1/2)\lg((4 \cdot 1,6+2)/(4 \cdot 1,6-2))] =$$

$$=19,6 \text{ Ом.}$$

Визначаємо коефіцієнт екранування ветикальньних електродів $Кев=0,62$

при орієтовній кількості ветикальньних електродів, яке дорівнює 5 [58].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без вхарування горизонтального заземлювача), при $R_{ЗН} = 4$ *Ом*:

$$N=R_O /(Кев\ R_{ЗН})= 19,6 / (0,62\cdot4)= 7,8 \approx 8\ шт.$$

Визначаємо длину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot2\cdot8=16,6\approx 16\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [58]:

$$R_П =0,366(\rho\cdot K_П/L_П)lg(2\cdot L_П^{\,2}/(B\cdot t))=$$

$$=0,366(40\cdot5/16)\cdot lg((2\cdot16^{\,2})/(0,06\cdot0,6))=20,2\ Ом.$$

де $K_П=5$ - табличне значення кліматичного коефіцієнта питомого опору грунта для відповідной климатичной зони для з'єднуючої полоси [58]:

$B = 60\ мм. = 0,06\ м.$- ширина з'єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [58]:

$$R =(R_O\cdot R_П)/(R_O\cdot\eta_П+ N\cdot R_П\cdot Кев)=$$

$$=(19,6\cdot 20,2)/(19,6\cdot 0,6+ 8\cdot 20,2\cdot 0,62)=3,53\ Ом.$$

де $\eta_П = 0,6$ - табличне значення коефіцієнта екранування з'єднуючої полоси [58].

Умова $R \le R_{ЗН}$ виконується ($3,53 \le 4$).

Остаточно отримали: кількість вертикальних електродів дорівнює *8* при $R = 3,53$ *Ом.*

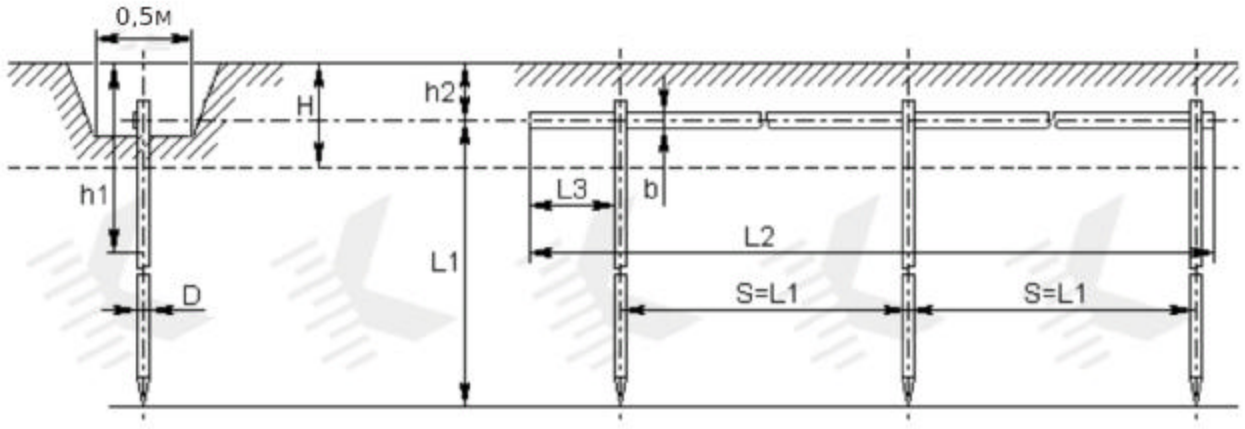


Рисунок 8.1 – Схема штучного заземлення

Висновки

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання магістерської роботи, призначено для системи моделювання та аналізу віртуальних соціальних мереж.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У магістерській роботі наведені теоретичне узагальнення й рішення наукового завдання дослідження методів моделювання та аналізу віртуальних соціальних мереж.

Рішення даного завдання полягало у вирішенні наступних задач:

– Дослідження існуючих систем для моделювання та аналізу віртуальних соціальних мереж.

– Розробка методів та алгоритмів моделювання та аналізу віртуальних соціальних мереж.

– Програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж.

Розроблені під час виконання магістерської роботи алгоритми дозволяють успішно вирішувати завдання моделювання та аналізу віртуальних соціальних мереж.

Проведено аналіз предметної галузі в ході якого були виявлені об’єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Програма реалізована на мові високого рівня Python у середовищі розробки Spyder. Дана мова програмування дозволяє найбільш ефективно обробляти дані соціальних мереж. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як наслідок, зменшити витрати на його розробку.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для захисту програмного забезпечення та даних використано метод шифрування даних AES.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 68934 грн. З урахуванням вартості розробки програми та обладнання, строк окупності становить 0,13 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Савчук Б.В. Підходи до аналізу соціальних мереж. Тези Всеукраїнської науково-практичної on-line конференції аспірантів, молодих учених та студентів, присвяченої Дню науки. Державний університет «Житомирська політехніка». 2017 – URI: http://eztuir.ztu.edu.ua/123456789/6684

2. Міхав В.В. Програмне забезпечення для моделювання мереж репутації користувачів соціальних веб-ресурсів на основі бінарних діаграм рішень // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2019. – Т. 5 (57). – С. 78-83. – doi: https://doi.org/10.26906/SUNZ.2019.5.078

3. Улічев О.С. Дослідження моделей розповсюдження інформації та інформаційних впливів в соціальних мережах // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2018. – Т. 4 (50). – С. 147-151. – doi: https://doi.org/10.26906/SUNZ.2018.4.147

4. Chen D., Giulio C., Linyuan L., Medo M., Zhang Y.-C., Zhou T. Enhancing topology adaptation in information-sharing social networks // Physical Review E. 2012. Vol. 85. № 4.

5. Newman M. E. J. The structure and function of complex networks / Department of Physics, University of Michigan, Ann Arbor, MI 48109, U.S.A. and Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, U.S.A.

6. Velz S. V. Modelirovanie informazionnogo protivoborstva v sozialnih setiah na osnove teorii igr [Modeling information warfare in social networks based on game theory and dynamic Bayesian networks] // Engineering journal: science and innovation. Publ., 2013. No. 11(23), pp. 39.

7. Holger Ebel, Lutz-Ingo Mielsch, Bornholdt S. Scale-free topology of e-mail networks / Institut f¨ur Theoretische Physik, Universit¨at Kiel, Leibnizstraße 15, D-24098 Kiel, Germany.

8. Мелешко Є. В., Гермак В. С., Охотний С. М. Дослідження методів

визначення центральності акторів у соціальних мережах для задач інформаційної безпеки // Системи управління, навігації та зв'язку. - 2016. - Вип. 4. - С. 67-70. - Режим доступу: http://nbuv.gov.ua/UJRN/suntz_2016_4_18

9. Davern M. Social Networks and Economic Sociolo-gy: A Proposed Research Agenda for a More Complete So-cial Science / M. Davern // The American Journal of Eco-nomics and Sociology. – 1997. – Vol. 56, No. 3. – P. 287-302.

10. Hanneman R. A. Introduction to Social Network Methods (free introductory textbook on social network analysis). / R. A. Hanneman, M.D. Riddle – 2005. [Електронний ресурс] Режим доступу: http://faculty.ucr.edu/~hanneman/

11. Bonacich P. Power and Centrality: A Family of Measures / P.Bonacich //American Journal of Sociology. – 1987. – Vol. 92, No. 5. – P. 1170-1182.

12. Жулькевська О.В. Специфіка застосування мережевого аналізу в соціології: дис. канд. соціол. наук: 22.00.02 / Жулькевська Олена Володимирівна. – Київ, 2003. – 240 с.

13. NetworkX Tutorial. This guide can help you start working with NetworkX. – URL:https://networkx.org/documentation/stable/tutorial.html

14. Knuth, Donald E. The Art of Computer Programming: Fundamental Algorithms. 3rd Ed. Addison-Wesley, 1997.

15. Knuth, Donald E. The Art of Computer Programming: Seminumerical Algorithms. 3rd Ed. Addison-Wesley, 1998.

16. Knuth, Donald E. The Art of Computer Programming: Sorting and Searching. 2nd Ed. Addison-Wesley, 1998.

17. Алгоритми і структура даних: Навчальний посібник / В.М.Ткачук. - ІваноФранківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. 286 с.

18. Алгоритми та структури даних. Навчальний посібник / Т. О. Коротєєва. Львів : Видавництво Львівської політехніки, 2014. - 280 с.

19. Мелешко Є. В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних // Навчальний посібник для студенів технічних спеціальностей денної та

заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с. – URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/8944

20. Смірнов О.А., Коваленко О.В., Мелешко Є.В., Константинова Л.В., Кожанова А.С. Інженерія програмного забезпечення // Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 6.050102 «Комп'ютерна інженерія». Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 18.03.13 року №1/11-5584 – Кіровоград: КНТУ 2013. – 335 с.

21. Albert R., Barabási A.-L. Statistical mechanics of complex networks (англ.) // Reviews of Modern Physics, Vol. 74. – 2002. – P. 47-97. – [Electronic resource] – Access mode: doi:10.1103/RevModPhys.74.47

22. Barabási A.-L. Network science // Cambridge University Press. – 2018. – 475 p. – [Electronic resource] – Access mode: http://networksciencebook.com/

23. Barabási A.-L., Albert R. Emergence of scaling in random networks (англ.) // Science, Vol. 286, No. 5439. – 1999. – P. 509-512. – [Electronic resource] – Access mode: https://doi.org/10.1126/science.286.5439.509

24. Barabási L.-A., Albert R., Jeong H. Diameter of the world-wide web // Nature, Vol. 401. – 1999. – P. 130-131.

25. Barabási L.-A., Albert R., Jeong H. Scale-free characteristics of random networks: the topology of the world-wide web // Physica, A281. – 2000. – 69-77.

26. Meleshko Ye. Computer model of virtual social network with recommendation system // Innovative technologies and scientific solutions for industries. – 2019. – №2(8). – P. 80-85.

27. Neo4j Documentation // Official website of the graph database Neo4j. – [Electronic resource] – Access mode: https://neo4j.com/docs/

28. Newman M., Barabási A.-L., Watts D. J. The Structure and dynamics of networks // Princeton University Press. – 2006. – P. 592.

29. Traag V.A. Algorithms and Dynamical Models for Communities and Reputation in Social Networks // Springer International Publishing. – 2014. – P. 229. – [Electronic resource] – Access mode: https://doi.org/10.1007/978-3-319-06391-1

30. Ulichev O., Meleshko Y., Khokh V. The computer simulation method of a social network structure for the research of dissemination processes of informational influences // Scientific and Practical Cyber Security Journal (SPCSJ) 4(3). – Georgia, Tbilisi, 2019. – P. 34-47.

31. Watts D.J., Strogatz S.H. Collective dynamics of "small-world" networks // Nature, Vol. 393(6684). – 1998. – P. 440-442. – [Electronic resource] – Access mode: https://www.nature.com/articles/30918

32. Улічев О.С., Мелешко Є.В. Математична модель розповсюдження інформації в сегменті соціальної мережі // Матеріали XX Міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування», м. Кропивницький, 13-14 квітня 2018 р. – Кропивницький: КЛА НАУ. – 2018. – С. 68-72.

33. Пасічник В. В., Іванущак Н. М. Дослідження та моделювання складних мереж. Східно-Європейський журнал передових технологій. 2010. Вип. 2, № 3 (44). С. 43–48.

34. Lutz M. Learning Python, 5th Edition Fifth Edition. - O'Reilly Media, 2016. - 1643 p.

35. Lutz M. Python: Pocket Reference Fourth Edition. - O'Reilly Media, 2016. - 210 p.

36. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. – Pearson, 2001. – 620 p.13. Кормен Т., Лейзерсон Ч., Риверст Р., Штайн К. Алгоритмы: построение и анализ, 3-е издание – М.: Диалектика, 2019. – 1328 p.

37. Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology 1st Edition. – Cambridge University Press, 2008. – 556 p.

38. The Python Tutorial. –https://docs.python.org/3/tutorial/index.html

39. Lambert K. A. Fundamentals of Python: First Programs, 2nd Edition. – Cengage, 2019.

40. NumPy. – URL: http://numpy.org .

41. Matplotlib. – URL: http://matplotlib.org .

42. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd Edition. – O'Reilly Media, 2018.

43. Wentworth P., Elkner J., Downey A., Meyers C. How to Think Like a Computer Scientist: Learning with Python 3. – Green Tea Press, 2018.

44. Боднарчук Ю.В., Олійник Б.В. Основи дискретної математики: Навч. посіб. – К. : Вид. дім дім "Києво-Могилянська Академія", 2009.

45. Основи дискретної математики. Частина 2. Математична логіка. Теорія графів : Навчальний посібник / В.С. Ільків, П.І. Каленюк, І.В. Когут, З.М. Нитребич, П.Я. Пукач, П.Л. Сохан, Р.Р. Столярчук, У.Б. Ярка. Львів: Видавництво Львівської політехніки, 2011. 184 с.

46. Meier A., Kaufmann M. SQL & NoSQL Databases, Springer Vieweg, Wiesbaden. 2019. P. 201-218.

47. Державні будівельні норми України: ДБН В.2.5-28:2018. - Режим доступу до ресурсу: https://goo.su/9AkQ (дата звернення 19.10.22).

48. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення 19.10.22).

49. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення 19.10.22).

50. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

51. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508 (дата звернення 19.10.22).

52. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення 19.09.22).

53. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. - 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

54. Оришака О.В. Охорона праці в галузі та цивільний захист / О.В Оришака, Г.П. Горбачова, О.М. Мезенцева , К.М. Марченко, К.О. Буравченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький: Видавець Лисенко В.Ф., 2019. – 226 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/9258 (дата звернення 19.09.22).

55. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. – 19 с. [Електронний ресурс]. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12240 (дата звернення 19.09.22).

56. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд. ; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. - Кіровоград:

КІСМ, 1997. - 20 с. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358 (дата звернення 19.09.22).

57. Охорона праці. Ч. 2. Занулення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ−сумісного типу / [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Е. Солових, С. Е. Катеринич] ; Мін-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - 2−ге вид., перероб. та доп. - Кропивницький : ЦНТУ, 2019. - 27 с. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/8769 (дата звернення 19.09.22).

58. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи моделювання та аналізу віртуальних соціальних мереж.

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. №__-__ від __.__.20__ року).

3 Мета та призначення розробки

Метою магістерської роботи є дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж.

4 Джерела розробки

Джерелом цієї магістерської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки

програмного забезпечення;

– аналіз умов праці розробників програмного забезпечення;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– моделювання та аналізу віртуальних соціальних мереж;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Застосунок, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Мова програмування високого рівня Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД.

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 1 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи повинні бути розглянуті умови праці програмістів під час розробки програмного забезпечення.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуші.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 94 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі магістерської роботи. Постановка задачі на виконання магістерської роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень магістерської роботи.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання магістерської роботи на попередній захист __.__.2023 р.

11.2 Подання магістерської роботи на захист __.__.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Є.В. Мелешко

Дослідження та програмна реалізація системи моделювання та аналізу віртуальних соціальних мереж

Лістинг програми

Код документу 12

Носій: DVD-диск

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2023 року

Файл modelNetwork.py – генерація структури віртуальної соціальної мережі

```
import networkx as nx
import random

import numpy as np
import matplotlib.pyplot as plt

import community  # Бібліотека для використання алгоритму Лувейна


# Параметри графа
n = 70  # Кількість користувачів (вузлів)
k = 4   # Початковий ступінь вузла
p = 0.7 # Ймовірність перепідключення замість додавання ребра


# Генерація рис користувачів (10 випадкових чисел для кожного)
user_attributes = {i: np.random.rand(10) for i in range(n)}


# Функція для обчислення коефіцієнта кореляції Пірсона між рисами користувачів
def compute_similarity(user1, user2):
    return np.corrcoef(user1, user2)[0, 1]


# Створення початкового регулярного графа
G = nx.Graph()
for i in range(n):
    for j in range(1, k // 2 + 1):
        G.add_edge(i, (i + j) % n)


# Перепідключення ребер з урахуванням схожості користувачів
for i in range(n):
    for j in range(1, k // 2 + 1):
        if random.random() < p:
            new_neighbor = random.randint(0, n - 1)
            while new_neighbor == i or G.has_edge(i, new_neighbor):
                new_neighbor = random.randint(0, n - 1)
            similarity = compute_similarity(user_attributes[i],
user_attributes[new_neighbor])
            connection_probability = similarity
            if random.random() < connection_probability:
                G.remove_edge(i, (i + j) % n)
                G.add_edge(i, new_neighbor)


# Знаходження співтовариств у графі за допомогою алгоритму Лувейна
partition = community.best_partition(G)


# Визначення позицій вершин з одного співтовариства поруч, але не ідентичними
pos = nx.spring_layout(G, seed=42)
for comm in set(partition.values()):
    nodes_in_comm = [node for node, comm_id in partition.items() if comm_id ==
comm]
    x_positions = [pos[node][0] for node in nodes_in_comm]
    y_positions = [pos[node][1] for node in nodes_in_comm]
    avg_x = sum(x_positions) / len(x_positions)
    avg_y = sum(y_positions) / len(y_positions)
    for node in nodes_in_comm:
        pos[node] = (avg_x + random.uniform(-0.1, 0.1), avg_y + random.uniform(-
0.1, 0.1))
```

```
plt.figure(figsize=(8, 6))
cmap = plt.get_cmap("viridis", max(partition.values()) + 1)
node_colors = [cmap(partition[node]) for node in G.nodes()]
node_sizes = [10 * G.degree[node] for node in G.nodes()]


# Візуалізація графа з вершинами з одного співтовариства поруч
nx.draw(G, pos, with_labels=False, node_size=node_sizes, node_color=node_colors,
edge_color='gray', width=0.5, alpha=0.7)
plt.show()


# Виведення співтовариств
print("Співтовариства:")
for comm, nodes in partition.items():
    print(f"Співтовариство {comm}: {list(nodes)}")
```

Файл SIR-model.py – моделювання поширення вірусної інформації у віртуальній соціальній мережі (SIR-модель)

```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
import random

# Створення графа

# Параметри графа
n = 70  # Кількість вузлів
k = 4   # Початковий ступінь вузла
p = 0.1 # Ймовірність перепідключення замість додавання ребра


# Створення початкового регулярного графа
G = nx.Graph()
for i in range(n):
    for j in range(1, k // 2 + 1):
        G.add_edge(i, (i + j) % n)


# Перепідключення ребер
for i in range(n):
    for j in range(1, k // 2 + 1):
        if random.random() < p:
            new_neighbor = random.randint(0, n - 1)
            while new_neighbor == i or G.has_edge(i, new_neighbor):
                new_neighbor = random.randint(0, n - 1)
            G.remove_edge(i, (i + j) % n)
            G.add_edge(i, new_neighbor)


# Ініціалізуємо стани SIR моделі
infected_nodes = random.sample(list(G.nodes()), k=2)# Початкові інфіковані
infected_nodes = random.sample(list(G.nodes()), k=2)
susceptible_nodes = set(G.nodes()) - set(infected_nodes)
recovered_nodes = set()


# Параметри моделі
beta = 0.2  # Імовірність передачі вірусу
gamma = 0.3  # Імовірність одужання


# Функція для виводу графа з кольорами відповідно до стану
def plot_graph(G, infected_nodes, susceptible_nodes, recovered_nodes):
    node_colors = []
    for node in G.nodes():
        if node in infected_nodes:
            node_colors.append('red')
        elif node in susceptible_nodes:
            node_colors.append('blue')
        else:
            node_colors.append('green')

    nx.draw(G, with_labels=True, node_color=node_colors)
    plt.show()


# Візуалізуємо початковий стан графа
plot_graph(G, infected_nodes, susceptible_nodes, recovered_nodes)
```

```
# Моделюємо поширення вірусу
iterations = 10
for t in range(iterations):
    new_infections = set()
    new_recoveries = set()

    for node in infected_nodes:
        for neighbor in G.neighbors(node):
            if neighbor in susceptible_nodes and random.random() < beta:
                new_infections.add(neighbor)
        if random.random() < gamma:
            new_recoveries.add(node)


    # infected_nodes |= new_infections
    infected_nodes = set(infected_nodes) | new_infections
    recovered_nodes |= new_recoveries
    susceptible_nodes -= new_infections


    # Візуалізуємо граф після кожної ітерації
    plot_graph(G, infected_nodes, susceptible_nodes, recovered_nodes)
```

Файл SIRS-model.py – моделювання поширення вірусної інформації у віртуальній соціальній мережі (SIRS-модель)

```
import networkx as nx
import random
import matplotlib.pyplot as plt
import numpy as np


# Створення графа та ініціалізація станів
n = 100  # Кількість користувачів (вузлів)
k = 4   # Початковий ступінь вузла
p = 0.7 # Ймовірність перепідключення замість додавання ребра


# Генерація рис користувачів (10 випадкових чисел для кожного)
user_attributes = {i: np.random.rand(10) for i in range(n)}


# Функція для обчислення коефіцієнта кореляції Пірсона між рисами користувачів
def compute_similarity(user1, user2):
    return np.corrcoef(user1, user2)[0, 1]


# Створення початкового регулярного графа
G = nx.Graph()
for i in range(n):
    for j in range(1, k // 2 + 1):
        G.add_edge(i, (i + j) % n)


# Перепідключення ребер з урахуванням схожості користувачів
for i in range(n):
    for j in range(1, k // 2 + 1):
        if random.random() < p:
            new_neighbor = random.randint(0, n - 1)
            while new_neighbor == i or G.has_edge(i, new_neighbor):
                new_neighbor = random.randint(0, n - 1)
            similarity = compute_similarity(user_attributes[i],
user_attributes[new_neighbor])
            connection_probability = similarity
            if random.random() < connection_probability:
                G.remove_edge(i, (i + j) % n)
                G.add_edge(i, new_neighbor)


# Параметри моделі
initial_infections = random.sample(range(100), k=5)  # Початкові інфіковані
recovery_probability = 0.3  # Імовірність одужання
loss_immunity_probability = 0.1  # Імовірність втрати імунітету
vulnerability = [random.uniform(0, 1) for _ in range(n)]  # Вразливість
користувачів до вірусу


# Початковий стан графа
node_states = {node: "S" for node in G.nodes()}
for node in initial_infections:
    node_states[node] = "I"

# Ініціалізація даних для графіків
infection_counts = []
immunity_counts = []
```

```
# Функція для візуалізації графа
def plot_graph(G, node_states):
    node_colors = {"S": "blue", "I": "red", "R": "green"}
    colors = [node_colors[state] for state in node_states.values()]
    pos = nx.spring_layout(G)
    nx.draw(G, pos, node_color=colors, with_labels=True)
    plt.show()


# Проведення ітерацій
num_iterations = 50
for t in range(num_iterations):
    new_infections = set()
    new_recoveries = set()

    for node in G.nodes():
        if node_states[node] == "I":
            # Інфікований може одужати з імовірністю recovery_probability
            if random.random() < recovery_probability:
                new_recoveries.add(node)

        if node_states[node] == "R":
            # "Відновлений" може втратити імунітет з імовірністю
loss_immunity_probability
            if random.random() < loss_immunity_probability:
                node_states[node] = "S"

        if node_states[node] == "S":
            # Сприйнятливий може заразитися від інфікованих сусідів
            for neighbor in G.neighbors(node):
                if node_states[neighbor] == "I":
                    if random.random() < vulnerability[node]:  # Імовірність
заразитися від сусіда
                        new_infections.add(node)
                        break

    for node in new_infections:
        node_states[node] = "I"
    for node in new_recoveries:
        node_states[node] = "R"


 # Візуалізація графа на кожній ітерації
    plot_graph(G, node_states)


  # Рахунок кількості інфікованих та осіб з імунітетом
    infection_counts.append(list(node_states.values()).count("I"))
    immunity_counts.append(list(node_states.values()).count("R"))



# Графіки поширення інфекції та імунітету у часі
plt.plot(range(num_iterations), infection_counts, label="Infections")
plt.plot(range(num_iterations), immunity_counts, label="Immunity")
plt.xlabel("Time")
plt.ylabel("Count")
plt.legend()
plt.show()
```

Файл networkClusters.py – визначення кластерів (співтовариств) віртуальної соціальної мережі

```
import networkx as nx
import matplotlib.pyplot as plt
import random

import community  # Бібліотека для використання алгоритму Лувейна


def freeman_network_model(n, p):
    G = nx.Graph()

    for i in range(n):
        G.add_node(i)

    for i in range(n):
        for j in range(i + 1, n):
            if random.random() < p:
                G.add_edge(i, j)

    return G


n = 50   # Кількість вузлів
p = 0.2  # Ймовірність створення зв'язку між вузлами


# Створення графа за моделлю Фреймана
G = freeman_network_model(n, p)


# Знаходження співтовариств у графі за допомогою алгоритму Лувейна
partition = community.best_partition(G)


# Визначення позицій вершин з одного співтовариства поруч, але не ідентичними
pos = nx.spring_layout(G, seed=42)

for comm in set(partition.values()):
    nodes_in_comm = [node for node, comm_id in partition.items() if comm_id ==
comm]
    x_positions = [pos[node][0] for node in nodes_in_comm]
    y_positions = [pos[node][1] for node in nodes_in_comm]
    avg_x = sum(x_positions) / len(x_positions)
    avg_y = sum(y_positions) / len(y_positions)
    for node in nodes_in_comm:
        pos[node] = (avg_x + random.uniform(-0.1, 0.1), avg_y + random.uniform(-
0.1, 0.1))


plt.figure(figsize=(8, 6))
cmap = plt.get_cmap("viridis", max(partition.values()) + 1)
node_colors = [cmap(partition[node]) for node in G.nodes()]
node_sizes = [10 * G.degree[node] for node in G.nodes()]


# Візуалізація графа з вершинами з одного співтовариства поруч
nx.draw(G, pos, with_labels=False, node_size=node_sizes, node_color=node_colors,
edge_color='gray', width=0.5, alpha=0.7)
plt.show()


# Виведення співтовариств і списків вершин у них текстом
print("Співтовариства:")
communities = {}
```

```
for node, comm_id in partition.items():
    if comm_id not in communities:
        communities[comm_id] = []
    communities[comm_id].append(node)

for comm_id, nodes in communities.items():
    print(f"Співтовариство {comm_id}: {nodes}")
```

Файл nodeCentrality1.py – визначення центральностей вузлів віртуальної соціальної мережі за степенем

```
import networkx as nx
import matplotlib.pyplot as plt
import random


def freeman_network_model(n, p):
    G = nx.Graph()

    for i in range(n):
        G.add_node(i)

    for i in range(n):
        for j in range(i + 1, n):
            if random.random() < p:
                G.add_edge(i, j)

    return G


def degree_centrality_size(G):
    # Визначення центральності за степенем
    degree_centrality = nx.degree_centrality(G)

    # Визначення розмірів вершин на основі центральності
    node_sizes = [2000 * degree_centrality[node] for node in G.nodes()]

    return degree_centrality, node_sizes


n = 50   # Кількість вузлів
p = 0.1  # Ймовірність створення зв'язку між вузлами


# Створення графа за моделлю Фреймана
G = freeman_network_model(n, p)


# Визначення позицій вершин
pos = nx.spring_layout(G, seed=42)


# Визначення центральності за степенем та розмірів вершин
degree_centrality, node_sizes = degree_centrality_size(G)

plt.figure(figsize=(8, 6))


# Візуалізація графа з вершинами різних розмірів
nx.draw(G, pos, with_labels=False, node_size=node_sizes, node_color='skyblue',
edge_color='gray', width=0.5, alpha=0.7)


# Виведення номерів вершин та їх центральностей текстом
print("Центральність вершин:")
for node, centrality in degree_centrality.items():
    print(f"Вершина {node}: Центральність = {centrality:.2f}")

plt.show()
```

Файл nodeCentrality2.py – визначення центральностей вузлів віртуальної соціальної мережі за близькістю

```
import networkx as nx
import matplotlib.pyplot as plt
import random

def freeman_network_model(n, p):
    G = nx.Graph()

    for i in range(n):
        G.add_node(i)

    for i in range(n):
        for j in range(i + 1, n):
            if random.random() < p:
                G.add_edge(i, j)

    return G


def closeness_centrality_size(G):
    # Визначення центральності за близькістю
    closeness_centrality = nx.closeness_centrality(G)


  # Визначення розмірів вершин на основі центральності
    node_sizes = [2000 * closeness_centrality[node] for node in G.nodes()]

    return closeness_centrality, node_sizes

n = 10   # Кількість вузлів
p = 0.3  # Ймовірність створення зв'язку між вузлами


# Створення графа за моделлю Фреймана
G = freeman_network_model(n, p)


# Визначення позицій вершин
pos = nx.spring_layout(G, seed=42)


# Визначення центральності за близькістю та розмірів вершин
closeness_centrality, node_sizes = closeness_centrality_size(G)

plt.figure(figsize=(8, 6))


# Візуалізація графа з вершинами різних розмірів
nx.draw(G, pos, with_labels=False, node_size=node_sizes, node_color='skyblue',
edge_color='gray', width=0.5, alpha=0.7)


# Виведення номерів вершин та їх центральностей текстом
print("Центральність вершин:")
for node, centrality in closeness_centrality.items():
    print(f"Вершина {node}: Центральність = {centrality:.2f}")

plt.show()
```

Файл nodeCentrality3.py – визначення центральностей вузлів віртуальної соціальної мережі за власним вектором

```
import networkx as nx
import matplotlib.pyplot as plt
import random


def freeman_network_model(n, p):
    G = nx.Graph()

    for i in range(n):
        G.add_node(i)

    for i in range(n):
        for j in range(i + 1, n):
            if random.random() < p:
                G.add_edge(i, j)

    return G


def eigenvector_centrality_size(G):
    # Визначення центральності за власним вектором
    eigenvector_centrality = nx.eigenvector_centrality(G)


    # Визначення розмірів вершин на основі центральності
    node_sizes = [2000 * eigenvector_centrality[node] for node in G.nodes()]

    return eigenvector_centrality, node_sizes

n = 50   # Кількість вузлів
p = 0.1  # Ймовірність створення зв'язку між вузлами


# Створення графа за моделлю Фреймана
G = freeman_network_model(n, p)


# Визначення позицій вершин
pos = nx.spring_layout(G, seed=42)


# Визначення центральності за власним вектором та розмірів вершин
eigenvector_centrality, node_sizes = eigenvector_centrality_size(G)

plt.figure(figsize=(8, 6))


# Візуалізація графа з вершинами різних розмірів
nx.draw(G, pos, with_labels=False, node_size=node_sizes, node_color='skyblue',
edge_color='gray', width=0.5, alpha=0.7)


# Виведення номерів вершин та їх центральностей текстом
print("Центральність вершин за власним вектором:")
for node, centrality in eigenvector_centrality.items():
    print(f"Вершина {node}: Центральність = {centrality:.2f}")

plt.show()
```

Файл networkParameters.py – визначення загальних властивостей графу віртуальної соціальної мережі

```
import networkx as nx
import matplotlib.pyplot as plt

import community  # Для визначення співтовариств


# Параметри графа
n = 20  # Кількість вузлів
k = 4   # Початковий ступінь вузла
p = 0.3 # Ймовірність перепідключення замість додавання ребра


# Генерація графа за моделлю Ваттса-Строгатца
G = nx.watts_strogatz_graph(n, k, p)


# Визначення центральності за ступенем
degree_centrality = nx.degree_centrality(G)


# Визначення співтовариств
partition = community.best_partition(G)


# Візуалізація графа з позначенням співтовариств
pos = nx.spring_layout(G, seed=42)  # Позиціонування графа
nx.draw(G, pos, with_labels=True, node_size=100, font_size=8)


# Виведення номерів співтовариств та їх складу
print("Співтовариства:")
for community_id in set(partition.values()):
    nodes_in_community = [node for node, comm_id in partition.items() if comm_id
== community_id]
    print(f"Співтовариство {community_id}: {nodes_in_community}")


# Виведення центральності за ступенем
print("\nЦентральність за ступенем:")
for node, centrality in degree_centrality.items():
    print(f"Вузол {node}: Центральність = {centrality:.4f}")

plt.title("Граф за моделлю Ваттса-Строгатца зі співтовариствами")
plt.show()


# Визначення параметрів графу
num_nodes = len(G.nodes())
num_edges = len(G.edges())
average_degree = 2 * num_edges / num_nodes  # Середній ступінь вузла
clustering_coefficient = nx.average_clustering(G)  # Коефіцієнт кластеризації
diameter = nx.diameter(G)  # Діаметр графу
density = nx.density(G)  # Густина зв'язків


# Виведення результатів
print(f"Кількість вузлів: {num_nodes}")
print(f"Кількість ребер: {num_edges}")
print(f"Середній ступінь вузла: {average_degree}")
print(f"Коефіцієнт кластеризації: {clustering_coefficient}")
print(f"Діаметр графу: {diameter}")
print(f"Густина зв'язків: {density}")
```

Файл VisualizationRandom.py – візуалізація графу соціальної мережі з випадковим розташуванням вершин

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import random


# Параметри графа
n = 20  # Кількість вузлів
k = 4   # Початковий ступінь вузла
p = 0.3 # Ймовірність перепідключення замість додавання ребра


# Генерація графа за моделлю Ваттса-Строгатца
G = nx.watts_strogatz_graph(n, k, p)


# Задаємо випадкові координати для вершин графу
pos = {node: (random.uniform(0, 1), random.uniform(0, 1)) for node in G.nodes()}


# Візуалізація графу з випадковим розташуванням вершин
nx.draw(G, pos, with_labels=True, node_size=500, node_color='skyblue',
font_size=12, font_color='black', font_weight='bold', edge_color='gray',
width=2)


# Налаштування відображення
plt.title("Граф соціальної мережі (з випадковим розташуванням)")
plt.axis('off')  # Вимкнення відображення координатних вісей
                                    plt.show()
```

Файл VisualizationGraphLaplace.py – візуалізація графу соціальної мережі у вигляді графа Лапласа

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt


# Параметри графа
n = 20  # Кількість вузлів
k = 4   # Початковий ступінь вузла
p = 0.3 # Ймовірність перепідключення замість додавання ребра


# Генерація графа за моделлю Ваттса-Строгатца
G = nx.watts_strogatz_graph(n, k, p)


# Отримання матриці суміжності
adj_matrix = nx.adjacency_matrix(G)


# Обчислення матриці Лапласа
laplacian_matrix = np.diag(np.array(adj_matrix.sum(axis=1)).flatten()) -
adj_matrix


# Отримання власних значень та векторів матриці Лапласа
eigenvalues, eigenvectors = np.linalg.eig(laplacian_matrix)


# Візуалізація графа за допомогою власних векторів
pos = nx.spring_layout(G)

eigen_vector = eigenvectors[:, 5]
node_colors = eigen_vector.tolist()
nx.draw(G, pos, node_color=node_colors, cmap=plt.get_cmap('coolwarm'),
with_labels=True)
plt.show()
```

Файл VisualizationFruchtermanReingold.py – візуалізація графу соціальної мережі алгоритмом Fruchterman-Reingold

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt


# Параметри графа
n = 20  # Кількість вузлів
k = 4   # Початковий ступінь вузла
p = 0.3 # Ймовірність перепідключення замість додавання ребра


# Генерація графа за моделлю Ваттса-Строгатца
G = nx.watts_strogatz_graph(n, k, p)


# Візуалізація графу за допомогою алгоритму Fruchterman-Reingold
pos = nx.spring_layout(G, seed=42)  # seed для відтворюваності розташування
вузлів
nx.draw(G, pos, with_labels=True, node_size=500, node_color='skyblue',
font_size=12, font_color='black', font_weight='bold', edge_color='gray',
width=2)


# Налаштування відображення
plt.axis('off')  # Вимкнення відображення координатних вісей
plt.show()
```

Файл BDwrite.py – запис структури графу віртуальної соціальної мережі в базу даних Neo4j

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

from py2neo import Graph # бібліотека для роботи з базою даних Neo4j


# Параметри графа
n = 20  # Кількість вузлів
k = 4   # Початковий ступінь вузла
p = 0.3 # Ймовірність перепідключення замість додавання ребра


# Генерація графа за моделлю Ваттса-Строгатца
G = nx.watts_strogatz_graph(n, k, p)


# Підключення до бази даних Neo4j
url = "bolt://localhost:7687"
username = "user1"
password = "123"

graph = Graph(url, auth=(username, password))


# Збереження графа у базу даних Neo4j
for node in G.nodes():
    node_properties = {"name": f"Node_{node}", "degree": G.degree[node]}
    graph.merge_one("Node", "name", node_properties)


for edge in G.edges():
    start_node = G.nodes[edge[0]]
    end_node = G.nodes[edge[1]]
    relationship_properties = {"type": "CONNECTED"}
    graph.merge_one("Node", "name", start_node)
    graph.merge_one("Node", "name", end_node)
    graph.merge_one(start_node, "CONNECTED", end_node,
properties=relationship_properties)
```

Файл BDread.py – читання структури графу віртуальної соціальної мережі з бази даних Neo4j

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

from py2neo import Graph # бібліотека для роботи з базою даних Neo4j


# Параметри підключення до бази даних Neo4j
url = "bolt://localhost:7687"
username = "user1"
password = "123"


# Створення об'єкту для підключення до бази даних
graph = Graph(url, auth=(username, password))


# Запит для отримання графа з бази даних
query = """
MATCH (n:Node)-[:CONNECTED]->(m:Node)
RETURN n, m
"""


# Виконання запиту та створення графа
result = graph.run(query)
G = nx.Graph()

for record in result:
    node1 = record["n"]
    node2 = record["m"]
    G.add_node(node1["name"])
    G.add_node(node2["name"])
    G.add_edge(node1["name"], node2["name"])


# Вивід інформації про граф
print("Вершини графа:", G.nodes())
print("Ребра графа:", G.edges())


# Візуалізація графу за допомогою алгоритму Fruchterman-Reingold
pos = nx.spring_layout(G, seed=42)  # seed для відтворюваності розташування
вузлів
nx.draw(G, pos, with_labels=True, node_size=500, node_color='skyblue',
font_size=12, font_color='black', font_weight='bold', edge_color='gray',
width=2)

# Налаштування відображення
plt.axis('off')  # Вимкнення відображення координатних вісей
plt.show()
```

Файл BDreportWrite.py – запис звіту після аналізу загальних властивостей віртуальної соціальної мережі у БД Neo4j

```
import numpy as np
import networkx as nx

import matplotlib.pyplot as plt
from py2neo import Graph



# Параметри підключення до бази даних Neo4j
url = "bolt://localhost:7687"
username = "user1"
password = "123"


# Створення об'єкту для підключення до бази даних
graph = Graph(url, auth=(username, password))


# Запит для отримання графа з бази даних
query = """
MATCH (n:Node)-[:CONNECTED]->(m:Node)
RETURN n, m
"""


# Виконання запиту та створення графа
result = graph.run(query)
G = nx.Graph()

for record in result:
    node1 = record["n"]
    node2 = record["m"]
    G.add_node(node1["name"])
    G.add_node(node2["name"])
    G.add_edge(node1["name"], node2["name"])


# Визначення параметрів графу
num_nodes = len(G.nodes())
num_edges = len(G.edges())
average_degree = 2 * num_edges / num_nodes
clustering_coefficient = nx.average_clustering(G)
diameter = nx.diameter(G)
density = nx.density(G)


# Запит для запису параметрів графу у базу даних
query = """
MATCH (g:Graph {name: 'SocialNetworkGraph'})
SET g.num_nodes = $num_nodes,
    g.num_edges = $num_edges,
    g.average_degree = $average_degree,
    g.clustering_coefficient = $clustering_coefficient,
    g.diameter = $diameter,
    g.density = $density
"""



params = {
    "num_nodes": num_nodes,
    "num_edges": num_edges,
    "average_degree": average_degree,
```

```
    "clustering_coefficient": clustering_coefficient,
    "diameter": diameter,
    "density": density
}

graph.run(query, params=params)
```

Файл BDreportRead.py – читання звіту про аналіз загальних властивостей віртуальної соціальної мережі з БД Neo4j

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from py2neo import Graph


# Параметри підключення до бази даних Neo4j
url = "bolt://localhost:7687"
username = "user1"
password = "123"


# Створення об'єкту для підключення до бази даних
graph = Graph(url, auth=(username, password))


# Запит для отримання параметрів графа з бази даних
query = """
MATCH (g:Graph {name: 'SocialNetworkGraph'})
RETURN g.num_nodes AS num_nodes,
       g.num_edges AS num_edges,
       g.average_degree AS average_degree,
       g.clustering_coefficient AS clustering_coefficient,
       g.diameter AS diameter,
       g.density AS density
"""


# Виконання запиту та отримання результатів
result = graph.run(query).single()


# Отримання параметрів з результатів запиту
num_nodes = result["num_nodes"]
num_edges = result["num_edges"]
average_degree = result["average_degree"]
clustering_coefficient = result["clustering_coefficient"]
diameter = result["diameter"]
density = result["density"]


# Виведення параметрів графа
print("Кількість вузлів:", num_nodes)
print("Кількість ребер:", num_edges)
print("Середній ступінь вузла:", average_degree)
print("Коефіцієнт кластеризації:", clustering_coefficient)
print("Діаметр графу:", diameter)
print("Густина зв'язків:", density)
```