Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
(повна назва)

Факультет Інфокомунікацій

Кафедра Інформаційно-мережної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Проєктування системи підтримки прийняття рішень при експлуатації IT-інфраструктури підприємства
(тема)

Виконав: студент 2 курсу, групи ІМІм-21-2
Войлов В.І.
(прізвище, ініціали)

Спеціальність: 172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)

Тип програми: освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма: Інформаційно-мережна інженерія
(повна назва освітньої програми)

Керівник: ст. викладач Калюжний М. М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Безрук В. М.
(підпис) (прізвище, ініціали)

Харків 2023

Не містить відомостей, заборонених
до відкритого публікування

Студент _____ / В.І. Войлов /

Керівник_____/ М.М. Калюжний /

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інформаційно-мережної інженерії
(повна назва)
Рівень вищої освіти другий (магістерський)

Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Інформаційно-мерережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ
Зав. кафедри «ІМІ»
_____проф. Безрук В. М.
(підпис)
«____»_____2023р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Войлову Владиславу Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Проєктування системи підтримки прийняття рішень при експлуатації ІТ-інфраструктури підприємства
затверджена наказом по університету від «27» квітня 2023р. №398 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 18.05.2023р.
3. Вихідні дані до роботи: розробити та продемонструвати практичний приклад успішного застосування машинного навчання та нейронних мереж у контексті автоматизації ІТ-інфраструктури підприємства
4. Перелік питань, що потрібно опрацювати в роботі:
    4.1 Аналіз існуючих підходів, методів і технологій автоматизації ІТ-інфраструктури підприємств
    4.2 Обґрунтування методу і моделі прогнозування даних для служби

підтримки користувачів ІТ інфраструктури підприємства.

4.3 Обґрунтування і вибір архітектури, мови програмування, бази даних для розробки програмного забезпечення (ПЗ)

4.4 Розробка ПЗ системи підтримки прийняття рішень для користувачів ІТ інфраструктури підприємства.

4.5 Експериментальна перевірка та тестування розробленої системи.

5. Перелік графічного матеріалу із зазначенням діаграм, інтерфейсу ПЗ, комп'ютерних ілюстрацій: Демонстраційний матеріал у вигляді PPT-презентації

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення і уточнення завдання	01.02-05.02.2023	Виконано
2	Підбір і вивчення літератури	01.02-10.02.2023	Виконано
3	Розробка розділу 1	10.02-22.02.2023	Виконано
4	Розробка розділу 2	23.02-22.03.2023	Виконано
5	Розробка розділу 3	23.03-30.04.2023	Виконано
6	Розробка і експериментальна перевірка ПЗ	23.03-30.04.2023	Виконано
7	Оформлення кваліфікаційної роботи	01.05-08.05.2023	Виконано

Дата видачі завдання 01.02.2023 року

Студент _____ Войлов В.І.
(підпис) (прізвище, ініціали)

Керівник роботи _____ ст. викладач Калюжний М.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи: 98 с., 41 рис., 8 табл., 50 джерел, 16 слайдів, лістинги програми 44 с., наукова публікація 3 с.

Ключові слова: НЕЙРОННА МЕРЕЖА, МАШИННЕ НАВЧАННЯ, ІТ-ІНФРАСТРУКТУРА, СИСТЕМА ПІДТРИМКИ.

У роботі було розроблено та продемонстровано практичний приклад успішного застосування машинного навчання та нейронних мереж у контексті автоматизації ІТ-інфраструктури підприємства. Для досягнення цієї мети було розглянуто існуючі підходи, методи і технології автоматизації ІТ-інфраструктури підприємств, а також обґрунтовано метод та модель прогнозування даних для служби підтримки користувачів ІТ інфраструктури підприємства.

Результати дослідження показують значний потенціал цих технологій у покращенні ефективності та ефективності ІТ-інфраструктури підприємства.

ABSTRACT

Explanatory Note for the diploma thesis: 98 pages, 41 figures, 8 tables, 50 references, 16 slides, program listings 44 pages, scientific publication 3 pages.

Keywords: NEURAL NETWORK, MACHINE LEARNING, IT INFRASTRUCTURE, SUPPORT SYSTEM.

The thesis presents and demonstrates a practical example of successful application of machine learning and neural networks in the context of automating enterprise IT infrastructure. To achieve this goal, existing approaches, methods, and technologies for automating enterprise IT infrastructure were examined, and a method and data forecasting model for the IT infrastructure user support service were justified.

The research findings highlight the significant potential of these technologies in improving the efficiency and effectiveness of enterprise IT infrastructure.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

GRNN - generalized regression neural network (узагальнено регресійна нейронна мережа);

MLP – multi-layer perceptron (багатошаровий персептрон);

RBN – radial-basis network (радіально-базисна мережа);

БД – база даних;

ІТ – інформаційна технологія;

ПЗ – програмне забезпечення;

СУБД – система управління базою даних.

ВСТУП

На сучасному етапі розвитку технологій, автоматизація IT-інфраструктури підприємства набуває особливого значення. Це зумовлено прагненням до зниження витрат на робочу силу, зменшення помилок, підвищення швидкості вирішення задач та забезпечення безперервної роботи системи. Застосування машинного навчання та нейронних мереж в інформаційних технологіях відкриває нові можливості для оптимізації роботи IT-інфраструктури підприємства.

Одним із напрямків застосування машинного навчання та нейронних мереж у IT-інфраструктурі є аналіз та оптимізація роботи служби підтримки користувачів, а також прогнозування та виявлення можливих проблем в роботі інфраструктури. Використання передових алгоритмів машинного навчання та нейронних мереж може допомогти підвищити ефективність процесів автоматизації, скоротити витрати часу та ресурсів та забезпечити стабільність IT-інфраструктури підприємства.

Для досягнення цих цілей, на підприємствах розглядається можливість розробки та імплементації моделей машинного навчання та нейронних мереж, які здатні автоматично аналізувати та передбачати типи проблем, що можуть виникнути, та пропонувати ефективні рішення для їх усунення. Такі моделі можуть допомогти в автоматизації процесів моніторингу, аналізу та виявлення проблем, що дозволить зменшити навантаження на фахівців IT-служби та забезпечити швидке вирішення проблем, які виникають у IT-інфраструктурі підприємства.

Важливим аспектом автоматизації IT-інфраструктури є вибір відповідних інструментів та технологій для розробки та впровадження моделей машинного навчання та нейронних мереж. У різних випадках можуть бути використані такі технології, як TensorFlow, PyTorch, Keras, а також ML

.NET. Вибір конкретної технології залежить від особливостей підприємства, його IT-інфраструктури та вимог до автоматизації.

У подальшому, успішно впроваджені моделі машинного навчання та нейронних мереж можуть стати основою для створення більш розгалуженої системи автоматизації IT-інфраструктури підприємства. Такі системи можуть включати автоматизацію рутинних завдань, таких як розгортання та оновлення програмного забезпечення, налаштування мережевих пристроїв, а також автоматичне виявлення та реагування на збої в роботі системи.

У рамках даної роботи буде досліджено підхід до автоматизації IT-інфраструктури підприємства за допомогою машинного навчання та нейронних мереж. Зокрема, будуть проаналізовані можливості застосування цих технологій для оптимізації роботи служби підтримки користувачів та прогнозування та виявлення можливих проблем в роботі IT-інфраструктури. Окрім того, буде розглянуто наявні інструменти та технології для реалізації подібних проектів та оцінено їх переваги та недоліки.

Актуальність даної роботи обумовлена стрімким розвитком інформаційних технологій та зростанням ролі IT-інфраструктури в житті сучасного підприємства. Інтенсивна інтеграція новітніх технологій у бізнес-процеси вимагає постійного удосконалення і оптимізації роботи IT-інфраструктури для забезпечення стабільності, ефективності та конкурентоспроможності підприємства на ринку.

Одним із способів підвищення ефективності роботи IT-інфраструктури є її автоматизація. Застосування машинного навчання та нейронних мереж стає все більш актуальним, оскільки ці технології дозволяють аналізувати та передбачати проблеми, що можуть виникнути в роботі інфраструктури, та пропонувати ефективні рішення для їх усунення. Автоматизація IT-інфраструктури за допомогою машинного навчання та нейронних мереж

відкриває нові можливості для покращення процесів управління, зменшення помилок та підвищення продуктивності ІТ-фахівців.

Актуальність цієї теми полягає у необхідності дослідження та вивчення потенціалу машинного навчання та нейронних мереж у контексті автоматизації ІТ-інфраструктури підприємства. Зростає попит на спеціалістів, які здатні розробляти та імплементувати інноваційні рішення, засновані на застосуванні машинного навчання та нейронних мереж.

Метою даного дослідження є вивчення можливостей машинного навчання та нейронних мереж для автоматизації ІТ-інфраструктури підприємства і розробка на їх основі системи підтримки прийняття рішень для оптимізації роботи служби підтримки користувачів.

Завдання дослідження:

– провести аналіз існуючих підходів, методів та технологій машинного навчання та нейронних мереж, які можуть бути використані для автоматизації ІТ-інфраструктури підприємства.

– вивчити потенціал машинного навчання та нейронних мереж у вирішенні різних завдань, пов'язаних з автоматизацією процесів управління ІТ-інфраструктурою, оптимізацією роботи служби підтримки користувачів;

– вибрати оптимальні інструменти та технології для розробки ПЗ;

– продемонструвати практичні приклади успішного застосування машинного навчання та нейронних мереж у контексті автоматизації ІТ-інфраструктури підприємства.

Об'єктом даного дослідження є ІТ-інфраструктура підприємства.

Предметом дослідження є застосування машинного навчання та нейронних мереж для автоматизації роботи служби підтримки користувачів ІТ-інфраструктури підприємства.

Наукова новизна. На основі дослідженого матеріалу було розроблено продукт, для автоматизації роботи служби підтримки користувачів IT інфраструктури. Отримані результати та висновки можуть бути використані для покращення процесів управління IT-інфраструктурою та підвищення ефективності роботи підприємства в цілому.

1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ, МЕТОДІВ І ТЕХНОЛОГІЙ АВТОМАТИЗАЦІЇ IT-ІНФРАСТРУКТУРИ ПІДПРИЄМСТВ

1.1 IT-інфраструктура компанії: поняття, завдання та особливості

IT-інфраструктура компанії є важливим компонентом, який забезпечує ефективну роботу компанії. IT-інфраструктура включає в себе все, що стосується комп'ютерної техніки та програмного забезпечення, мережевої інфраструктури та інших компонентів, що використовуються для забезпечення роботи компанії. Вона є основою для обслуговування клієнтів, роботи з постачальниками та управління бізнес-процесами. Завдяки IT-інфраструктурі компанія може швидко та ефективно взаємодіяти з клієнтами та постачальниками, забезпечувати якісний сервіс та максимальну задоволеність клієнтів. Крім того, IT-інфраструктура є основою для управління бізнес-процесами компанії, що дозволяє забезпечити ефективний контроль та управління різними процесами, такими як виробництво, логістика, маркетинг та інші.

IT-інфраструктура визначає можливості компанії сьогодні та її можливості у майбутньому [1]. Компанії, які мають потужну та ефективну IT-інфраструктуру, здатні швидко реагувати на зміни на ринку та забезпечувати інноваційний розвиток бізнесу. В той же час, компанії, які мають застарілу та неефективну IT-інфраструктуру, можуть відставати від конкурентів та втрачати свої позиції на ринку.

Тому важливо не тільки забезпечити наявність IT-інфраструктури в компанії, але й розробити стратегію її розвитку та діяти цілеспрямовано для

забезпечення її ефективної та безперебійної роботи в майбутньому. Важливим етапом є регулярне оновлення технічної бази та програмного забезпечення, що дозволяє забезпечувати стабільну та безперебійну роботу IT-інфраструктури.

Послуги, які компанія надає своїм клієнтам, постачальникам та співробітникам, є прямою функцією її IT-інфраструктури. В ідеалі ця інфраструктура має підтримувати бізнес-стратегію фірми та стратегію інформаційних систем. Нові інформаційні технології впливають на бізнес та IT-стратегії, а також на послуги, які можуть бути надані клієнтам.

IT-інфраструктуру можна розглядати як технологічні чи сервісні кластери. Визначення, що базується на послугах, фокусується на послугах, що надаються апаратним та програмним забезпеченням, такими як [2]:

– обчислювальні платформи;
– телекомунікації;
– керування фізичними об'єктами;
– прикладне програмне забезпечення;
– управління даними;
– управління IT;
– IT-стандарти;
– IT-освіта та IT-дослідження та розробки.

В еволюції IT-інфраструктури можна виділити п'ять етапів [3]:

– децентралізована інфраструктура: це перший етап еволюції, коли комп'ютери та інші пристрої були розташовані в окремих відділах компанії, не були пов'язані між собою та працювали на окремих програмах;

– централізована інфраструктура: на другому етапі, компанії почали об'єднувати свої комп'ютери в мережі, що дозволяло забезпечувати більш ефективний обмін інформацією та ресурсами;

– віртуалізована інфраструктура: третій етап еволюції пов'язаний з віртуалізацією, коли ресурси IT-інфраструктури стали доступні для використання з будь-якої точки мережі, що дозволяє ефективніше

використовувати обчислювальну потужність та зменшити витрати на обладнання [4];

– хмарна інфраструктура: на четвертому етапі IT-інфраструктура компанії стає більш гнучкою та доступною завдяки хмарним технологіям. Компанії можуть отримати доступ до потрібних інструментів та ресурсів від хмарного провайдера з будь-якої точки світу;

– інфраструктура майбутнього: останній етап пов'язаний зі збільшенням використання штучного інтелекту та інших передових технологій в IT-інфраструктурі. Інфраструктура майбутнього повинна бути більш гнучкою, ефективною та забезпечувати безпеку даних компанії [5].

У сучасних реаліях важко уявити бізнес, який працює без хоча б якоїсь базової IT-інфраструктури. Цифрові рішення та обладнання, пов'язані один з одним, допомагають підвищити продуктивність компанії. А хороша інфраструктура покращує внутрішні процеси та взаємодію між різними підрозділами.

IT-інфраструктура заснована на різних компонентах, спрямованих на управління внутрішньою бізнес-екосистемою або надання послуг за межами бізнесу. Чим краще та продуманіше організована інфраструктура, тим більше бізнес може отримувати прибутки та розширюватися.

Не можливо створити IT-інфраструктуру з нічого. Вся екосистема складається з програмних рішень, різного обладнання та мережевих підключень, які працюють одночасно та доповнюють один одного. Вся ідея полягає в покращенні зв'язку між різними пристроями, чи то настільні комп'ютери, планшети, сканери, різні сервери та хмарні сховища.

Сучасна організація будь-якого розміру завжди спиратиметься на три найважливіші IT-будівельні блоки. Їх можна розділити на додаткові підкомпоненти (рис. 1.1).

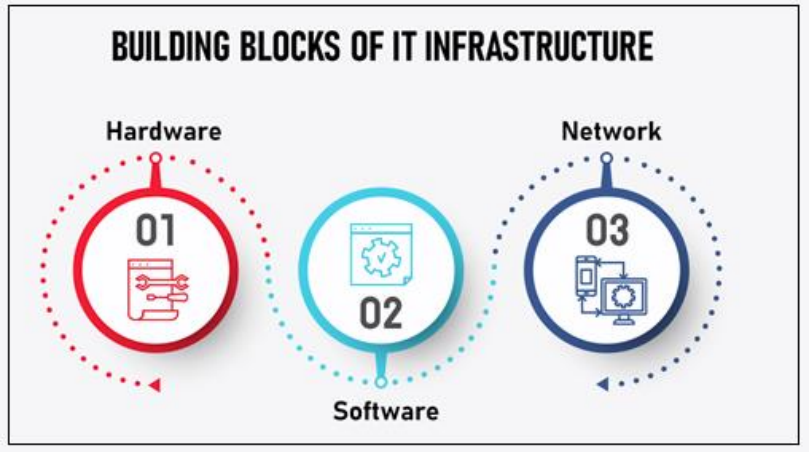


Рисунок 1.1 – Будівельні блоки IT-інфраструктури

Докладніше розглянемо основні компоненти IT-інфраструктури:

Апаратне забезпечення

Апаратне забезпечення IT-інфраструктури складається з різних компонентів, що працюють разом, щоб забезпечити ефективну роботу системи. До складу апаратної інфраструктури зазвичай входять наступні елементи [6]:

– сервери: це потужні комп'ютери, які забезпечують зберігання даних та обчислювальну потужність для інших компонентів IT-інфраструктури. Сервери можуть використовуватися для різних цілей, таких як зберігання файлів, робота з базами даних, забезпечення мережевої безпеки та ін;

– робочі станції: це комп'ютери, що використовуються користувачами для роботи з даними та програмами. Робочі станції можуть мати різні характеристики, залежно від потреб користувачів та вимог до продуктивності;

– принтери та інші периферійні пристрої. До складу апаратної інфраструктури можуть входити різноманітні пристрої, такі як принтери, сканери, копіри та інші пристрої, що допомагають користувачам працювати з даними та виводити результати своєї роботи на папері або іншому носії [7];

– мережеві комутатори та маршрутизатори: це пристрої, які дозволяють комп'ютерам та іншим пристроям підключатися до мережі та взаємодіяти між собою. Мережеві комутатори допомагають передавати дані між пристроями в межах локальної мережі, тоді як маршрутизатори дозволяють передавати дані між різними мережами;

– сховища даних: це пристрої, які дозволяють зберігати великі обсяги даних на дисках та інших носіях. Сховища даних можуть бути звичайними жорсткими дисками, мережевими сховищами, флеш-накопичувачами та іншими пристроями, які дозволяють зберігати та забезпечувати доступ до даних в IT-інфраструктурі [8];

– безперебійне живлення та системи охолодження: це компоненти, які допомагають забезпечити неперервну роботу всіх інших компонентів апаратної інфраструктури. Безперебійні живлення забезпечують постачання електроенергії в разі відключення головного джерела живлення, тоді як системи охолодження допомагають уникнути перегріву компонентів та забезпечити оптимальну температуру в приміщенні, де розташована IT-інфраструктура;

– системи забезпечення безпеки, які дозволяють забезпечити безпеку даних та пристроїв в мережі. До таких систем можуть входити антивірусні програми, файрволи, системи ідентифікації користувачів та інші рішення для забезпечення безпеки.

Апаратне забезпечення IT-інфраструктури постійно еволюціонує, адаптуючись до змін в технологічному середовищі та вимог користувачів. Тому важливо регулярно оновлювати компоненти та пристосовувати їх до нових вимог та можливостей.

Програмне забезпечення

Програмне забезпечення IT-інфраструктури - це набір програмних продуктів та рішень, які використовуються для забезпечення функціональності та продуктивності IT-системи [9]. Програмне забезпечення

може бути розгорнуте на різних рівнях IT-інфраструктури та включати різні категорії програмних продуктів.

Основні категорії програмного забезпечення, які використовуються в IT-інфраструктурі [10]:

– операційні системи: це програми, які керують апаратним забезпеченням та забезпечують доступ до ресурсів комп'ютера. Операційні системи включають в себе відповідні системні сервіси та драйвери, які забезпечують взаємодію між апаратним та програмним забезпеченням;

– системи управління базами даних: програмні продукти, які використовуються для зберігання, організації та управління даними. Вони забезпечують доступ до даних для додатків та користувачів, а також забезпечують захист та резервне копіювання даних;

– серверні програмні продукти: включають в себе різні сервіси, які виконуються на серверах та забезпечують функціональність мережевих додатків та служб. Серверні програмні продукти можуть включати веб-сервери, поштові сервери, сервери баз даних та інші сервіси;

– клієнтські програмні продукти: це продукти, що використовуються для доступу до ресурсів та послуг IT-інфраструктури. Клієнтські програмні продукти можуть бути веб-браузерами, поштовими клієнтами, програмами для роботи з документами, програмами для роботи з базами даних та іншими програмами [11];

– системи віддаленого управління та моніторингу. Використовуються для віддаленого керування та моніторингу комп'ютерами та іншими пристроями в мережі. Вони дозволяють адміністраторам віддалено керувати робочими станціями та серверами, відстежувати стан мережі та здійснювати резервне копіювання та відновлення даних;

– антивірусне програмне забезпечення: це програми, що використовуються для захисту IT-інфраструктури від вірусів, шкідливого програмного забезпечення та інших загроз безпеці. Антивірусні програми

забезпечують захист на різних рівнях IT-інфраструктури - від робочих станцій та серверів до мережевих пристроїв та хмарних сервісів;

– програми для забезпечення безпеки мережі використовуються для забезпечення безпеки мережевих пристроїв та комунікацій між ними. Вони дозволяють адміністраторам моніторити мережу, виявляти та блокувати загрози безпеки, забезпечувати безпеку доступу та захищати дані від несанкціонованого доступу.

Враховуючи те, що програмне забезпечення IT-інфраструктури має бути добре скоординоване, організація повинна мати стратегію, яка визначає правила вибору програмного забезпечення. Вибір програмного забезпечення залежить від потреб організації, її бізнес-процесів та конкретних задач, що ставляться перед системою.

Мережі IT-інфраструктури

Мережі IT-інфраструктури - це системи, що використовуються для забезпечення комунікації між різними пристроями та обладнанням, що входять до складу IT-інфраструктури. Компоненти мереж IT-інфраструктури включають [12]:

– сервери: забезпечують доступ до різних сервісів та додатків, таких як електронна пошта, бази даних, веб-сервери, файлові сервери тощо. Сервери можуть бути фізичними або віртуальними, залежно від вимог до масштабування інфраструктури;

– комутатори: це пристрої, які забезпечують з'єднання між різними комп'ютерами та пристроями в мережі. Комутатори дозволяють пересилати дані між пристроями, використовуючи різноманітні протоколи передачі даних, такі як Ethernet. Комутатори можуть бути керовані та некеровані залежно від складності мережі;

– роутери: це мережеві пристрої, які забезпечують маршрутизацію даних та з'єднання між різними мережами. Роутери дозволяють передавати дані між мережами з використанням різних протоколів маршрутизації, таких

як OSPF, BGP, RIP. Роутери також забезпечують безпеку мережі та захист від зовнішніх загроз;

– брандмауери: це пристрої, які забезпечують захист мережі від несанкціонованого доступу та атак. Брандмауери блокують потенційно небезпечний трафік та запобігають розповсюдженню шкідливого програмного забезпечення. Брандмауери можуть бути апаратними або програмними;

– мережеві кабелі: це фізичні засоби з'єднання різних пристроїв в мережі. Вони дозволяють передавати дані та інформацію між різними пристроями в мережі. Мережеві кабелі можуть бути виконані з різних матеріалів, таких як мідь, волоконно-оптичний кабель та інші. Різні типи мережевих кабелів мають різні швидкості передачі даних та максимальні відстані передачі;

– принтери та інше периферійне обладнання: це компоненти мережі IT-інфраструктури, які забезпечують доступ до фізичних документів та інформації. Принтери, сканери, копіювальні апарати та інше периферійне обладнання можуть бути підключені до мережі IT-інфраструктури для забезпечення доступу до цих пристроїв з різних пристроїв, що входять до складу мережі [13];

– програмне забезпечення для моніторингу та керування мережею: це компонент мережі IT-інфраструктури, який дозволяє моніторити та керувати різними пристроями та обладнанням в мережі. Це може бути програмне забезпечення для моніторингу мережі, забезпечення безпеки мережі, програмне забезпечення для керування сервісами та додатками тощо;

– додаткове обладнання для забезпечення безпеки та збереження даних: це компоненти мережі IT-інфраструктури, які забезпечують безпеку мережі та збереження даних. До таких компонентів можуть належати резервні копії даних, системи збереження даних, системи відновлення даних, антивірусні програми, мережеві інтерфейси для збереження даних та інше.

– активне та пасивне обладнання мережі: це обладнання мережі ІТ-інфраструктури, яке забезпечує різні функції. Активне обладнання мережі, таке як сервери, комутатори та роутери, потребує джерела живлення та виконує різні обчислювальні функції. Пасивне обладнання мережі, таке як мережеві кабелі, зазвичай не потребує додаткового живлення та не виконує обчислювальних функцій;

– власний чи орендований простір для розміщення обладнання та забезпечення його захисту. Мережеве обладнання потребує відповідного простору для його розміщення та забезпечення його захисту від фізичних та інших загроз. Це може бути власний простір в будівлі компанії, дата-центр, орендоване приміщення в іншій компанії тощо.

Усі компоненти мереж ІТ-інфраструктури повинні бути добре налаштовані та підтримуватися в робочому стані. Це може забезпечуватися за допомогою різноманітних інструментів та програмного забезпечення для моніторингу та керування мережею. Підтримка та регулярне оновлення мережі ІТ-інфраструктури допомагає забезпечити її безпеку та ефективну роботу.

Одним з головних завдань ІТ-інфраструктури є забезпечення ефективного та безпечного використання технологій та інформації в організації. Нижче розглянуто деякі з ключових завдань ІТ-інфраструктури:

– забезпечення надійності та стійкості мережі. Повинна забезпечувати надійну та стійку мережу, щоб забезпечити безперебійну роботу всіх систем та пристроїв, підключених до мережі;

– захист інформації. має забезпечувати захист конфіденційної та важливої інформації від несанкціонованого доступу, зловживань та втрати даних [14];

– підтримка безперебійної роботи. Повинна забезпечувати безперебійну роботу всіх систем та пристроїв, підключених до мережі, та забезпечувати швидке відновлення роботи після можливих збоїв або аварій;

– управління ресурсами. Має забезпечувати ефективне використання ресурсів та оптимізацію роботи систем та пристроїв, підключених до мережі;

– підтримка різноманітності. Повинна забезпечувати підтримку різноманітних систем та пристроїв, підключених до мережі, та забезпечувати їх сумісність;

– підтримка мобільності. Має забезпечувати підтримку різноманітних мобільних пристроїв та забезпечувати їх безперебійний доступ до ресурсів та інформації;

– підтримка розподіленості. Повинна забезпечувати можливість роботи з даними та ресурсами, які знаходяться на віддалених серверах та комп'ютерах;

– масштабованість. Інфраструктура повинна бути готовою до масштабування, щоб забезпечити розвиток та розширення організації;

– підтримка віртуалізації. Повинна забезпечувати підтримку віртуалізації, що дозволяє знизити витрати на обладнання та збільшити ефективність використання ресурсів;

– забезпечення комунікації та співпраці. Має забезпечувати можливість спілкування та співпраці між користувачами та підрозділами організації, що сприяє покращенню ефективності та продуктивності;

– моніторинг та управління. Повинна забезпечувати можливість моніторингу та управління всіма системами та пристроями, що знаходяться в мережі, для забезпечення ефективності та стійкості мережі [15].

Загалом, завдання ІТ-інфраструктури полягає у забезпеченні ефективного та безпечного використання технологій та інформації в організації, що дозволяє забезпечити більш ефективну та продуктивну роботу організації та збереження даних та ресурсів.

1.2 Актори та функції системи

Для кращого розуміння та опису функціональності створеної системи важливо мати детальний опис того, як система буде використовуватися користувачами. В цьому контексті діаграма прецедентів є потужним інструментом, який дозволяє відобразити відносини між акторами та прецедентами.

Прецедент може бути описаний як можливість моделювання системи, що визначає, як користувач може взаємодіяти з системою, щоб отримати певний результат. Кожен прецедент є окремим сервісом, що надається системою, і відповідає за один з варіантів використання системи.

Діаграма прецедентів дозволяє описати типові способи взаємодії користувачів з системою, відображає взаємодії між користувачами та системою, а також дозволяє визначити зовнішні вимоги до системи. Діаграма прецедентів дозволяє описати поведінку системи з точки зору користувачів і уточнити вимоги до системи та забезпечити більш точний та конкретний опис її функціональності.

Узагалі, використання діаграм прецедентів є важливим етапом при проектуванні будь-якої системи, що дозволяє визначити функціональні вимоги до неї та забезпечити ефективну взаємодію між користувачами та системою.

Таблиця 1.1 – Функціональні вимоги до додатку

Вимоги	**Опис**
REQ-1	Можливість здійснення реєстрації та авторизації користувачів у системі
REQ-2	Можливість ведення журналу подій, що відбулися в системі
REQ-3	Можливість додавати та редагувати інформацію про облікові записи користувачів

REQ-4	Можливість додати та редагувати інформацію різноманітних тем
REQ-5	Можливість тренування нейронних мереж під різні теми для служб підтримки користувачів
REQ-6	Можливість автоматизованої підтримки користувачів по різним темам за допомогою тренованих нейронних мереж
REQ-7	Система повинна дозволяти користувачеві формувати різноманітну звітність

Таблиця 1.2 – Нефункціональні вимоги до додатку

Вимоги	**Опис**
REQ-8	Додаток повинен мати простий дизайн та зручну навігації
REQ-9	Поля повинні бути не порожніми, унікальними відносно вже існуючих записів

Таблиця 1.3 – Актори та цілі додатку

Актори	**Цілі**
Адміністратор	Мета адміністратора полягає у роботі із обліковими записами
Користувач	Мета користувача полягає у роботі із системою
База даних	Мета бази даних полягає у зберігання інформації

Таблиця 1.4 – Опис варіантів використання додатку

Варіант використання	**Ім’я**	**Опис**
UC1	Реєстрація в системі	Дозволяє користувачеві зареєструватися в системі

UC2	Автентифікація в системі	Дозволяє користувачеві пройти процедуру автентифікації в системі
UC3	Вивід каталогу користувачів	Дозволяє користувачеві з правами системного адміністратора переглянути каталог всіх користувачів системи
UC4	Додати користувача	Дозволяє користувачеві з правами системного адміністратора додати нового користувача в систему
UC5	Редагувати користувача	Дозволяє користувачеві з правами системного адміністратора редагувати інформацію про вибраного користувача зі списку
UC6	Видалити користувача	Дозволяє користувачеві з правами системного адміністратора видалити вибраного користувача зі списку
UC7	Вивід каталогу тем	Дозволяє користувачеві вивести каталог всіх тем, для служби підтримки
UC8	Додати тему	Дозволяє користувачеві додати нову тему для служби підтримки
UC9	Редагувати тему	Дозволяє користувачеві редагувати інформацію вибраної теми
UC10	Видалити тему	Дозволяє користувачеві видалити вибрану тему
UC11	Провести навчання нейронної мережі	Дозволяє користувачу провести навчання нейромережі
UC12	Видалити натреновану мережу	Дозволяє користувачу видалити нейронну мережу

UC13	Надання послуг	Дозволяє користувачеві получити відповідь на задані запитання для вибраної теми
UC14	Формування статистики по темі	Дозволяє користувачеві формувати статистику по вибраній темі
UC15	Формування статистики по даті	Дозволяє користувачу формувати статистику за вибраний період часу
UC16	Вивід системних подій	Дозволяє вивести всі події, які відбулися в системі

З визначених вимог тепер можна виставити повний опис вимог із сценаріями, що будуть вхідними даними для візуального моделювання мовою UML.

UC1 Реєстрація в системі

Актор: адміністратор/користувач.

Ціль актора: пройти реєстрацію у системі.

Задіяний актор: база даних.

Передумова: користувач ввівши всі необхідні дані натискає кнопку «Зберегти».

Післяумова: система проводить реєстрацію клієнта та відображає форму для входу в систему користувача.

UC2 Автентифікація в системі

Актор: адміністратор/користувач.

Ціль актора: пройти автентифікацію у системі.

Задіяний актор: база даних.

Передумова: користувач ввівши всі необхідні дані натискає кнопку «Зберегти».

Післяумова: система проводить автентифікацію користувача та відображає головне вікно програми.

UC3 Вивід каталогу користувачів

Актор: адміністратор.

Ціль актора: вивести інформацію про всіх користувачів системи.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Користувачі».

Післяумова: система відображає екран зі списком всіх зареєстрованих користувачів в системі.

UC4 Додати користувача

Актор: адміністратор.

Ціль актора: додати нового користувача.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про новий обліковий запис натискає на кнопку «Додати».

Післяумова: система додає інформацію про новий обліковий запис та відображає екран зі списком всіх користувачів.

UC5 Редагувати користувача

Актор: адміністратор.

Ціль актора: редагувати інформацію про вибраного користувача.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку користувача.

Післяумова: система відображає екран для редагування інформації про вибраного користувача.

UC6 Видалити користувача

Актор: адміністратор.

Ціль актора: видалити користувача із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку користувача.

Післяумова: система відображає екран з можливістю видалення даних про користувача.

UC7 Вивід каталогу тем

Актор: адміністратор/користувач.

Ціль актора: вивести інформацію про всі теми.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Теми».

Післяумова: система відображає екран для виведення списку всіх тем.

UC8 Додати тему

Актор: адміністратор/користувач.

Ціль актора: додати нову тему.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про тему натискає на кнопку «Додати».

Післяумова: система додає нову тему та відображає список всіх тем.

UC9 Редагувати тему

Актор: адміністратор/користувач.

Ціль актора: редагувати вибрану із списку тему.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку тему.

Післяумова: система відображає екран для редагування інформації про вибрану із списку тему.

UC10 Видалити тему

Актор: адміністратор/користувач.

Ціль актора: видалити вибрану тему із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку тему.

Післяумова: система відображає екран з можливістю видалення даних вибраної тему.

UC11 Провести навчання нейронної мережі

Актор: адміністратор/користувач.

Ціль актора: провести навчання нової нейронної мережі.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Навчання нейронної мережі».

Післяумова: система відображає екран з можливістю навчання нової нейронної мережі.

UC12 Видалити натреновану мережу

Актор: адміністратор/користувач.

Ціль актора: видалити інформацію про натреновану нейронну мережу.

Задіяний актор: база даних.

Передумова: користувач у правій частині вікна у необхідному записі про нейронну мережу натискає кнопку «Видалити».

Післяумова: система видаляє вибрану нейронну мережу та всю інформацію про неї.

UC13 Надання послуг

Актор: адміністратор/користувач.

Ціль актора: поставити запитання нейронній мережі на вибрану тему.

Задіяний актор: база даних.

Передумова: користувач переходить по пункту меню «Надання послуг».

Післяумова: система відображає екран для можливості формування запитань до нейромережі.

UC14 Формування статистики по темі

Актор: адміністратор/користувач.

Ціль актора: формувати статистику по вибраній із списку темі.

Задіяний актор: база даних.

Передумова: користувач переходить по меню програми «Статистика» –> «По темі запитань».

Післяумова: система відображає екран з можливістю формування статистики по вибраній темі.

UC15 Формування статистики по даті

Актор: адміністратор/користувач.

Ціль актора: формувати статистику за вибраний період часу.

Задіяний актор: база даних.

Передумова: користувач переходить по меню програми «Статистика» –> «По даті».

Післяумова: система відображає екран з можливістю формування статистики за вибраний період часу.

UC16 Вивід системних подій

Актор: адміністратор.

Ціль актора: вивести список всіх подій в системі.

Задіяний актор: база даних.

Передумова: користувач вибирає пункт меню «Системний журнал».

Післяумова: система відображає екран із списком всіх системних подій.

На основі отриманих даних було побудовано use-case діаграми прецедентів для ролі системного адміністратор та користувача, що зображені на рис. 1.2 та рис. 1.3 відповідно.

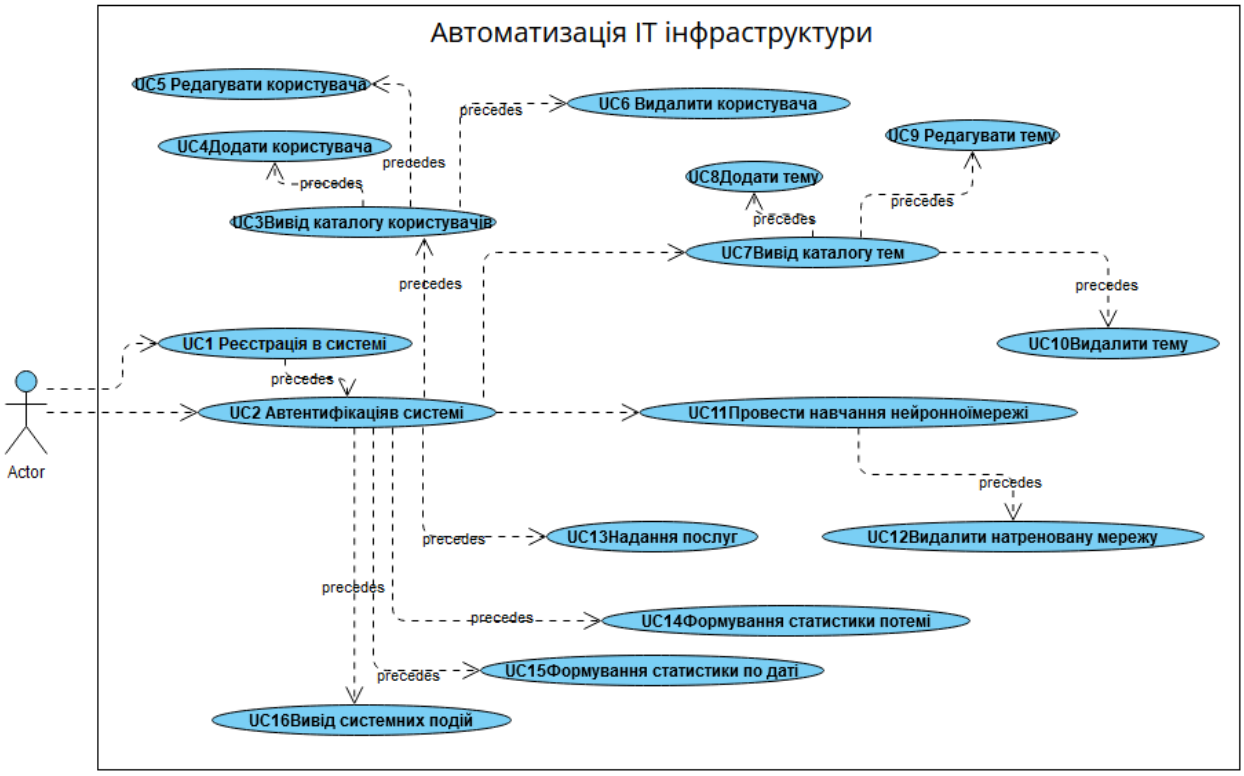


Рисунок 1.2 – Діаграма use-case для ролі «системний адміністратор»

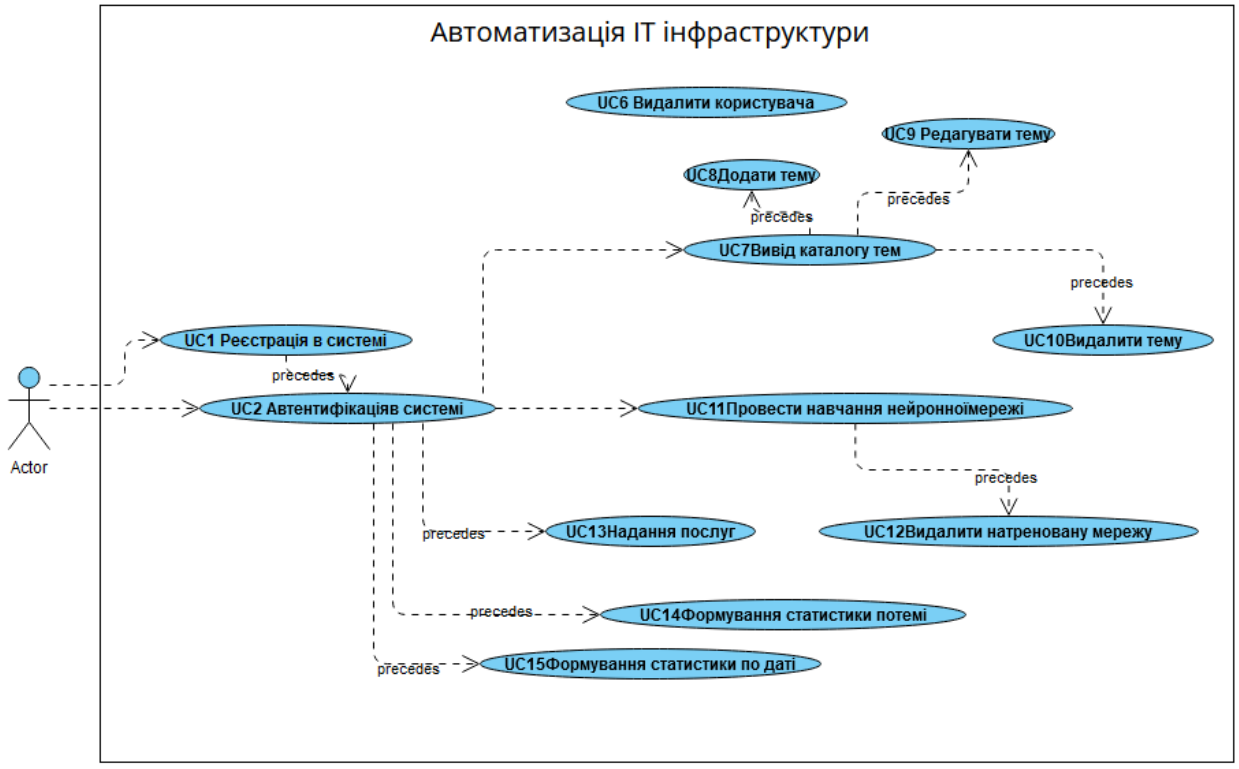


Рисунок 1.3 – Діаграма use-case для ролі «користувач»

1.3 Аналіз аналогічних програмних рішень

З метою автоматизації IT інфраструктури підприємства за допомогою машинного навчання та нейронних мереж на ринку існує кілька програмних рішень. Для аналізу аналогічних програмних рішень, буде розглянуто такі варіанти: Zendesk, HappyFox, Freshdesk.

Платформа Zendesk

Zendesk - це популярна хмарна платформа, яка дозволяє компаніям надавати ефективну підтримку клієнтів [16]. Розроблена була Zendesk Inc. в 2007 році в Сан-Франциско, США. Сьогодні Zendesk має офіси по всьому світу та обслуговує більше 200 тисяч компаній у різних галузях.

Інтерфейс для роботи з платформою Zendesk зображено на рис. 1.4.

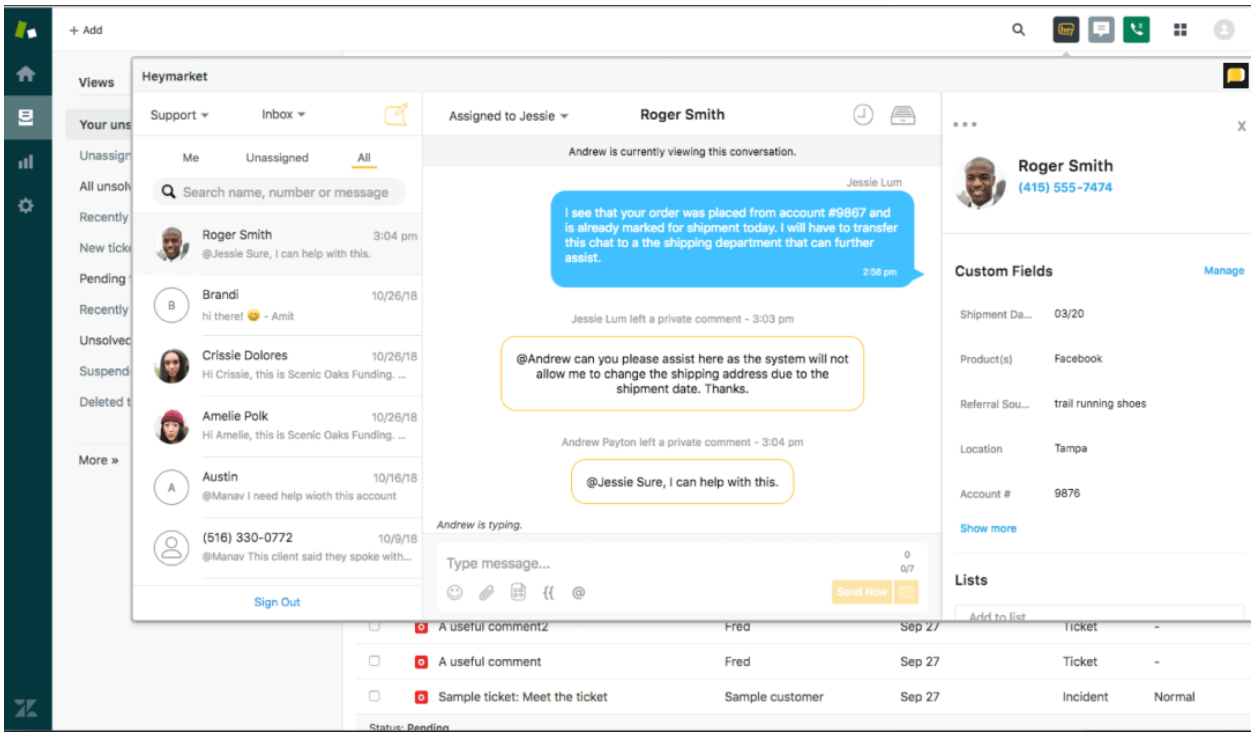


Рисунок 1.4 – Інтерфейс користувача Zendesk

Основними функціями Zendesk є :

– автоматизація обробки запитів користувачів;
– моніторинг задач підтримки;
– відстеження проблем та спільна робота над ними в команді;

– забезпечення зручного та ефективного спілкування з клієнтами.

Zendesk також має інструменти для аналізу даних, збирання зворотного зв'язку від клієнтів, створення звітів та багато іншого.

Переваги Zendesk [17]:

– гнучкість. Надає можливість налаштування системи підтримки клієнтів під вимоги конкретної компанії;

– широкий функціонал. Має різноманітні інструменти для підтримки клієнтів, такі як Help Desk, Live Chat, Knowledge Base та інші;

– ефективність. Допомагає зменшити час очікування клієнтів на відповіді, забезпечуючи швидке та ефективне вирішення проблем;

– інтеграція з іншими системами. Дозволяє легко інтегруватися з іншими системами, наприклад, з CRM-системами, соціальними мережами, електронною поштою та іншими;

– зручний та простий інтерфейс. Має інтуїтивно зрозумілий інтерфейс, який дозволяє швидко орієнтуватися в системі та працювати з нею.

Недоліки Zendesk:

– високі витрати. Вартість використання Zendesk може бути значною для малих та середніх компаній;

– залежність від Інтернету. Потребує Інтернет-підключення для роботи, що може створити проблеми у випадку відсутності Інтернету;

– неінтуїтивність певних функцій. Деякі функції Zendesk можуть бути неінтуїтивними, що може вимагати додаткового навчання співробітників;

– обмеження в налаштуванні деяких функцій. Деякі функції Zendesk можуть бути обмеженими в налаштуванні, що може бути проблемою для деяких компаній.

У загальному, Zendesk є потужним інструментом для підтримки клієнтів, який надає широкий функціонал та гнучкість налаштування. Однак, вартість використання може бути дуже високою для малих та середніх

компаній, а також можуть виникати проблеми зі залежністю від Інтернету та неінтуїтивністю деяких функцій. Також, обмежена можливість налаштування вигляду та дизайну може бути проблемою для деяких компаній. Однак, загалом Zendesk є високоякісним та ефективним рішенням для компаній, які потребують потужного інструменту для підтримки клієнтів.

Платформа HappyFox

HappyFox - це популярна хмарна платформа для підтримки клієнтів, яка дозволяє компаніям надавати ефективну підтримку своїм клієнтам (рис. 1.5). Розроблена була HappyFox Inc. в 2011 році в Каліфорнії, США. Сьогодні HappyFox має клієнтів по всьому світу та обслуговує більше 12 000 компаній.

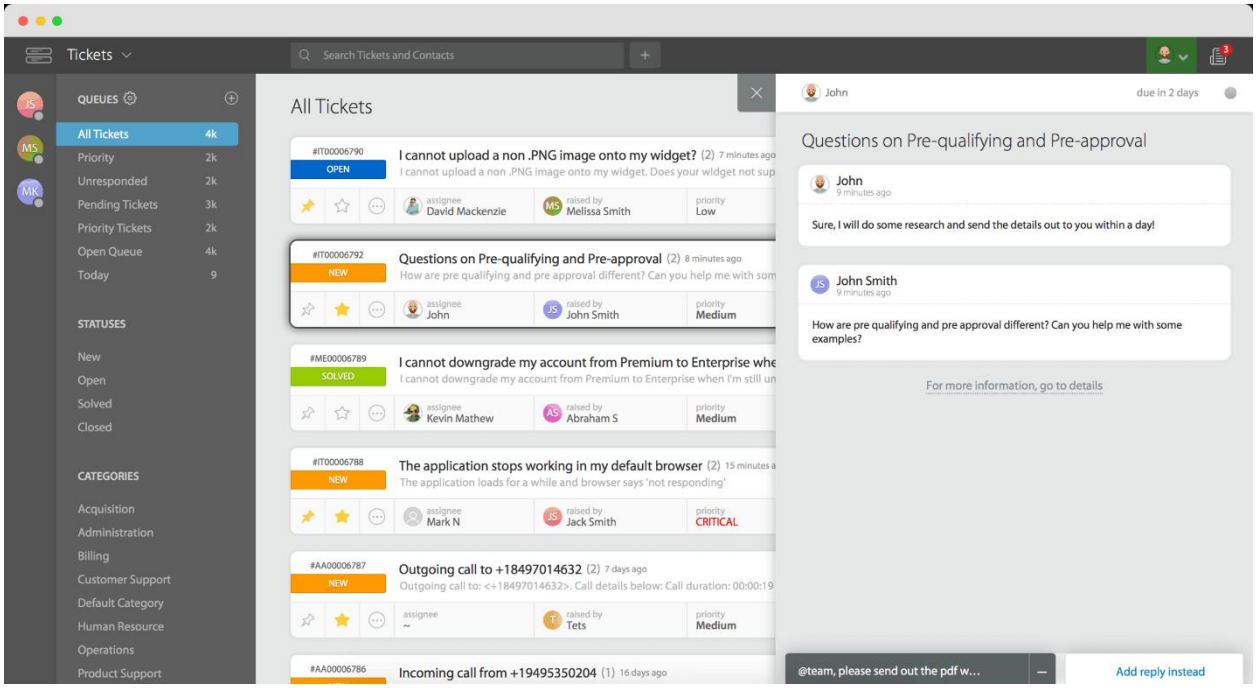


Рисунок 1.5 – Інтерфейс користувача платформи HappyFox

Основна функціональність HappyFox включає:

– керування запитами користувачів. Створення, відстеження та вирішення запитів користувачів за допомогою Help Desk;

– Live Chat. Спілкування з клієнтами в режимі реального часу за допомогою інструменту Live Chat;

– Knowledge Base. Зберігання та організація інформації про продукти та послуги компанії, яка дозволяє клієнтам знайти відповіді на свої запитання самостійно;

– Workflows. Автоматизація бізнес-процесів та задач підтримки клієнтів, що дозволяє збільшити ефективність роботи та зменшити навантаження на співробітників;

– Insights. Аналіз даних про підтримку клієнтів для отримання корисної інформації про те, як можна покращити якість обслуговування;

– Self Service. Створення та розміщення на сайті компанії бази знань для допомоги клієнтам знайти відповіді на свої запитання самостійно.

– інтеграція з іншими системами. Легка інтеграція з іншими системами, наприклад, з CRM-системами, соціальними мережами, електронною поштою та іншими.

Деякі переваги HappyFox включають [18]:

– широкий спектр функціональності та інтеграцій, який дозволяє підприємствам налаштувати систему підтримки клієнтів за своїми потребами;

– легкість використання та інтуїтивний інтерфейс, що дозволяє співробітникам швидко оволодіти системою;

– ефективність та швидкість обробки запитів користувачів, що дозволяє компаніям надавати високоякісну підтримку;

– можливість налаштування автоматичних відповідей та розсилки повідомлень, що дозволяє зменшити навантаження на співробітників та забезпечити більш ефективну роботу;

– високий рівень безпеки даних та конфіденційності, що забезпечує захист важливої інформації компанії та її клієнтів.

Недоліки HappyFox включають [19]:

– вартість. Для деяких компаній вартість використання HappyFox може бути високою;

– обмеження під час використання безкоштовної версії. Безкоштовна версія HappyFox має обмеження щодо кількості користувачів та функцій;

– обмежена можливість налаштування дизайну та вигляду. Деякі компанії можуть вимагати більшої гнучкості щодо налаштування вигляду та дизайну системи підтримки клієнтів;

– залежність від Інтернету. HappyFox - це хмарний сервіс, тому компанії повинні мати постійний доступ до Інтернету для користування ним.

Отже, HappyFox - це хмарна платформа для підтримки клієнтів, що дозволяє компаніям забезпечувати ефективну підтримку своїх клієнтів. Основні переваги HappyFox полягають у широкому спектрі функціональності, легкості використання та ефективності вирішення запитів користувачів. Однак, вартість та обмеження під час використання безкоштовної версії можуть бути недоліками для деяких компаній.

Платформа Freshdesk

Freshdesk - це хмарна платформа для підтримки клієнтів, розроблена компанією Freshworks. Freshdesk надає інструменти для створення, відстеження та вирішення запитів користувачів.

Інтерфейс для роботи користувача із платформою зображено на рис. 1.6.

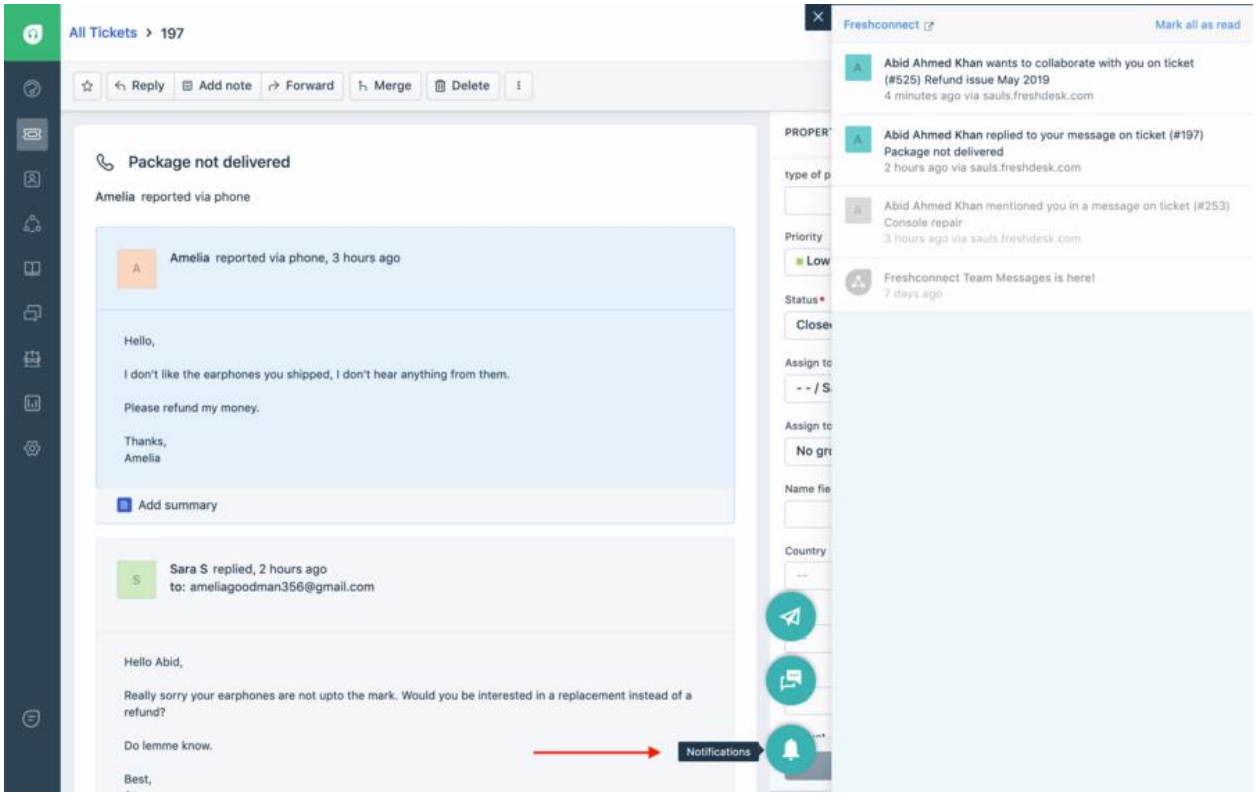


Рисунок 1.6 – Інтерфейс користувача для роботи із платформою Freshworks

Основні функції Freshdesk включають [20]:

– керування запитами користувачів. створення, відстеження та вирішення запитів користувачів за допомогою Help Desk;

– спілкування з клієнтами в режимі реального часу за допомогою інструменту Live Chat;

– зберігання та організація інформації про продукти та послуги компанії, яка дозволяє клієнтам знайти відповіді на свої запитання самостійно;

– автоматизація бізнес-процесів та задач підтримки клієнтів, що дозволяє збільшити ефективність роботи та зменшити навантаження на співробітників;

– аналіз даних про підтримку клієнтів для отримання корисної інформації про те, як можна покращити якість обслуговування;

– створення та розміщення на сайті компанії бази знань для допомоги клієнтам знайти відповіді на свої запитання самостійно;

– інтеграція з іншими системами. Легка інтеграція з іншими системами, наприклад, з CRM-системами, соціальними мережами, електронною поштою та іншими.

Ці функції дозволяють компаніям надавати високоякісну підтримку клієнтів та забезпечувати ефективну роботу з запитами користувачів.

Деякі переваги Freshdesk включають [21]:

– широкий спектр функціональності та інтеграцій, який дозволяє підприємствам налаштувати систему підтримки клієнтів за своїми потребами;

– легкість використання та інтуїтивний інтерфейс, що дозволяє співробітникам швидко оволодіти системою;

– ефективність та швидкість обробки запитів користувачів, що дозволяє компаніям надавати високоякісну підтримку;

– можливість налаштування автоматичних відповідей та розсилки повідомлень, що дозволяє зменшити навантаження на співробітників та забезпечити більш ефективну роботу;

– високий рівень безпеки даних та конфіденційності, що забезпечує захист важливої інформації компанії та її клієнтів.

Деякі недоліки Freshdesk включають:

– вартість. Для деяких компаній вартість використання Freshdesk може бути високою;

– обмеження під час використання безкоштовної версії. Безкоштовна версія Freshdesk має обмеження щодо кількості користувачів та функцій;

– обмежена можливість налаштування дизайну та вигляду. Деякі компанії можуть вимагати більшої гнучкості щодо налаштування вигляду та дизайну системи підтримки клієнтів;

– залежність від Інтернету. Freshdesk - це хмарний сервіс, тому компанії повинні мати постійний доступ до Інтернету для користування ним.

Отже, Freshdesk - це хмарна платформа для підтримки клієнтів, розроблена компанією Freshworks. Основні переваги Freshdesk полягають у широкому спектрі функціональності, легкості використання та ефективності вирішення запитів користувачів. Проте вартість та обмеження під час використання безкоштовної версії можуть бути недоліками для деяких компаній.

Для більш кращого розуміння функціоналу розглянутих додатків, необхідно провести їх порівняння по загальним функціональним характеристикам у вигляді таблиці (табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз систем

Характеристика	*Zendesk*	*HappyFox*	*Freshdesk*
Розробник	Zendesk Inc.	HappyFox Inc.	Freshworks Inc.
Легкість використання	Легкий інтерфейс	Легкий інтерфейс	Легкий інтерфейс
Ціна	Висока вартість, але є безкоштовна версія з обмеженнями	Вартість на середньому рівні, є безкоштовна версія з обмеженнями	Вартість на середньому рівні, є безкоштовна версія з обмеженнями
Інтеграція	Широкий вибір	Широкий вибір	Широкий вибір
Підтримка клієнтів	Швидке та ефективне вирішення запитів	Швидке та ефективне вирішення запитів	Швидке та ефективне вирішення запитів
Надійність	Висока стабільність та надійність	Висока стабільність та надійність	Висока стабільність та надійність

Безпека	Високий рівень безпеки даних та конфіденційності	Високий рівень безпеки даних та конфіденційності	Високий рівень безпеки даних та конфіденційності
Аналітика	Має вбудований аналітичний звіт	Має вбудований аналітичний звіт	Має вбудований аналітичний звіт

У результаті аналізу Zendesk, HappyFox та Freshdesk можна зробити висновок, що всі три платформи мають багатий функціонал та можуть задовольнити потреби різних компаній, що шукають інструменти для підтримки клієнтів.

Жодна з перерахованих платформ не має функціональності для роботи з нейронними мережами для аналізу даних служби підтримки користувачів та передбачення типу проблеми та способу її вирішення.

Таким чином, необхідно розглянути можливість на розробку власної системи на базі відкритих бібліотек для машинного навчання.

1.4 Постановка задачі

Із проведено аналізу предметної області було прийнято рішення розробити програмне забезпечення для тренування моделі на історичних даних служби підтримки користувачів з метою передбачення типу проблеми та способу її вирішення. Навчена модель повинна повертати результат передбачення типу проблеми та способу її вирішення.

Даний функціонал може бути використаний для автоматизації обробки запитів користувачів, наприклад, для автоматичної класифікації та маршрутизації запитів до відповідних служб підтримки. Це може знизити час очікування користувачів та підвищити ефективність роботи служби підтримки.

Отже, розробка системи припускає:

– розробку схеми БД;

– розробку та реалізацію системи, що включає три основні модулі:

1) модуль вводу/редагування інформації в БД;

2) модуль захисту інформації;

3) модуль навчання нейронної мережі з метою передбачення типу проблеми користувачів та способу її вирішення.

Реалізація додатка виконується з використанням технологій .NET та мови програмування C#.

Розроблена система повинна містити:

Сутності:

– Користувачі системи (інформація про облікові записи користувачів): прізвище, ім’я, ім’я облікового запису, пароль, загальна інформація, ідентифікатор ролі;

– тема (інформація про тему для навчання): назва та опис;

– нейронна мережа (інформація про нейронну мережу): назва, шлях до файлу тренованої моделі, ідентифікатор теми;

– історія відповідей (інформація про історію відповідей клієнтів): запитання, відповідь, рейтинг, ідентифікатор користувача, ідентифікатор теми, відповідь клієнта та дата-час запитання;

– події (інформація про події в системі) ідентифікатор користувача, опис події, дата/час події.

Введення:

– вводяться та редагуються: користувачі системи, теми;

– тренуються нейронні мережі для підтримки користувачів з метою передбачення типу проблеми та способу її вирішення;

– фіксуються рейтинги, щодо відповіді клієнтів на вирішення питань;

– фіксуються записи подій, що відбулися в системі.

Статистика:

– по темі запитань: виводиться детальна статистика по вибраній темі запитань користувачів, а також підраховується середній рейтинг відповідей;

– по дані: виводиться детальна статистика за вибраний період часу.

1.5 Висновок

У даному розділі було проведено аналіз поняття, завдань та особливостей IT інфраструктури компанії. Було визначено акторів та функції системи, які повинні бути розроблені для забезпечення ефективної роботи системи.

В рамках аналізу аналогічних програмних рішень було розглянуто три популярні платформи для підтримки клієнтів - Zendesk, HappyFox та Freshdesk. Було проведено порівняння основних характеристик та функціональності цих платформ. У результаті аналізу було визначено, що всі три платформи мають широкий спектр функціональності та можуть задовольнити потреби різних компаній, які шукають інструменти для підтримки клієнтів.

Проте, жодна з перерахованих платформ не надає функціональності для автоматичного аналізу даних служби підтримки користувачів та передбачення типу проблеми та способу її вирішення з використанням нейронних мереж. Таким чином є перспективною розробка власної системи на базі відкритих бібліотек для машинного навчання.

Отже, у результаті проведеного аналізу можна зробити висновок, що для ефективної підтримки клієнтів важливо мати систему, яка забезпечить широкий спектр функціональності та зможе задовольнити потреби різних компаній.

При розробці системи підтримки прийняття рішень при експлуатації IT інфраструктури підприємства, необхідно враховувати потреби користувачів та

визначати акторів та їх функції. Для цього можна використовувати методологію проектування вимог до програмного забезпечення, що забезпечить ефективний процес розробки та відповідність функціональності системи потребам користувачів.

2 МЕТОДИ І МОДЕЛІ ПРОГНОЗУВАННЯ ДАНИХ СЛУЖБИ ПІДТРИМКИ КОРИСТУВАЧІВ

2.1 Аналітичний огляд існуючих методів розв’язання задачі

Зазвичай, коли вибирається архітектура мережі для вирішення конкретної задачі, випробовуються різні конфігурації з різною кількістю елементів. У випадку, коли задача полягає у прогнозуванні, що є окремим випадком задачі регресії, для вирішення цієї задачі можуть бути використані такі типи нейронних мереж, як багатошаровий персептрон (MLP), радіально-базисна мережа (RBN), узагальнено-регресійна мережа (GRNN) та мережа Воль Ельмана. Однак, для порівняння ефективності можна обмежитися тільки трьома типами: MLP, RBN та GRNN.

2.1.1 Багатошаровий персептрон (MLP)

Багатошарові персептрони є ефективними для вирішення тих самих завдань, що й одношарові персептрони, проте вони мають значно більшу обчислювальну потужність [22]. Ця здатність дозволяє їм більш точно описувати багатовимірні залежності з великим ступенем нелінійності та високим рівнем перехресного та групового впливу вхідних змінних на вихідні.

У контексті автоматизації IT інфраструктури підприємства за допомогою машинного навчання та нейронних мереж, багатошаровий персептрон може бути використаний для багатьох завдань. Наприклад, він може бути використаний для прогнозування відмов комп'ютерного обладнання, визначення потреб у ресурсах, виявлення аномальної поведінки в мережі, автоматичної класифікації та маршрутизації запитів до відповідних служб підтримки та інші.

При необхідності використання мереж складнішої структури додаються нові приховані шари або нарощується кількість нейронів у прихованих шарах (рис. 2.1).

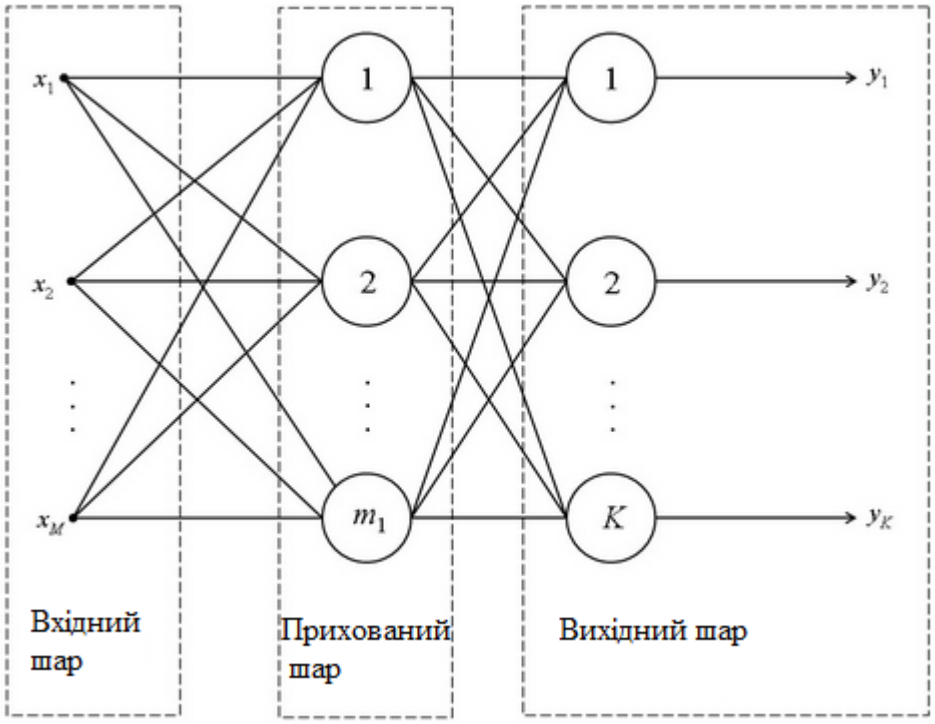


Рисунок 2.1 – Схема багатошарової штучної нейронної мережі

Кількість вагових коефіцієнтів, що налаштовуються в процесі навчання багатошарового персептрону з L прихованими шарами m_l нейронів у кожному, розраховується наступним чином:

$$N_w = (M+1)m_l + \sum_{l=2}^{L}(m_{l-1}+1)m_l + (m_L+1)K \qquad (2.1)$$

Для кожного нейрона мережі, крім синаптичних зв'язків з елементами вхідного вектора, налаштовується зв'язок з фіктивним одиничним входом (коефіцієнт зміщення) [23].

При вирішенні завдань з використанням багатошарових персептронів важливо правильно вибрати структуру нейронної мережі. Для цього можна використовувати правило 2-5, яке говорить про те, що кількість вагових коефіцієнтів, які налаштовуються під час навчання, повинна бути в 2-5 разів меншою, ніж кількість навчальних прикладів [24]. Якщо це співвідношення менше 2, то мережа може втратити здатність до узагальнення навчальної інформації, а якщо менше 1, то мережа просто запам'ятовує відповіді для

кожного навчального прикладу. Якщо ж кількість навчальних прикладів занадто велика для обраної структури мережі, то нейромережева модель може усереднювати вихідні значення для різних комбінацій вхідних векторів, що може призвести до втрати здатності до коректного відгуку на окремі випадки та збільшення максимальної вибіркової помилки. Таким чином, важливо правильно вибирати структуру нейронної мережі для кожної конкретної задачі, щоб досягти оптимального результату.

При автоматизації IT інфраструктури підприємства за допомогою багатошарових персептронів важливо правильно вибрати структуру мережі. Необхідно дотримуватися правила, згідно з яким кількість вагових коефіцієнтів повинна бути в 2-5 разів меншою за кількість прикладів навчальної вибірки [25]. Крім того, необхідно встановити кількість нейронів у прихованому шарі, що не менше кількості виходів. Завдання навчання багатошарових персептронів може бути сформульовано як оптимізаційна задача, де цільовою функцією є загальна помилка, що розраховується за навчальною вибіркою. Для налаштування вагових коефіцієнтів багатошарових персептронів не можна використовувати метод Уідроу-Хоффа, оскільки він дозволяє коригувати тільки нейрони вихідного шару [26]. Для розв’язання завдання навчання багатошарових персептронів можна використовувати методи детермінованого, градієнтного або стохастичного пошуку в рамках задачі багатовимірної оптимізації.

2.1.2 Узагальнено регресійна нейронна мережа (GRNN)

У шарі образів кожен вхідний образ розбивається на кілька кластерів, і кожен кластер асоціюється з ваговим коефіцієнтом, який є параметром гаусової функції ядра [27]. Підсумовувальний шар обчислює скалярний добуток між вектором ваг і вектором вхідних змінних, які знаходяться в кожному кластері. На вихідному шарі функцією активації є функція Гауссова,

яка обчислює відстань між вектором вхідних змінних та центром кластеру, після чого результати обчислення сумуються, щоб отримати вихідну змінну.

GRNN є досить простою в навчанні і використанні мережею, оскільки вона не вимагає задання функціональної залежності між вхідними та вихідними змінними. Навчання виконується з учителем, тобто з навчальною вибіркою, включаючи вхідні та вихідні змінні [28]. Задача навчання полягає у налаштуванні параметрів гаусових функцій ядра і вагових коефіцієнтів. Крім того, GRNN має властивість гладкості, що дозволяє їй працювати з деякими помилками вхідних даних. Це робить її корисною для моделювання завдань, які включають певний рівень шуму в даних.

GRNN може бути використана для розв'язання різноманітних завдань регресії, включаючи прогнозування часових рядів, моделювання фізичних процесів та вирішення задач в галузі фінансів [29].

Архітектура мережі представлена на рис. 2.2.

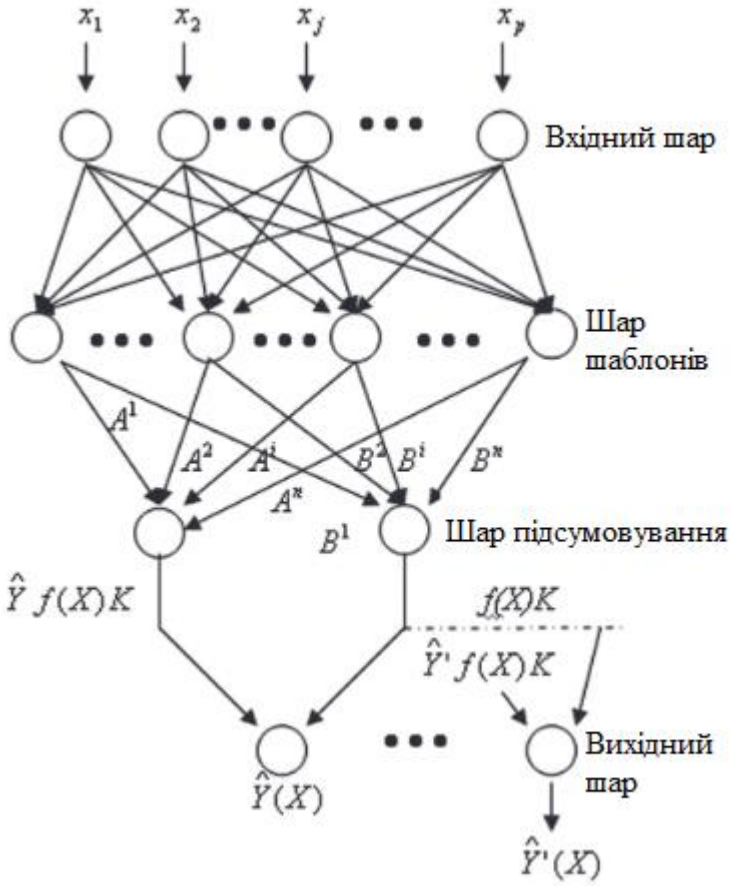


Рисунок 2.2 – Архітектура мережі GRNN

На вхідний шар подається навчальна вибірка. При цьому ваговим коефіцієнтам присвоюються значення вибірки та подаються на шар образів, у шарі образів вагові коефіцієнти проходять через функцію активації, яка має такий вигляд:

$$\sum_{i=1}^{n} \exp\left(-\frac{D_i^2}{2\sigma^2}\right) \qquad (2.2)$$

де D_i^2 – квадрат Евклідової відстані;

$$D_i^2 = (X - X^i)^T (X - X^i) \qquad (2.3)$$

σ — параметр згладжування, для найрезультативнішого значення мережі вибирається в проміжку від 2 до 6 [30]. Перетворені на попередньому кроці значення подаються на шар підсумовування, що складається з двох нейронів: S-нейрон накопичує суму зважених виходів шару образів.

$$\sum_{i=1}^{n} Y^i \exp\left(-\frac{D_i^2}{2\sigma^2}\right) \qquad (2.4)$$

D-нейрон обчислює суму незважених виходів шару образів

$$\sum_{i=1}^{n} \exp\left(-\frac{D_i^2}{2\sigma^2}\right) \qquad (2.5)$$

У вихідному шарі розраховується виважене середнє за формулою (2.6):

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} Y^i \exp\left(-\frac{D_i^2}{2\sigma^2}\right)}{\sum_{i=1}^{n} \exp\left(-\frac{D_i^2}{2\sigma^2}\right)} \qquad (2.6)$$

Узагальнено-регресійні нейронні мережі можуть бути застосовані для аналізу даних служби підтримки користувачів шляхом передбачення часу відповіді та рішення проблеми користувача. Наприклад, за допомогою GRNN можна створити модель, яка використовує дані з історії запитів користувачів до служби підтримки для передбачення часу відповіді та рішення проблеми [31].

Модель може бути навчена на історичних даних запитів користувачів, включаючи інформацію про час запиту, тип проблеми та час відповіді [32]. На основі цієї інформації модель може передбачити час відповіді на новий запит користувача та найбільш ефективний спосіб вирішення проблеми.

Результати передбачення можуть бути використані для автоматизації обробки запитів користувачів, наприклад, для автоматичної класифікації та маршрутизації запитів до відповідних служб підтримки. Це може знизити час очікування користувачів та підвищити ефективність роботи служби підтримки.

2.1.3 Радіально-базисна мережа (RBN)

Мережі з радіальною базисною функцією (РБФ-мережі) є особливим сімейством нейронних мереж, де приховані нейрони відображають радіальний простір навколо заданої точки або кластера [33]. У таких мережах сигнали

надходять від вхідного шару, оброблюються прихованим шаром з нейронами радіального типу, а потім суперпозуються вихідним нейроном для отримання відображення багатовимірного простору.

Мережі радіального типу є природним доповненням сигмоїдальних мереж [34]. Зазвичай структура РБФ-мережі складається з вхідного шару, прихованого шару з нейронами радіального типу та вихідного шару, що складається з лінійних нейронів. Функція вихідного нейрона полягає в зваженому підсумовуванні сигналів, що генеруються прихованими нейронами.

Використання в розкладанні p базисних функцій, де p – це кількість навчальних вибірок, неприпустимо з практичної точки зору, оскільки кількість цих вибірок може бути велика, і в результаті обчислювальна складність алгоритму, що навчає, може стати надмірною. Тому шукається субоптимальне рішення у просторі меншої розмірності, яке з достатньою точністю апроксимує точне рішення.

Якщо обмежиться K базисними функціями, то апроксимуюче рішення можна подати у вигляді:

$$F(x) = \sum_{i=1}^{K} w_i \cdot \varphi(\|x - c_i\|), \qquad (2.7)$$

де $K < p$, а c_i ($i = 1, 2, \ldots, K$) – множина центрів, які потрібно визначити. У разі, якщо прийняти $K = p$, можна отримати точне рішення $c_i = x_i$.

Завдання апроксимації радіальної базисної мережею полягає у підборі відповідної кількості радіальних функцій та їх параметрів, а також у такому підборі ваг ($i = 1, 2, \ldots, K$), щоб рішення рівняння (2.7) було найближчим до точного [35].

Проблему підбору параметрів радіальних функцій та значень ваги мережі можна звести до мінімізації цільової функції, яку можна записати в такій формі:

$$E = \sum_{i=1}^{p}\left[\sum_{j=1}^{K} w_j \cdot \varphi(\|x_i - c_i\|) - t_1\right]^2 \qquad (2.8)$$

У цьому рівнянні K представляє кількість радіальних нейронів, а p – кількість пар (x, t), де x – це вхідний вектор, а t – відповідна йому очікувана величина.

Найчастіше як радіальна функція застосовується функція Гауса. При розміщенні її центру в точці c_i вона може бути визначена як

$$\varphi(x) = \varphi(\|x - c_i\|) = exp\left[-\frac{\|x-c_i\|^2}{2\sigma_i^2}\right] \qquad (2.9)$$

У цьому виразі σ_i– параметр, від якого залежить ширина розмаху функції.

Архітектура радіальних мереж має структуру, аналогічну багатошаровій структурі сигмоїдальних мереж з одним прихованим шаром (рис. 2.3) [36].

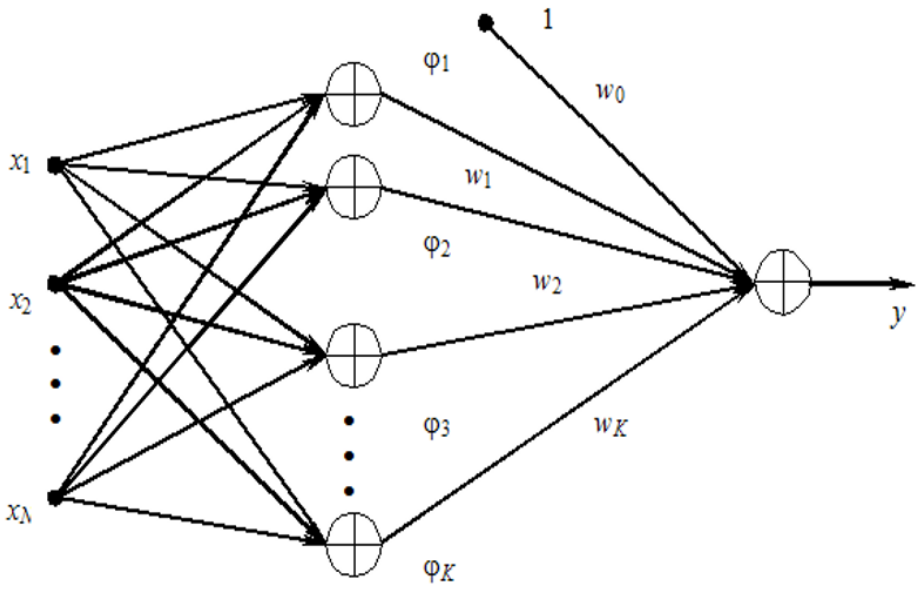


Рисунок 2.3 – Архітектура радіальних мереж

Радіальна мережа має фіксовану структуру з одним прихованим шаром та лінійними вихідними нейронами, тоді як сигмоїдальна мережа може містити різну кількість шарів, а вихідні нейрони бувають як лінійними, так і нелінійними.

Одним із найпростіших, хоч і не найефективнішим, способом визначення параметрів базисних функцій вважається випадковий вибір. У цьому випадку центри c_i базових функцій вибираються випадковим чином на основі рівномірного розподілу. Такий підхід припустимий стосовно класичних радіальних мереж за умови, що рівномірний розподіл навчальних даних добре відповідає специфіці завдання.

Мережі з радіальною базисною функцією можуть бути застосовані для аналізу даних служби підтримки користувачів, наприклад, для класифікації запитів користувачів та прогнозування часу розв'язання проблем.

Для цього можна використовувати дані, що надходять від користувачів, наприклад, текстові описи проблем, дати звернень та інші вхідні параметри. Ці дані можна підготувати та передати в мережу з радіальною базисною функцією для навчання моделі. Після навчання мережі можна використовувати для класифікації нових звернень та прогнозування часу їх розв'язання на основі аналізу вхідних даних.

2.2 Структура нейронної мережі

Нейронна мережа для аналізу даних служби підтримки користувачів буде побудована з використанням бібліотеки ML .NET. Така модель буде тренована на історичних даних служби підтримки користувачів з метою передбачення типу проблеми та способу її вирішення. Навчена модель буде повертати результат передбачення типу проблеми та способу її вирішення.

1. Вхідні та вихідні дані:

– вхідні дані представлені класом Input, який містить два поля: "Question" та "Answer". Ці поля відображають текстові питання та відповіді, які модель буде використовувати для навчання;

– вихідні дані представлені класом Output, який містить поля "Prediction", "Scores" та "PredictedAnswer". Після навчання моделі, вона буде передбачати відповіді на питання, що входять до тестового набору даних, а результати передбачень будуть зберігатись у відповідних полях.

2. Завантаження даних:

– дані зчитуються з текстового файлу та завантажуються у колекцію об'єктів класу Input.

3. Розділення даних на навчальний та тестовий набори:

– дані розбиваються на навчальний та тестовий набори в співвідношенні 80% / 20%.

4. Побудова моделі за допомогою створення конвеєру передобробки, який включає наступні кроки:

– FeaturizeText: перетворює текст питання в числові ознаки;

– ConvertType: конвертує відповіді в рядки;

– MapValueToKey: перетворює рядки в ключові значення;

– MapKeyToValue: перетворює ключові значення в рядки;

– SdcaNonCalibrated: застосовує алгоритм класифікації для навчання моделі.

5. Навчання моделі:

– виконується навчання моделі на навчальному наборі даних.

6. Оцінка моделі:

– модель оцінюється на тестовому наборі даних, вимірюючи метрики MicroAccuracy та MacroAccuracy.

Таким чином, структура нейронної мережі включає в себе побудову конвеєра передобробки, навчання моделі та її оцінку.

2.3 Метод навчання нейронної мережі за допомогою бібліотеки ML .NET

ML.NET - це відкрита платформа для розробки машинного навчання (Machine Learning), яка створена компанією Microsoft [37]. Це фреймворк, що дозволяє розробникам використовувати методи машинного навчання, щоб створювати програми, які можуть прогнозувати результати на основі вхідних даних.

Бібліотека ML.NET використовується для розв'язання різноманітних задач машинного навчання, таких як класифікація, регресія, кластеризація, обробка тексту, розпізнавання зображень та багато іншого. Фреймворк працює з багатьма мовами програмування, включаючи C#, F# та Python [38].

Для роботи з ML.NET не потрібно мати глибоких знань з машинного навчання або статистики, так як фреймворк містить вбудовані алгоритми та моделі, які можна використовувати безпосередньо. Однак, для розв'язання складніших задач можна створювати власні моделі, що базуються на алгоритмах машинного навчання, таких як SVM, Naive Bayes, K-Means та ін.

Основна перевага ML.NET полягає в тому, що він є частиною екосистеми Microsoft, що дає можливість інтегрувати фреймворк з іншими продуктами Microsoft, такими як Visual Studio, Azure та Excel [39]. Крім того, фреймворк є безкоштовним та має відкритий код, що дозволяє розробникам внести внесок у проект та створити свої власні моделі машинного навчання на основі фреймворка.

Перш ніж розпочати роботу з ML.NET, необхідно розібратись з основною концепцією ML.NET, яку необхідно використовувати для розробки додатків машинного навчання. Дану концепцію можна розділити на 4 етапи:

– завантаження даних. Для досягнення ідеального прогнозування результатів нам необхідно мати достатньо багато даних для навчання моделі. У ML.NET ми можемо надавати дані як для навчання, так і для перевірки в

різних форматах (текстовий файл CSV/TSV, реляційна база даних (тепер підтримуються SQL Server, Oracle, MySQL та ін.), бінарні файли, IEnumerable та інші) [40];

– навчання. Потрібно вибрати правильний алгоритм для навчання моделі в залежності від потреб, для навчання та прогнозування результатів;

– оцінка. Вибрати тип машинного навчання для моделі навчання та прогнозування. Якщо потрібно працювати з сегментом, можна вибрати модель кластеризації, якщо потрібно знайти ціну прогнозування акцій, можна вибрати регресію і, якщо потрібно знайти аналіз настроїв, можна вибрати модель класифікації;

– прогнозовані результати. На основі даних про навчання та тестування з навченою моделлю остаточний прогноз відображатиметься за допомогою програми ML.NET. Навчена модель буде збережена у двійковому форматі, який також може бути інтегрований з іншими програмами .NET.

На рис. 2.4 пояснено послідовність процесів, які будуть використовуватися для розробки програм машинного навчання з використанням ML.NET.

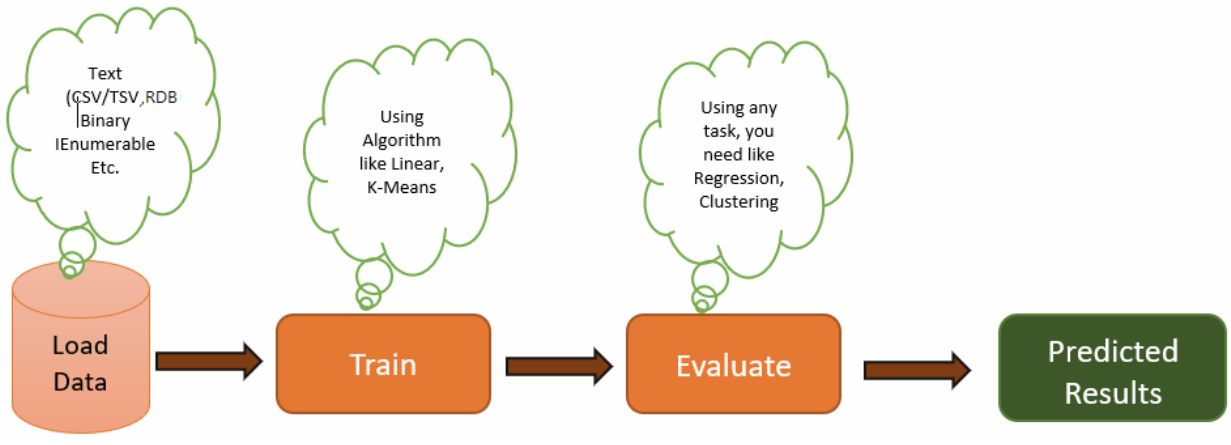


Рисунок 2.4 – Послідовність процесів машинного навчання з використанням ML.NET

Структура підходу для використання ML.NET повинна включати наступні ключові компоненти, що відображають чотири етапи, зазначені у наведеній інформації:

1. Завантаження даних:

– визначення схеми даних. Створення класів вхідних та вихідних даних, які представляють структуру даних;

– зчитування даних. Використання ML.NET для завантаження даних з різних джерел, таких як текстові файли (CSV/TSV), реляційні бази даних (SQL Server, Oracle, MySQL тощо), бінарні файли або IEnumerable [41];

– передопрацювання даних. Застосування операцій передопрацювання, таких як очищення, трансформація або видалення непотрібних стовпців.

2. Навчання моделі:

– вибір алгоритму. Обрання відповідного алгоритму машинного навчання в залежності від потреб, таких як класифікація, регресія або кластеризація;

– конфігурація конвеєру. Створення конвеєру обробки даних та налаштування властивостей алгоритму;

– навчання моделі: використання обраних алгоритмів на навчальному наборі даних для створення моделі.

3. Оцінка моделі:

– розділення даних. Розділення зчитаних даних на навчальний та тестовий набори;

– оцінка якості моделі. Застосування моделі до тестового набору даних та вимірювання її точності або інших метрик якості.

4. Прогнозування результатів:

– застосування моделі. Використання навченої моделі для роботи з новими даними та отримання прогнозів;

– збереження та використання моделі. Збереження моделі у двійковому форматі для подальшого використання в інших .NET-програмах або середовищах.

Використовуючи бібліотеку ML.NET буде створено нейронну мережу для аналізу даних служби підтримки користувачів, яка буде тренуватися на історичних даних для передбачення типу проблеми та способів її вирішення.

Використання навченої моделі для передбачення типу проблеми та способів її вирішення дозволяє автоматизувати обробку запитів користувачів, класифікацію та маршрутизацію запитів до відповідних служб підтримки. Цей підхід забезпечує зменшення часу очікування для користувачів та покращення ефективності роботи служби підтримки.

ML.NET у комбінації з машинним навчанням та нейронними мережами дозволяє розробляти автоматизовані рішення для підвищення продуктивності IT-інфраструктури підприємства та оптимізації роботи служб підтримки користувачів.

2.4 Алгоритм процесу аналізу та прогнозування на основі бібліотеки ML .NET

Застосовуючи нейронні мережі для розв'язання задачі автоматизації IT інфраструктури підприємства за допомогою машинного навчання, необхідно здійснити навчання мережі таким чином, щоб точність прогнозування була якомога вищою.

На рис. 2.5 зображено блок схему алгоритму реалізації методу навчання.

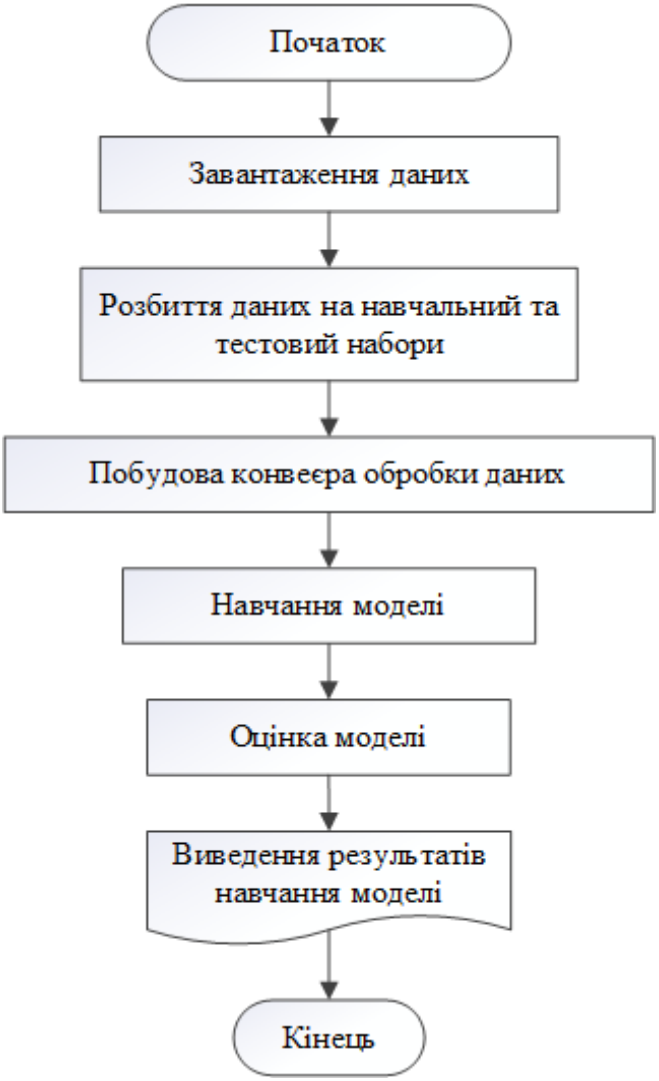


Рисунок 2.5 – Алгоритм процесу навчання нейронної мережі

Блок-схема ілюструє основні етапи створення, навчання та використання моделі машинного навчання за допомогою бібліотеки ML.NET.

Алгоритм навчання нейронної мережі за допомогою бібліотеки ML.NET включає наступні кроки:

1. Завантаження даних. Для навчання та перевірки моделі забезпечуються дані у вигляді текстового файлу у форматі CSV. За допомогою OpenFileDialog відкривається діалогове вікно для вибору файлу з даними. Після вибору файлу, його вміст зчитується та конвертується у список об'єктів Input.

2. Підготовка даних. Завантажені дані розбиваються на дві частини - навчальну та тестову, використовуючи метод TrainTestSplit. Навчальні дані використовуються для побудови моделі, а тестові дані - для її оцінки.

3. Побудова та навчання моделі. Створюється конвеєр обробки даних (pipeline), що складається з наступних етапів:

– FeaturizeText: перетворює текстові дані з питань на числові ознаки, що можуть бути використані алгоритмами машинного навчання;

– ConvertType: перетворює типи даних, які будуть використані у виході (мітках) моделі;

– MapValueToKey: відображає значення мітки на ключ;

– MapKeyToValue: відображає ключ мітки на значення;

– SdcaNonCalibrated: застосовує алгоритм багатокласової класифікації для навчання моделі на основі навчальних даних;

– MapKeyToValue: відображає передбачені ключі на значення.

4. Навчання моделі. Використовуючи метод Fit, модель навчається на навчальних даних, що проходять через конвеєр обробки даних (pipeline).

5. Оцінка моделі. Навчена модель перевіряється на тестових даних за допомогою методу Transform. Отримані передбачення оцінюються за допомогою методу Evaluate, який порівнює передбачені значення з реальними мітками. Розраховуються метрики, такі як мікросередня та макросередня точність, які дають змогу оцінити якість моделі та визначити, наскільки вона може передбачати правильний тип проблеми та спосіб її вирішення.

6. Використання моделі для прогнозування. Після навчання та оцінки моделі, її можна зберегти та використовувати для передбачення типу проблеми та способу її вирішення на основі нових даних, які надходять у вигляді текстових питань. Модель здатна здійснювати передбачення на основі навчальних даних та видає результат у вигляді передбаченого типу проблеми та способу її вирішення.

У результаті, алгоритм навчання нейронної мережі для бібліотеки ML.NET має такі характеристики:

– використовує ML.NET для побудови, навчання та оцінки моделі машинного навчання;

– передбачає тип проблеми та спосіб її вирішення на основі навчальних даних, що забезпечуються у форматі CSV;

– використовує конвеєр обробки даних для підготовки даних та застосування алгоритмів машинного навчання;

– розраховує метрики точності для оцінки якості моделі.

Завдяки детальному опису алгоритму, можна легко адаптувати та використовувати навчену модель у різних сценаріях автоматизації обробки запитів користувачів та підвищення ефективності роботи служби підтримки.

2.5 Висновок

У даному розділі було досліджено та розглянуто різні моделі та методи прогнозування даних для служби підтримки користувачів з метою автоматизації IT інфраструктури підприємства за допомогою машинного навчання та нейронних мереж. Спочатку було проведено аналітичний огляд існуючих методів розв'язання задачі, зокрема багатошарового персептрону (MLP), узагальненої регресійної нейронної мережі (GRNN) та радіально-базисної мережі (RBN). Це допомогло зрозуміти переваги та недоліки кожної моделі та вибрати найбільш підходящий метод для виконання задачі.

Далі було досліджено структуру нейронної мережі, що допомогло визначити оптимальну архітектуру мережі для розв'язання поставленої задачі.

Також було розроблено алгоритм процесу аналізу та прогнозування на основі бібліотеки ML .NET, який може бути застосований для аналізу даних служби підтримки користувачів та прогнозування потенційних проблем. Завдяки цьому, можливо покращити ефективність служби підтримки,

скоротити час очікування користувачів на вирішення проблем та оптимізувати розподіл ресурсів.

3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО РІШЕННЯ

3.1 Засоби розробки

3.1.1 Вибір архітектурного шаблону

Задача розробки програмного забезпечення полягає у створенні ефективної, надійної та масштабованої системи, яка здатна задовольнити потреби користувачів. Вибір правильної архітектури є ключовим етапом в розробці будь-якої програмної системи, оскільки він може впливати на практично всі аспекти програми, від швидкодії та масштабованості до простоти розширення та підтримки. У цьому підрозділі буде розглянуто основні архітектурні шаблони та визначено їх переваги та недоліки.

Трирівнева архітектура - це архітектурний шаблон для розробки програмного забезпечення, що складається з трьох рівнів: інтерфейсу користувача, бізнес-логіки та рівня доступу до даних [42]. Кожен рівень виконує свої функції та взаємодіє з іншими рівнями для забезпечення більшої ефективності.

Мікросервісна архітектура - це підхід до розробки програмного забезпечення, який полягає в створенні невеликих, автономних сервісів, що працюють разом для забезпечення більш великої функціональності [43]. Кожен мікросервіс може бути розроблений та випущений незалежно від інших сервісів та взаємодіє з ними за допомогою API.

MVC - це абревіатура, що означає Model-View-Controller (Модель-Вид-Контролер) і використовується як архітектурний шаблон для розробки веб-додатків [44]. Він розділяє логіку програми на три компоненти: Модель (дані та логіка), Вид (представлення даних) та Контролер (управління даними та взаємодія між Видом та Моделлю). Це дозволяє забезпечити простоту розробки, підтримку тестування та модифікацію окремих компонентів без впливу на решту системи.

У табл. 3.1 проведено порівняльний аналіз архітектурних шаблонів.

Таблиця 3.1 – Порівняльний аналіз архітектурних шаблонів

Характеристика	*Трирівнева архітектура*	*Мікросервісна архітектура*	*MVC*
Рівні	3 (інтерфейс, бізнес-логіка, доступ до даних)	Залежить від компонентів системи	3 (Модель, Вид, Контролер)
Залежність між компонентами	Сильна	Слабка	Середня
Масштабованість	Обмежена	Висока	Середня
Керованість	Легко керується	Складно керується	Легко керується
Загальна складність	Середня	Висока	Середня
Швидкість розробки	Швидка	Повільна	Швидка
Сумісність	Добра	Середня	Добра
Тестування	Легко тестується	Складно тестується	Легко тестується
Виконання	Швидке виконання	Залежить від сервісів	Швидке виконання

Трирівнева архітектура є одним з найпоширеніших архітектурних шаблонів, який зазвичай використовується для розробки додатків. Цей шаблон дозволяє розділити компоненти системи на три рівні, що спрощує розробку, підтримку та модифікацію додатку.

Для розробки додатку було обрано трирівневу архітектуру з наступних причин:

– простота розробки: дозволяє розробити додаток швидко та ефективно, оскільки кожен рівень відповідає за свої функції та має чітку взаємодію з іншими рівнями;

– легка підтримка: розділення компонентів на три рівні дозволяє здійснювати зміни та підтримку окремих компонентів без впливу на інші частини системи;

– гнучкість: дозволяє замінювати та модифікувати окремі компоненти системи, що дозволяє адаптувати додаток до змінних потреб бізнесу;

– підтримка тестування: розділення на три рівні дозволяє проводити тестування окремих компонентів системи та виконувати їхню інтеграцію для забезпечення якості додатку.

Отже, трирівнева архітектура дозволяє розробити додаток, який буде простим у підтримці, гнучким та легким у тестуванні.

3.1.2 Вибір мови програмування

Вибір мови програмування є ключовим етапом у процесі розробки програмного забезпечення.

C#, Java та C++ - це мови програмування, які використовуються для розробки програмного забезпечення.

C# (C Sharp) - це мова програмування, розроблена компанією Microsoft [45]. Вона використовується для розробки додатків для платформи NET Framework. C# є об'єктно-орієнтованою мовою програмування, яка має синтаксис, подібний до мови Java.

Java - це мова програмування, розроблена компанією Sun Microsystems [46]. Вона використовується для розробки програмного забезпечення для різних платформ, включаючи сервери, комп'ютери та мобільні пристрої. Java є об'єктно-орієнтованою мовою програмування та забезпечує підтримку багатьох програмних бібліотек.

C++ - це мова програмування, яка використовується для розробки програмного забезпечення, що працює на різних платформах, включаючи комп'ютери та мобільні пристрої [47]. C++ є мовою програмування загального призначення та надає можливість контролювати рівень доступу до пам'яті та ресурсів системи.

У табл. 3.2 проведено порівняльний аналіз можливостей кожної мови програмування.

Таблиця 3.2 – Порівняльний аналіз мов програмування

Характеристика	*C#*	*Java*	*C++*
Парадигма програмування	Об'єктно-орієнтована	Об'єктно-орієнтована	Об'єктно-орієнтована, процедурна
Виконання коду	Компілюється до проміжного байт-коду та виконується на платформі .NET	Компілюється до байт-коду та виконується на віртуальній машині Java	Компілюється в машинний код та виконується безпосередньо на процесорі
Використання пам'яті	Керується автоматично збирачем сміття	Керується автоматично збирачем сміття	Ручне керування пам'яттю
Швидкість виконання	Середня швидкість виконання	Середня швидкість виконання	Швидке виконання
Масштабованість	Добра масштабованість	Добра масштабованість	Добра масштабованість
Наявність бібліотек та фреймворків	Є багато бібліотек та фреймворків для	Є багато бібліотек та фреймворків для	Є багато бібліотек та фреймворків для

	різних типів проектів	різних типів проектів	різних типів проектів

Для розробки додатку обрано мову програмування C#. При цьому було враховано наступні фактори: підтримка платформи .NET, об'єктно-орієнтована парадигма програмування, наявність багатьох фреймворків та бібліотек, зручний інтерфейс Visual Studio та підтримка популярних платформ та сервісів. Ці фактори дозволяють зручно та ефективно розробляти та підтримувати додаток. Таким чином, обрання мови програмування C# було обґрунтовано інструментарієм, який надається для розробки на цій мові та її можливостями у роботі з платформою .NET.

3.1.3 Вибір СУБД

В сучасному світі, коли об'єми даних зростають з кожним днем, важливість систем управління базами даних (СУБД) стає все більш вагомою. Вони є ключовим елементом у збереженні, організації та доступі до даних для різноманітних додатків та систем. Вибір правильної СУБД може впливати на швидкість роботи додатків, безпеку даних та їх доступність для аналізу та використання.

У даному підрозділі буде розглянуто декілька популярних СУБД визначено та їх переваги та недоліки. Також, необхідно проаналізувати, які фактори впливають на вибір СУБД, такі як розмір даних, складність запитів, доступність та ціна.

MS SQL Server - це реляційна база даних, яка розроблена компанією Microsoft [48]. Вона використовується для зберігання даних на різних платформах, включаючи Windows та Linux. MS SQL Server має широкі можливості управління даними, дозволяє працювати з різними форматами даних та має вбудовану підтримку засобів бізнес-аналітики.

Oracle - це одна з найпоширеніших реляційних баз даних у світі, розроблена компанією Oracle Corporation [49]. Вона використовується для

зберігання та управління даними у різних галузях, включаючи бізнес, організації та установи. Oracle має велику кількість вбудованих функцій та засобів, які дозволяють підвищити ефективність роботи з даними та оптимізувати роботу з базою даних.

MS Access - це реляційна база даних, яка розроблена компанією Microsoft та призначена для роботи з невеликими базами даних [50]. MS Access має простий інтерфейс та можливості для швидкого створення та редагування баз даних. Вона підтримує стандартні функції реляційних баз даних та має вбудований засіб створення звітів та форм.

Порівняння основних характеристик MS SQL Server, Oracle та MS Access наведено в табл. 3.3.

Таблиця 3.3 – Порівняльний аналіз сховищ БД

Характеристика	*MS SQL Server*	*Oracle*	*MS Access*
Розробник	Microsoft	Oracle Corporation	Microsoft
Тип бази даних	Реляційна	Реляційна	Реляційна
Версія	2019	19c	2019
Масштабованість	Велика	Велика	Мала
Підтримка мов програмування	T-SQL, C#, Java та ін.	PL/SQL, Java та ін.	VBA, SQL
Інтерфейси	ODBC, OLE DB, JDBC, ADO.NET	ODBC, JDBC, ADO.NET	ODBC, OLE DB
Підтримка операційних систем	Windows, Linux	Windows, Linux, UNIX	Windows

Для зберігання даних додатку було обрано СУБД MS Access. Однією з переваг MS Access є її простий інтерфейс та зручність для створення

невеликих баз даних, що відповідають потребам користувача. MS Access також ходить до складу пакету Microsoft Office, що робить її доступною для користувачів, які вже користуються іншими програмами Office.

3.2 Проєктування бази даних

Розробка проекту розпочинається із проектування бази даних. У даному випадку було використано метод "сутність-зв'язок" (ER метод) - це підхід до проектування баз даних, що передбачає визначення сутностей, їх атрибутів та зв'язків між ними. Застосування цього методу дозволяє створити логічну модель бази даних, що відображає предметну область та відносини між її елементами.

У процесі проектування бази даних за методом "сутність-зв'язок" для предметної області було виділено наступні сутності та атрибути:

– історія відповідей: ідентифікатор, запитання, відповідь, рейтинг, ідентифікатор користувача, ідентифікатор теми, зауваження користувача,

– події: ідентифікатор події, дата події, ідентифікатор користувача та опис події;

– користувачі: ідентифікатор користувача, прізвище, ім’я, назва облікового запису, пароль, ідентифікатор ролі та додатковий опис;

– нейронна мережа: ідентифікатор, назва нейронної мережі, шлях до файлу збереженої моделі, ідентифікатор теми;

– тема: ідентифікатор, назва теми та її опис.

Під час проектування бази даних було визначено зв'язки між сутностями та встановлені зовнішні та внутрішні ключі для їхнього взаємозв'язку. Застосування зовнішнього та внутрішнього ключа дозволило встановити зв'язки між таблицями та визначити типи зв'язків, такі як "один до одного", "один до багатьох" та "багато до багатьох". Визначення типів зв'язків між сутностями допомогло правильно побудувати логічну модель бази даних, що забезпечить її коректну роботу та ефективне використання.

Організовано такі зв'язки між таблицями:

"Themes" (поле "ThemesId") – "Neural" (поле "ThemesId"), вид зв’язку 1:N;

"Themes" (поле "ThemesId") – "AnsverHistory" (поле "ThemesId"), вид зв’язку 1:N;

"Users" (поле "UsersId") – "Logs" (поле "UsersId"), вид зв’язку 1:N;

"Users" (поле "UsersId") – "AnsverHistory" (поле "UsersId"), вид зв’язку 1:N.

ER-діаграма є важливим етапом проектування бази даних, оскільки дозволяє відображати зв'язки між сутностями та їх атрибутами. В результаті було створено ER-діаграму для бази даних, яка наглядно відображає взаємозв'язки між таблицями та їх полями (рис. 2.3). Ця модель є важливою при подальшому проектуванні та розробці функціоналу додатку.

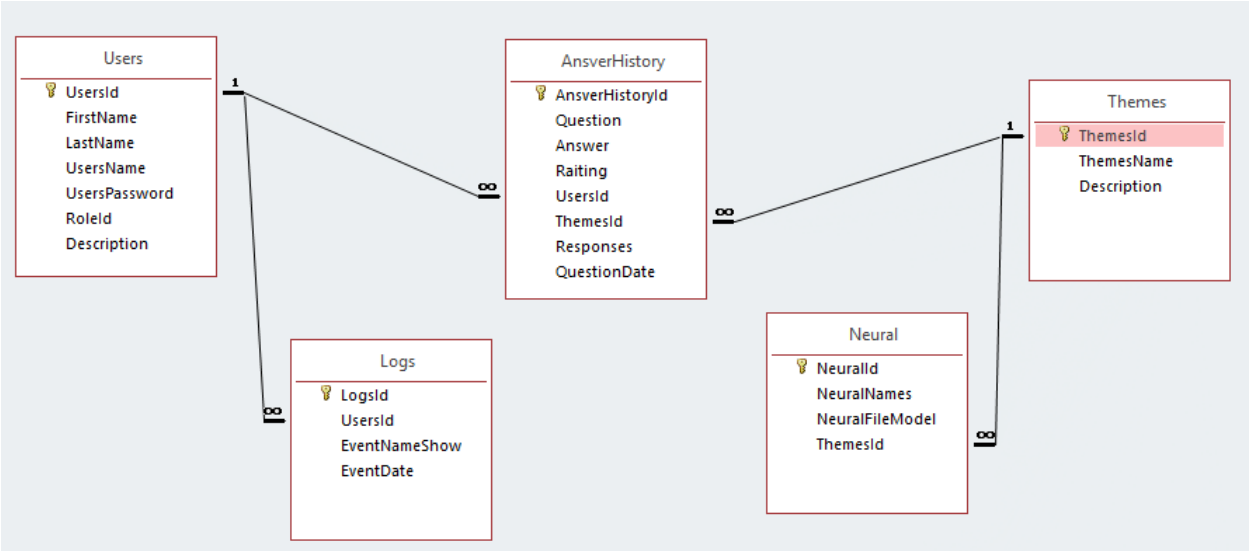


Рисунок 3.1 – ER-діаграма бази даних

3.3 Архітектура проекту

Архітектура проекту ~~буде~~ базується на трирівневій архітектурі, яка складається з наступних рівнів:

– рівень презентації - це верхній рівень, який відповідає за взаємодію користувача з системою. В даному проекті рівень презентації буде забезпечувати інтерфейс користувача, який буде розроблено з використанням середовища розробки Visual Studio 2019 та мови програмування C#. Інтерфейс користувача буде містити форми, на яких будуть відображені дані та інші елементи, які забезпечують взаємодію з користувачем;

– рівень бізнес-логіки - це середній рівень, який відповідає за обробку даних та реалізацію бізнес-логіки проекту. На цьому рівні будуть знаходитися класи, які забезпечують роботу з даними та виконання бізнес-логіки, що потрібна для розробки інформаційної системи підтримки прийняття рішень для інтернет продаж;

– рівень доступу до даних - це нижній рівень, який відповідає за зберігання та доступ до даних. На цьому рівні будуть знаходитися класи, які забезпечують роботу з базою даних та доступ до неї.

Трирівнева архітектура дозволить розділити функціональні можливості проекту на окремі рівні, що дозволить підтримувати легкість модифікації, тестування та розширення проекту.

На початку було реалізовано шар доступу до даних Для роботи з базою даних реалізовано 5-ть класів. Назва кожного класу починається із відповідній їй назві таблиці в базі даних та закінчується приставкою «Provider». Рис. 3.2 відображає діаграму класів з методами для роботи з базою даних.

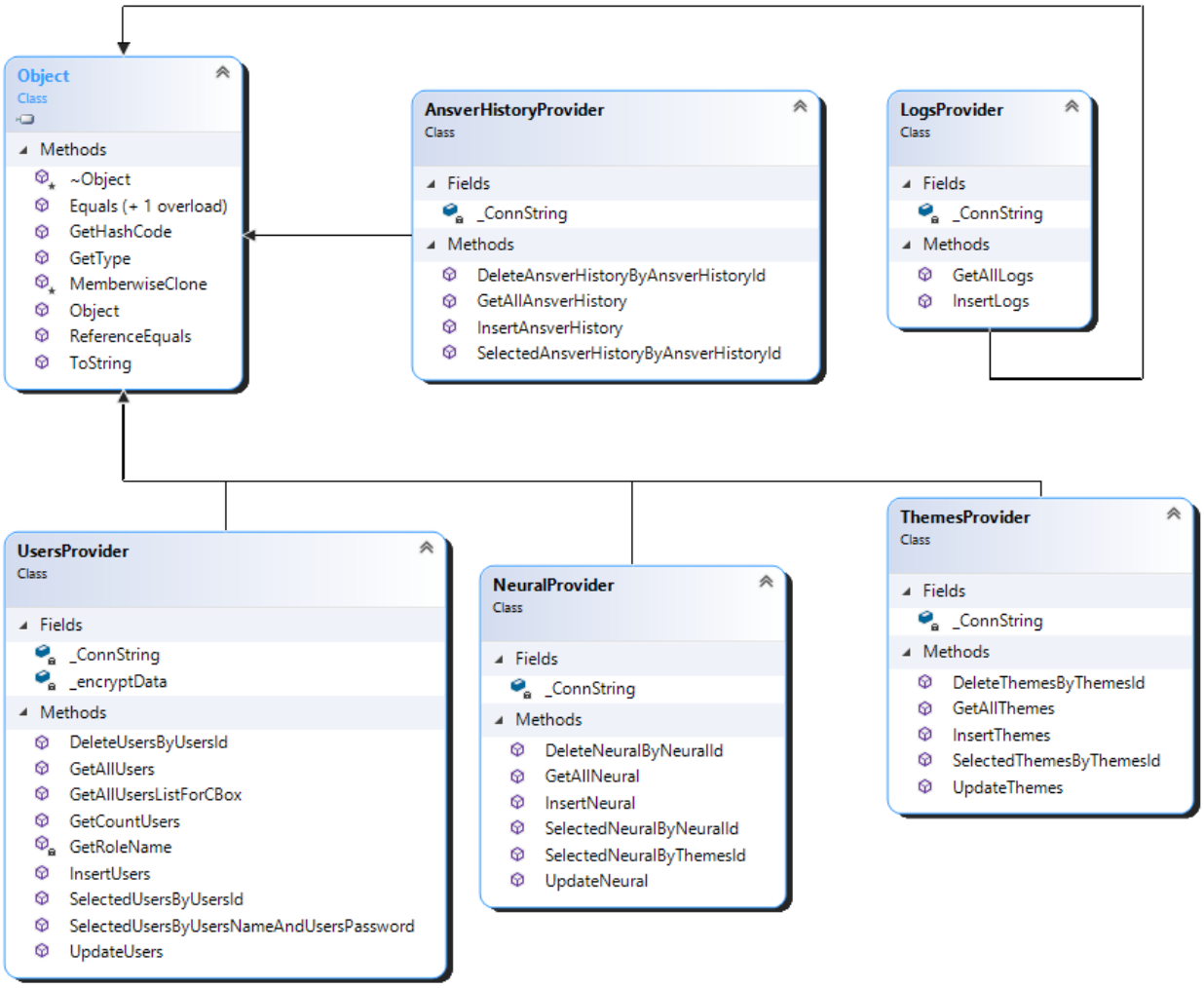


Рисунок 3.2 – Діаграма класів рівня даних

Як видно із рис. 3.2, діаграма класів рівня даних складається із 5-ти основних класів:

– клас AnswerHistoryProvider забезпечує зберігання історії відповідей на запити користувачів. Він містить методи для додавання нових записів до історії відповідей та видалення існуючих записів. Клас також містить методи для отримання історії відповідей на запити користувачів;

– клас LogsProvider забезпечує зберігання логів дій користувачів у системі. Він містить методи для додавання нових записів до логів, видалення та оновлення існуючих записів. Клас також може містити методи для отримання логів дій користувачів у системі, наприклад, для відображення їх на сторінці адміністратора;

– клас NeuralProvider забезпечує роботу нейронної мережі;

– клас ThemesProvider забезпечує роботу з темами служби підтримки. Він містить методи для додавання нових тем, видалення та оновлення існуючих тем, а також для отримання інформації про теми;

– клас UsersProvider забезпечує роботу з користувачами системи. Він містить методи для додавання нових користувачів, видалення та оновлення інформації про існуючих користувачів. Клас також містить методи для автентифікації користувачів та для контролю доступу до ресурсів системи.

На рівні інтерфейсу користувача було розроблено форми, що складаються з набору елементів керування (кнопки, текстові поля, таблиці), які взаємодіють з іншими рівнями системи через відповідні обробники подій. Користувач може взаємодіяти з системою через цей рівень, вводити необхідну інформацію, здійснювати пошук і отримувати результати у зручному для нього форматі. Рис. 3.3 відображає діаграму системи рівня інтерфейсу користувача.

Рисунок 3.3 – Діаграма інтерфейсу системи

Діаграма даного рівня складається з дванадцяти класів рівня UI та є похідними від класу Form, тобто мають графічний інтерфейс.

Клас «InfrAutoMDI» є центральним вікном програми. Він містить головне меню, що дозволяє користувачеві відкривати інші форми та виконувати потрібні дії. Для реалізації функціоналу головного меню використовується клас «MenuStrip», який дозволяє створювати меню з підменю та елементами меню. Він додається до головного вікна за допомогою методу «Controls.Add()».

Клас AutoAnswerForm забезпечує автоматичну відповідь на запит користувача. Він містить методи для обробки вхідних даних користувача та генерації відповіді на основі використання нейронної мережі.

Клас TeachingNeuralForm забезпечує процес навчання нейронної мережі. Він містить методи для створення та налаштування нейронної мережі, для вибору набору даних для навчання та для запуску процесу навчання.

ThemesForm - це клас, який забезпечує роботу з темами для служби підтримки IT відділів.

UpdateThemesForm - це клас, який забезпечує оновлення даних про теми для служби підтримки IT відділів. Він містить методи для видалення та оновлення існуючих тем.

ThemesReportForm - це клас, який забезпечує формування звіту про дослідження за вибраною із списку темою.

ReportForPeriodDateForm - це клас, який забезпечує формування звіту за період часу. Він містить методи для вибору періоду часу та для аналізу даних, які були зібрані за цей період.

LoginForm - це клас, який забезпечує автентифікацію користувачів. Він містить методи для перевірки введених даних користувача, для визначення рівня доступу та для запуску процесу авторизації.

PersonalizationForm - це клас, який забезпечує персоналізацію додатку для користувача.

Створення об'єктів інших класів та виклик їх методів відбувається в результаті взаємодії користувача з елементами графічного інтерфейсу програми.

Останнім етапом реалізації архітектури було створення бізнес-логіки додатку. Створення бізнес-логіки додатку є дуже важливим етапом в розробці

програмного забезпечення. Бізнес-логіка дозволяє обробляти різні бізнес-операції. Вона зазвичай не пов'язана з базою даних або інтерфейсом користувача, а забезпечує правильне функціонування додатку в цілому.

Створення бізнес-логіки дозволяє розділити функціонал додатку на логічні блоки, що сприяє більшій модульності та гнучкості в майбутньому. Це дозволяє змінювати окремі частини функціоналу без впливу на інші частини додатку, що полегшує підтримку та розвиток проекту.

Крім того, розділення бізнес-логіки від бази даних та інтерфейсу користувача дозволяє використовувати різні бази даних та різні інтерфейси без впливу на логіку додатку в цілому. Це забезпечує більшу гнучкість та адаптивність додатку до змін у вимогах бізнесу та технологіях.

Діаграму даного шару показано на рис. 3.4.

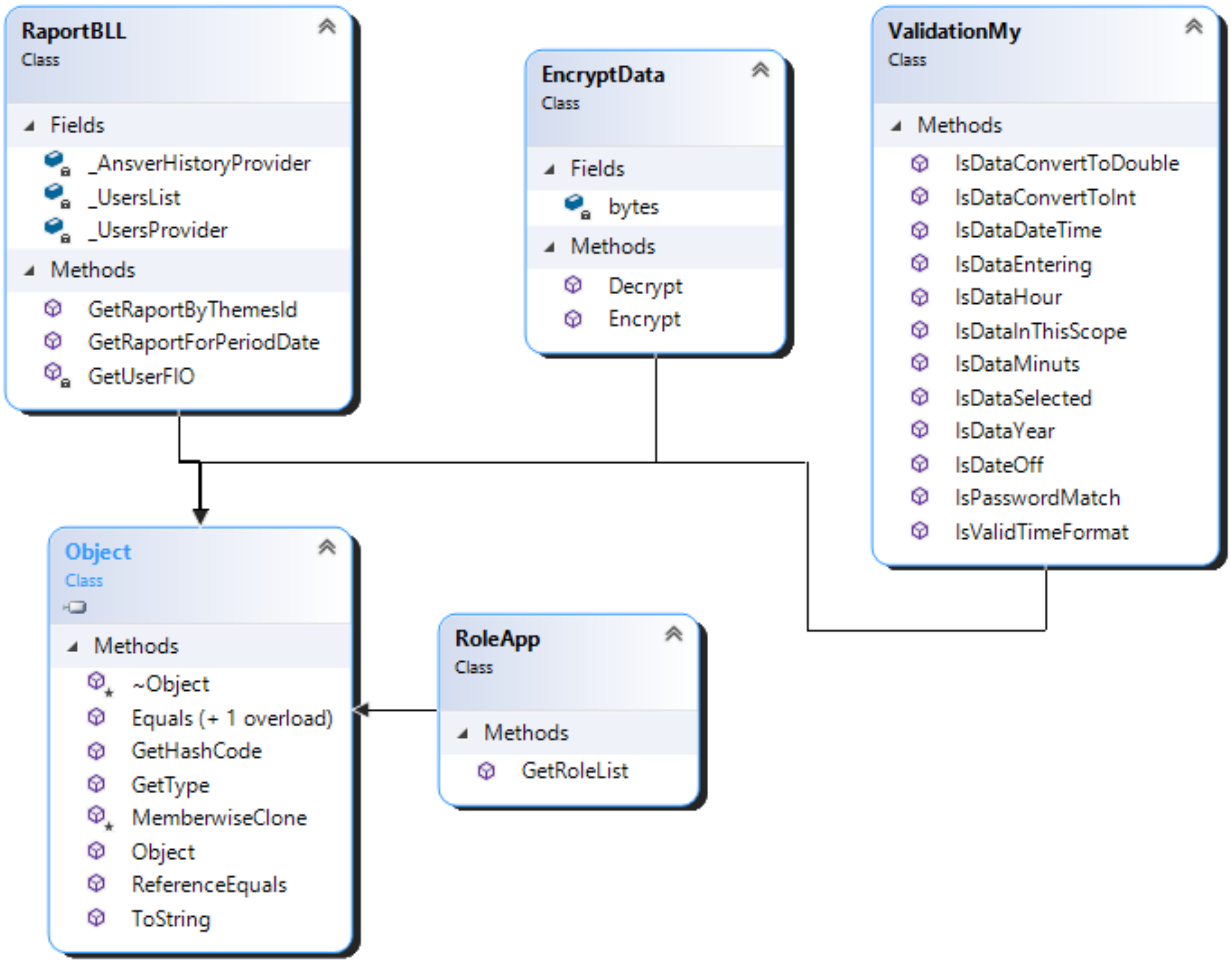


Рисунок 3.4 – Діаграма класів бізнес-логіки

Діаграма даного рівня складається з чотирьох основних класів:

– клас ValidationMy відповідає за перевірку даних на коректність та валідність. Він містить методи для перевірки різних типів даних, таких як рядки, числа, дати та інші. Цей клас дозволяє забезпечити коректну роботу додатку та запобігти помилкам при обробці даних;

– клас EncryptData відповідає за шифрування та дешифрування даних. Він може містить методи для захисту конфіденційної інформації в додатку, такої як паролі, дані користувачів. Цей клас дозволяє забезпечити безпеку даних в додатку та запобігти несанкціонованому доступу до конфіденційної інформації;

– клас RoleApp відповідає за керування ролями користувачів в додатку. Він містить методи для призначення ролей користувачам, перевірки доступу до різних частин функціоналу додатку в залежності від ролі користувача. Цей клас дозволяє забезпечити безпеку та контроль доступу до функціоналу додатку, а також дозволяє забезпечити персоналізований доступ користувачів до різних частин додатку;

– клас RaportBLL відповідає за бізнес-логіку звітів в додатку. Він містить методи для формування звітів на основі даних з бази даних, а також для обробки даних перед відображенням їх на екрані. Цей клас дозволяє забезпечити коректне та ефективне формування звітів в додатку, що є важливою функціональною частиною бізнес-систем.

3.4 Розробка модулів проекту

Після створення бази даних наступним кроком потрібно підключити її до системи за допомогою Visual Studio 2019. Це дасть можливість підключати всі таблиці до форм, на яких буде можливість додавати, редагувати та видаляти дані. Щоб здійснити підключення, необхідно створити змінну

"CONNECT" у файлі проекту "App.config" і задати значення параметрів, які можна переглянути на рис. 3.5.

```
<appSettings>
  <!-- Підключення до бази даних -->
  <add key="CONNECT" value="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=|DataDirectory|\DataBase.mdb;" />
  <add key="ClientSettingsProvider.ServiceUri" value="" />
</appSettings>
```

Рисунок 3.5 – Змінна із параметрами налаштування бази даних

Для ефективної роботи з базою даних буде використано простір імен System.Data.OleDb. Цей простір містить класи для зв'язку з базою даних OleDbConnection, що дозволяє взаємодіяти з даними у базі даних, виконувати запити та зберігати зміни. Щоб створити меню в проекті додано елемент menuStrip до головного вікна. Це дає можливість користувачам вибирати опції та викликати функції програми за допомогою простого інтерфейсу. Результат можна побачити на рис. 3.6.

Рисунок 3.6 – Додавання головного меню

Для кожного пункту меню було додано відповідний код, який створює екземпляр форми. При виборі певного пункту меню, відповідна форма відкривається, і її вікно стає активним. Крім того, було включено код, який забезпечує закриття попередньо відкритого вікна перед відкриттям нового. Це дозволяє коректно відображати вікна та уникнути можливих конфліктів. Аналогічний код застосовується для всіх пунктів меню, забезпечуючи їх правильну роботу та взаємозв'язок з формами. На рисунку 3.7 наведено приклад такого коду для одного з пунктів меню.

```
1 reference
private void навчанняНейронноїМережіToolStripMenuItem_Click(object sender, EventArgs e) {
    CloseAllWindows();
    TeachingNeuralForm teachingNeuralForm = new TeachingNeuralForm();
    teachingNeuralForm.MdiParent = this;
    teachingNeuralForm.WindowState = FormWindowState.Maximized;
    teachingNeuralForm.Show();
}
```

Рисунок 3.7 – Код пунктів меню

Після цього було розроблено класи для роботи із базою даних. Нижче приведено часткові реалізації методів різних класів із детальним описом. Наприклад, для додавання інформації про нейронну мережу у базу даних був створений метод із назвою «InsertNeural», код даного методу показано на рис. 3.8.

```
public void InsertNeural(string NeuralNames, int ThemesId, string NeuralFileModel) {
    string SqlString = "INSERT INTO Neural (NeuralNames, ThemesId, NeuralFileModel) Values(?, ?, ?)";

    using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
            cmd.CommandType = CommandType.Text;
            cmd.Parameters.AddWithValue("NeuralNames", NeuralNames);
            cmd.Parameters.AddWithValue("ThemesId", T
            cmd.Parameters.AddWithValue("NeuralFileMo
            conn.Open();
            cmd.ExecuteNonQuery();
            conn.Close();
        }
    }
}
```

class System.String
Represents text as a sequence of UTF-16 code units.

Рисунок 3.8– Код методу «InsertNeural»

Даний метод відповідає за вставку нових даних в таблицю "Neural" бази даних. Він містить параметри, що передаються як аргументи методу: назву нейромережі, ідентифікатор теми та шлях до файлу тренованої моделі.

Код методу формує SQL-запит на вставку даних в таблицю "Neural". Потім він створює новий екземпляр OleDbConnection та OleDbCommand, в якому передається підготовлений SQL-запит та параметри, які будуть додані до запиту. Параметри передаються за допомогою методу AddWithValue.

Далі викликається метод ExecuteNonQuery(), який виконує SQL-запит на вставку даних у таблицю. Після цього з'єднання з базою даних закривається.

Цей метод може використовуватися для додавання нових записів в таблицю "Neural", що є важливим для функціоналу навчання та використання нейронних мереж в додатку.

Для видалення даних про нейронну мережу, також був розроблений метод, код якого представлено на рис. 3.9.

```
1 reference
public void UpdatePets(string PetsName, string Breed, string Sex, int Age, DateTime ArrivalDate,
  bool IsSterilization, string Description, byte[] PetsImage, int KindId, int PetsId) {
  MySqlConnection connection = new MySqlConnection(_ConnString);
  string query = "UPDATE Pets SET PetsName = @PetsName, Breed = @Breed, Sex = @Sex, Age = @Age, ArrivalDate = @ArrivalDate, " +
    "IsSterilization=@IsSterilization, Description=@Description, PetsImage=@PetsImage, KindId=@KindId " +
      " WHERE PetsId = @PetsId";
  MySqlCommand cmd = new MySqlCommand(query, connection);
  cmd.Parameters.AddWithValue("@PetsName", PetsName);
  cmd.Parameters.AddWithValue("@Breed", Breed);
  cmd.Parameters.AddWithValue("@Sex", Sex);
  cmd.Parameters.AddWithValue("@Age", Age);
  cmd.Parameters.AddWithValue("@ArrivalDate", ArrivalDate.ToString("yyyy-MM-dd HH:mm:ss"));
  cmd.Parameters.AddWithValue("@IsSterilization", IsSterilization);
  cmd.Parameters.AddWithValue("@Description", Description);
  cmd.Parameters.AddWithValue("@PetsImage", PetsImage);
  cmd.Parameters.AddWithValue("@KindId", KindId);
  cmd.Parameters.AddWithValue("@PetsId", PetsId);
  connection.Open();
  cmd.ExecuteNonQuery();
  connection.Close();
  connection.Dispose();
  connection = null;
}
```

Рисунок 3.9– Код методу «DeleteNeuralByNeuralId»

Метод DeleteNeuralByNeuralId відповідає за видалення запису з таблиці "Neural" бази даних за значенням поля NeuralId. Він містить параметр NeuralId, що передається як аргумент методу.

Код методу формує SQL-запит на видалення запису з таблиці "Neural", де значення поля NeuralId дорівнює переданому аргументу. Потім створюється новий екземпляр OleDbConnection та OleDbCommand, в якому передається підготовлений SQL-запит.

Далі викликається метод ExecuteNonQuery(), який виконує SQL-запит на видалення запису з таблиці "Neural". Після цього з'єднання з базою даних закривається.

Даний метод буде використовуватися для видалення запису про нейронну мережу з таблиці "Neural", що є важливим для підтримки актуальності та ефективності роботи функціоналу навчання та використання нейронних мереж в додатку.

Після розробки всіх класів рівня даних було розроблено форми для взаємодії користувача із програмою. Наприклад, розроблена форма для навчання нейронної мережі зображена на рис. 3.10.

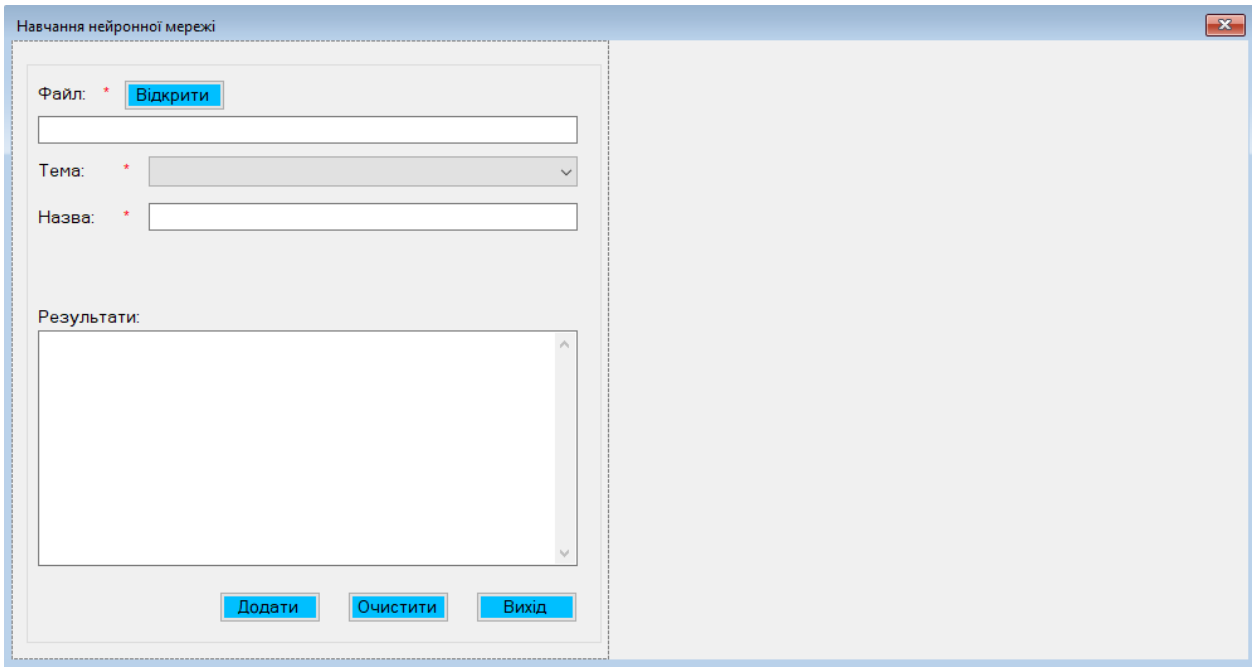


Рисунок 3.10– Розробка форми для тренування нейронної мережі

Під час завантаження форми у її конструкторі викликається метод «LoadAllDate», який завантажує дані про різні теми служби підтримки у випадаючий список (рис. 3.11).

```
1 reference
private void LoadAllDate() {
    _ThemesList = _ThemesProvider.GetAllThemes();
    ThemesCBox.DataSource = _ThemesList;
    ThemesCBox.ValueMember = "ThemesId";
    ThemesCBox.DisplayMember = "ThemesName";
}
```

Рисунок 3.11– Код методу «LoadAllDate»

Також у будь-якій розробленій формі всі введені дані перевіряються на коректність. Для кожної форми написано свій метод «IsDataEnteringCorrect», що перевіряє дані всіх обов'язкових полів. Метод для перевірки даних на коректність форми «TeachingNeuralForm» зображений на рис. 3.12.

```
1 reference
private bool IsDataEnteringCorrect() {
  bool isCorrect = true;
  if (!_IsModelTrain) {
    MessageBox.Show("Неможливо зберегти дані. \r\nЩе не навчено нейронну мережу!","Увага!");
    isCorrect = false;
  }
  if (Convert.ToInt32(ThemesCBox.SelectedValue) > 0) {
    ThemesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    ThemesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataEntering(NeuralNamesTBox.Text)) {
    NeuralNamesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    NeuralNamesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  return isCorrect;
}
```

Рисунок 3.12– Код методу для перевірки коректності введених даних

Метод відповідає за перевірку коректності введених даних у формі додавання даних навчання нейронної мережі. Метод повертає булеве значення, яке вказує на те, чи коректно введені дані.

Код методу містить перевірки на наявність навченої моделі нейронної мережі, введення назви нейронної мережі та вибір теми, до якої буде додана мережа. Якщо якась з перевірок не пройдена, то відповідний лейбл встановлюється на відповідне повідомлення, а метод повертає значення false. Якщо всі перевірки успішно пройдені, метод повертає значення true.

Даний метод використовується для запобігання додаванню некоректних даних у таблицю "Neural" бази даних, що є важливим для підтримки цілісності даних та правильної роботи функціоналу навчання та використання нейронних мереж в додатку.

Для видалення даних про тренування нейронної мереж, реалізована подія «NeuralGridView_CellClick», код якої представлено на рис. 3.13.

```
1 reference
private void NeuralGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
    if (e.ColumnIndex == 4 && NeuralGridView[0, e.RowIndex].Value.ToString() != _NeuralList[0].Message) {
        if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити", MessageBoxButtons.YesNo) == DialogResult.Yes) {
            _NeuralProvider.DeleteNeuralByNeuralId(Convert.ToInt32(NeuralGridView[0, e.RowIndex].Value.ToString()));
            DataLoad();
        }
    }
}
```

Рисунок 3.13– Код події «NeuralGridView_CellClick»

Подія «NeuralGridView_CellClick» відповідає за обробку кліку на комірку таблиці з даними про нейронні мережі.

Код перевіряє, чи був клік на комірку з номером колонки, що дорівнює 4-му (це колонка з кнопкою "Видалити"). Також перевіряється, чи не є значення в першій комірці таблиці рядка заголовком. Якщо обидві перевірки успішно пройдені, то викликається діалогове вікно з запитанням про підтвердження видалення запису про нейронну мережу. Якщо користувач підтверджує видалення, то викликається метод DeleteNeuralByNeuralId() екземпляру класу NeuralProvider з передачею в якості аргументу ідентифікатора нейронної мережі, що визначається за допомогою значення в першій комірці таблиці. Після виконання методу викликається метод DataLoad(), що оновлює дані в таблиці.

Даний код використовується для видалення запису про нейронну мережу з таблиці "Neural" бази даних за допомогою кліку на кнопку "Видалити" у таблиці відповідного вікна додатку.

Отже, в даному підрозділі було проведено частковий опис реалізованої системи.

3.5 Розробка моделі навчання та застосування нейронної мережі

Як було описано вище для навчання нейронної мережі було використано ML.NET (безкоштовний фреймворк машинного навчання). Цей фреймворк може бути використаний для розробки моделей машинного навчання, які можуть бути вбудовані в додатки на .NET.

Щоб підключити ML.NET до WinForms додатку, необхідно спочатку встановити необхідні пакети. Для цього було виконано наступні кроки:

– відкрито проект WinForms у Visual Studio;

– у вікні "Керування пакетами NuGet" (NuGet Package Manager) знайдено та встановлено пакет "Microsoft.ML";

– відкрито файл форми, де додано код для використання ML.NET;

– додано на початку файлу простори імен за допомогою директиви using: Microsoft.ML та Microsoft.ML.Data.

– додано код, який оголошує новий об'єкт MLContext у коді форми: private MLContext context. Ініціалізація даного об'єкту context відбувається при відкритті файлу із даними для навчання нейронної мережі.

Після цього було розроблено 2 класи "Input" та "Output", що показані на рис. 3.14.

```
8 references
public class Input {
    [LoadColumn(0), ColumnName("Question")]
    public string Question;

    [LoadColumn(1), ColumnName("Answer")]
    public string Answer;
}

2 references
public class Output {
    [ColumnName("PredictedLabel")]
    public uint Prediction;

    [ColumnName("Score")]
    public float[] Scores;

    public string PredictedAnswer;
}
```

Рисунок 3.14– Класи для навчання нейронної мережі

Клас Input використовується для представлення вхідних даних для навчання або застосування моделі машинного навчання. Він містить два поля: Question та Answer, які відповідають запитанню та відповіді ~~відповідно~~.

Атрибути [LoadColumn] та [ColumnName] вказують на номер стовпця вхідних даних та назву стовпця відповідно, що необхідно для коректного завантаження та обробки даних моделлю.

Клас Output використовується для представлення результатів передбачення моделі машинного навчання. Він містить три поля: Prediction, Scores та PredictedAnswer.

Атрибут [ColumnName] вказує на назву стовпця з результатами передбачення моделі. Поле Prediction містить передбачене значення відповіді, а поле Scores містить оцінки вірогідності передбачення для кожної можливої відповіді.

Поле PredictedAnswer містить передбачену відповідь, яка може бути використана для подальшої обробки результатів.

Після цього було розроблено подію «OpenBtn_Click», яка викликається при натисканні кнопки «Відкрити» для відкриття файлу із даними для навчання. Оскільки даний метод великий за обсягом, його код приводиться у додатка лістингів програми.

При натисканні відбувається відкриття діалогового вікна для вибору файлу, налаштування його властивостей та відображення. Після вибору файлу здійснюється його зчитування із заданого шляху, розбивка на рядки, поділ на поля та додавання їх до списку datas.

Далі створюється об'єкт MLContext, що використовується для завантаження даних у форматі, зрозумілому для ML.NET. Крім того, дані розділяються на навчальні та тестові. Після цього створюється конвеєр моделі машинного навчання, який складається з наступних етапів:

– FeaturizeText: перетворює текстове поле "Question" на числові ознаки у вигляді векторів;

– ConvertType: перетворює текстове поле "Answer" у категоріальний тип;

– MapValueToKey: замінює категоріальне поле на цілочисельний індекс;

– MapKeyToValue: замінює цілочисельний індекс на відповідний йому текстовий запис;

– SdcaNonCalibrated: використовує алгоритм машинного навчання для навчання моделі;

– MapKeyToValue: замінює цілочисельний індекс, отриманий на попередньому кроці, на відповідний текстовий запис "PredictedAnswer".

Далі підготовлені дані навчаються методом Fit(), результати оцінюються методом Evaluate(), та метрики, такі як мікро та макро точність, виводяться в текстовому полі для звіту RaportTBox. В кінці роботи методу значення змінної _IsModelTrain встановлюється в true, щоб вказати на успішне навчання моделі.

Для збереження даних навчання нейронної мережі було розроблено подію «AddBtn_Click», що спрацьовує при натисканні на кнопку «Додати» (рис. 3.15).

```
1 reference
private void AddBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    //Save model
    string pathName = @"\teach\" + GenerateFileName() + ".zip";
    string localProj = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);
    _NeuralProvider.InsertNeural(NeuralNamesTBox.Text, Convert.ToInt32(ThemesCBox.SelectedValue), pathName);
    context.Model.Save(model, data.Schema, localProj + pathName);
    //context.Model.Save(model, data.Schema, "model.zip");
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Було навчено нейронну мережу " +
      NeuralNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
  }
}
```

Рисунок 3.15– Код події «AddBtn_Click»

Як можна побачити із рис. 3.15 подія "AddBtn_Click" відповідає за додавання навченої моделі в базу даних і збереження її на диск. Спочатку метод перевіряє коректність введених даних за допомогою методу

IsDataEnteringCorrect(). Якщо введені дані коректні, то виконується наступний блок коду.

Створюється шлях для збереження моделі у вигляді архіву з розширенням .zip в папці "teach" у директорії запущеної програми. Потім викликається метод InsertNeural() з об'єктом NeuralProvider, який відповідає за взаємодію з таблицею Neural у базі даних, щоб додати запис про навчену модель зі значеннями: назвою нейронної мережі, ідентифікатором теми та створеним шляхом до збереження моделі.

Далі за допомогою методу context.Model.Save() модель зберігається на диск у вигляді архіву з розширенням .zip у відповідну папку. Потім виконується метод ClearAllData(), який очищує всі дані, введені користувачем у формі, тобто змінні. Запис про навчену модель також додається до журналу за допомогою методу InsertLogs() з об'єктом LogsProvider. На останок, з'являється повідомлення "Дані успішно збережено!" за допомогою методу MessageBox.Show().

3.6 Експериментальна перевірка\тестування

Експериментальна перевірка або тестування - це один з ключових етапів розробки програмного продукту, який дозволяє перевірити, чи працює програма належним чином, чи відповідає вона вимогам технічного завдання і чи задовольняє вона потреби користувачів.

У випадку нашого проекту, експериментальна перевірка полягала у тестуванні навченої моделі на різних тестових вибірках даних, щоб переконатися, що вона правильно працює і дозволяє відповідати на запитання користувачів.

Для тестування нейронної мережі було використано раніше зібрані дані з попередніх запитань і відповідей, що були збережені в CSV файлі (рис. 3.16).

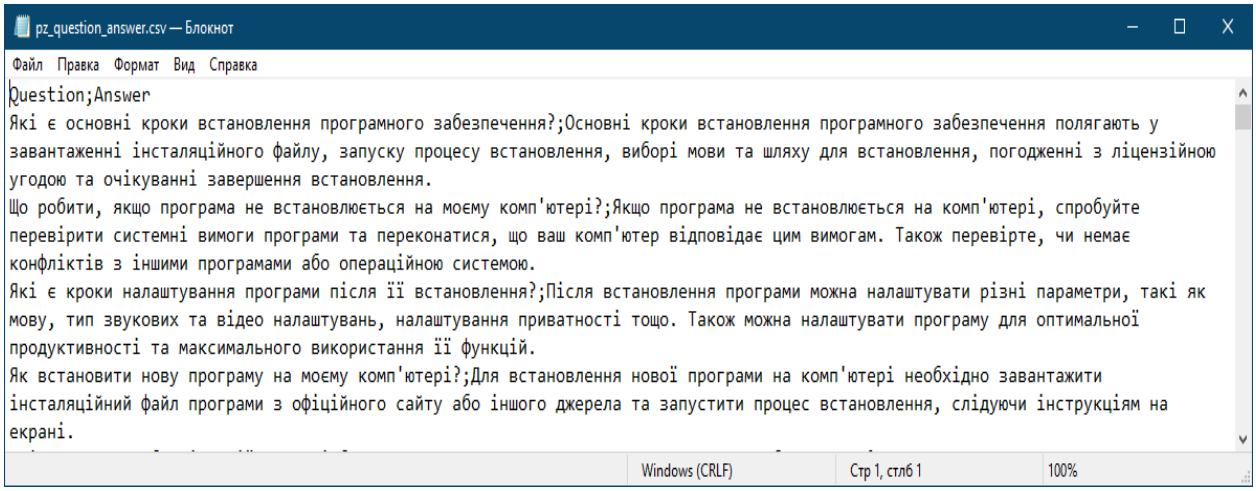


Рисунок 3.16– Фрагмент вікна із даними навчання

Після цього у програмі було вибрано цей файл та проведено навчання нейронної мережі (рис. 3.17).

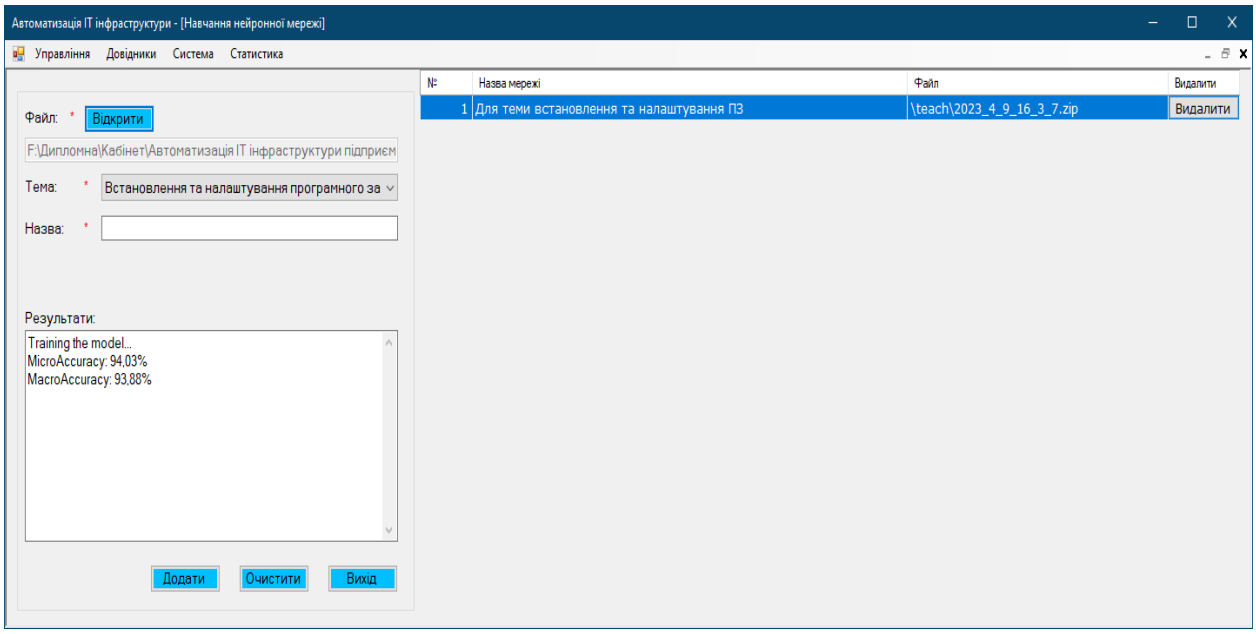


Рисунок 3.17– Форма із результатом навчання нейронної мережі

Після відкриття файлу мережа була навчена за допомогою вищевказаного методу. Процес навчання відображається знизу у лівій частині вікна. Як можна побачити із рис. 3.17 були розраховані метрики для оцінки точності навчання.

Метрика MicroAccuracy обчислюється шляхом підрахунку загальної кількості правильно класифікованих прикладів і ділення її на загальну кількість прикладів. MacroAccuracy обчислюється шляхом підрахунку точності для кожного класу і взяття середнього значення.

MicroAccuracy у даному випадку дорівнює 94,03%, що означає, що модель правильно класифікувала 94,03% всіх прикладів в тестовому наборі. Кількість прикладів у тестовому наборі складає 285 шт. MacroAccuracy дорівнює 93,88%, що означає, що середня точність для кожного класу становить 93,88%. Ці метрики є досить високими, що свідчить про високу точність моделі. Однак, їхнє значення може залежати від конкретної задачі та даних, на яких модель була навчена та перевірена.

Після навчання було проведено експериментальні тести, використовуючи тестову вибірку даних, щоб оцінити точність навченої моделі нейронної мережі. Для цього було задано декілька запитань, на які було получено відповідь від нейронної мережі (рис. 3.18). Запитання задавались із використанням скорочених назв слів та допусканням різноманітних помилок у запитаннях.

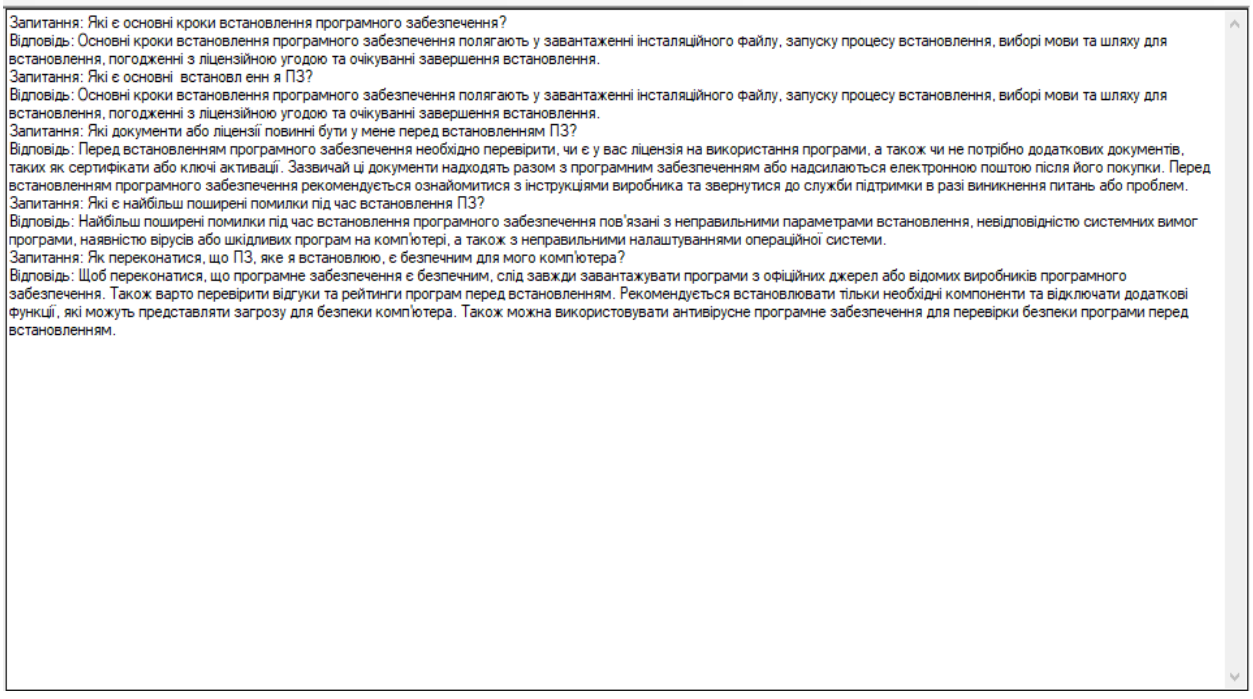


Рисунок 3.18– Проведення експериментів із заданими запитаннями

Із проведеного експерименту було зроблено висновок, що навчена нейронна мережа чітко та без помилок відповідала на поставлені запитання. Проте, якщо задавати запитання, які не стосуються вибраної теми чи запитання, відповідь на яке не було визначено у тестовому наборі даних для навчання, вона підбиратиме відповідь не коректно.

3.7 Інструкція користувача

Перед тим, як розпочати роботу з додатком "Автоматизація IT інфраструктури", необхідно запустити його. Після цього на екрані з'явиться вікно авторизації, в якому користувач повинен ввести свій логін та пароль (рис. 3.19). Це дозволяє ідентифікувати користувача та забезпечити безпеку його персональних даних.

Після введення логіну та пароля користувачу буде надано доступ до головного інтерфейсу додатку, де він зможе виконувати різноманітні дії в залежності від його ролі у системі. Таким чином, для початку роботи з додатком "Автоматизація IT інфраструктури" необхідно пройти процес авторизації та отримати доступ до головного інтерфейсу додатку, щоб мати можливість працювати зі всіма його функціями.

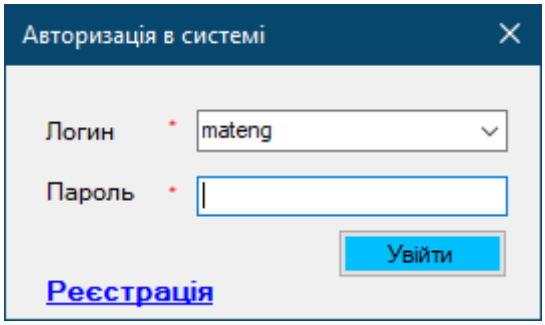


Рисунок 3.19 – Авторизація користувача в системі

У додатку "Автоматизація IT інфраструктури" користувач може зареєструватися, якщо він не має облікового запису. Для цього користувач

повинен натиснути кнопку "Реєстрація", яка знаходиться на екрані авторизації. Після цього користувач буде перенаправлений на вікно реєстрації, яке містить форму для введення необхідної інформації. У формі реєстрації користувач повинен ввести своє ім'я, прізвище, логін та пароль. Після заповнення всіх полів користувач повинен натиснути кнопку "Зареєструватися" (рис. 3.20). Після успішної реєстрації користувач може використовувати свій новий обліковий запис для входу в систему.

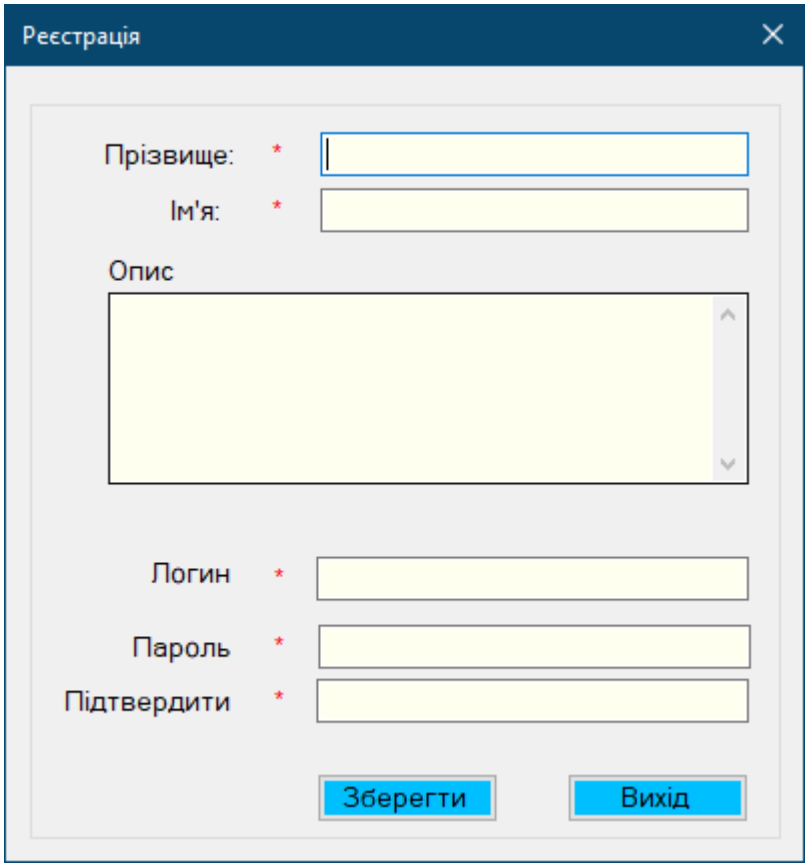


Рисунок 3.20 – Форма реєстрації користувачів

Якщо у програмі ще не зареєстровано жодного користувача, то при реєстрації першому користувачу автоматично буде присвоєна роль системного адміністратора, іншим всім користувачам буде присвоєно ролі звичайних користувачів.

Для швидкого пошуку імені в програмі можна скористатись випадаючим списком, який містить доступні імена користувачів. Крім того,

при введенні імені з клавіатури програма автоматично поставить його у верхню частину списку, що спрощує інтерфейс і дозволяє швидше знайти потрібне ім'я.

Після успішної автентифікації користувача системи буде відкрите головне вікно програми (рис.3.21). Це вікно містить основне меню, що складається з доступних опцій та функцій програми. Наприклад, користувач може відкривати різні форми програми, переглядати інформацію про користувачів, додавати різноманітні теми для служби підтримки, проводити навчання нейронних мереж, тестувати нейромережі задаючи їм різноманітні запитання, формувати статистику, тощо

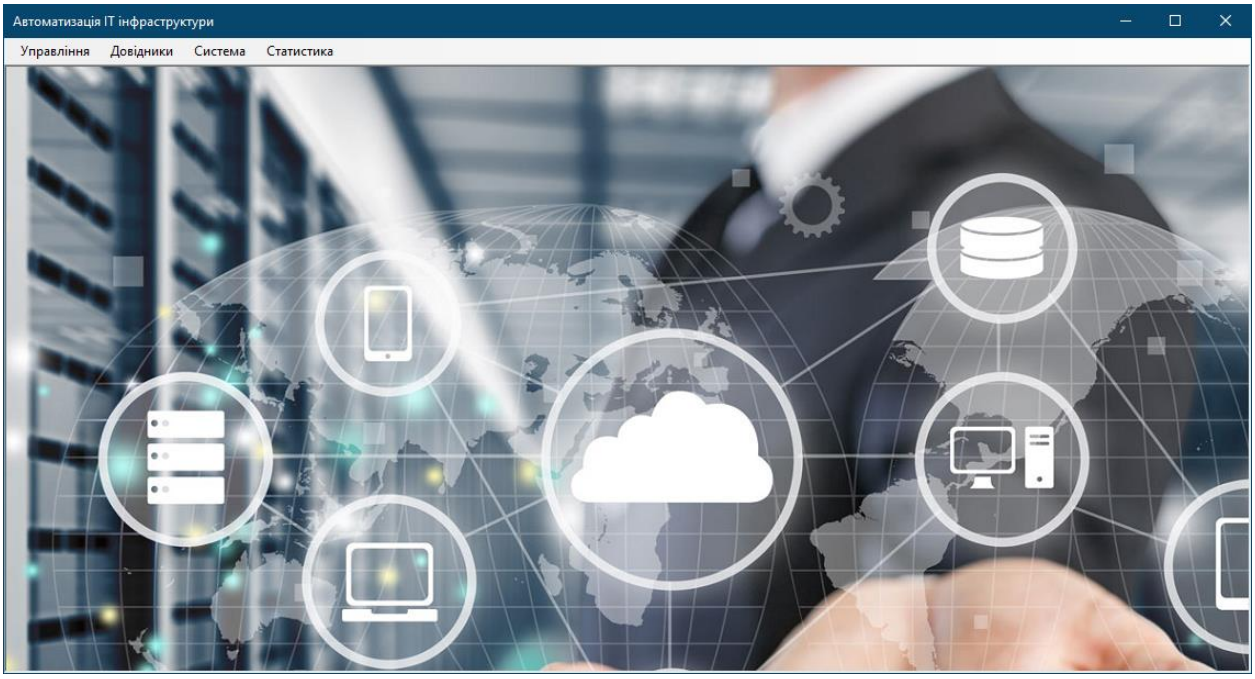


Рисунок 3.21 – Головне меню програми

У системі реалізовано дві ролі: системного адміністратора та звичайного користувача. Роль системного адміністратора дозволяє керувати: обліковими записами та переглядати події в системі.

Для початку роботи із системою необхідно додати інформацію про різноманітні теми для служби підтримки користувачів та провести навчання нейронних мереж.

Щоб додати інформацію про нову тему, користувачу системи необідно перейти по меню програми «Довідники» –> «Теми», після чого відкриється форма, що зображено на рис. 3.22.

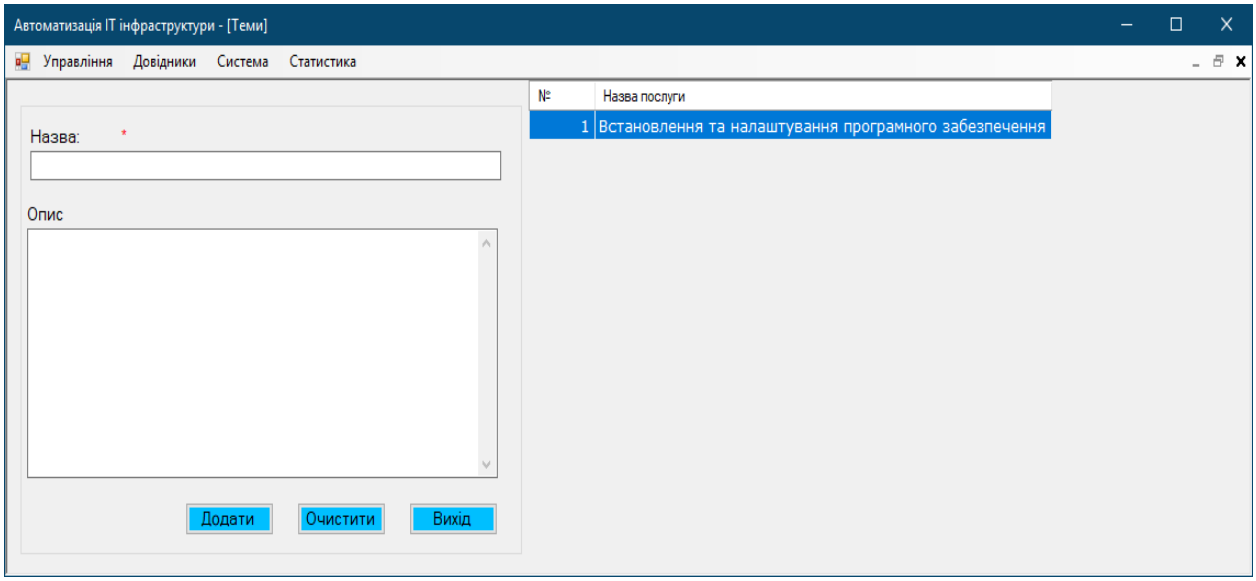


Рисунок 3.22 – Форма для опрацювання тем служби підтримки

Також, збережені дані тем можна змінювати, для цього у правій частині форми необхідно вибрати відповідну тему, після чого відкриється форма, що представлена на рис. 3.23.

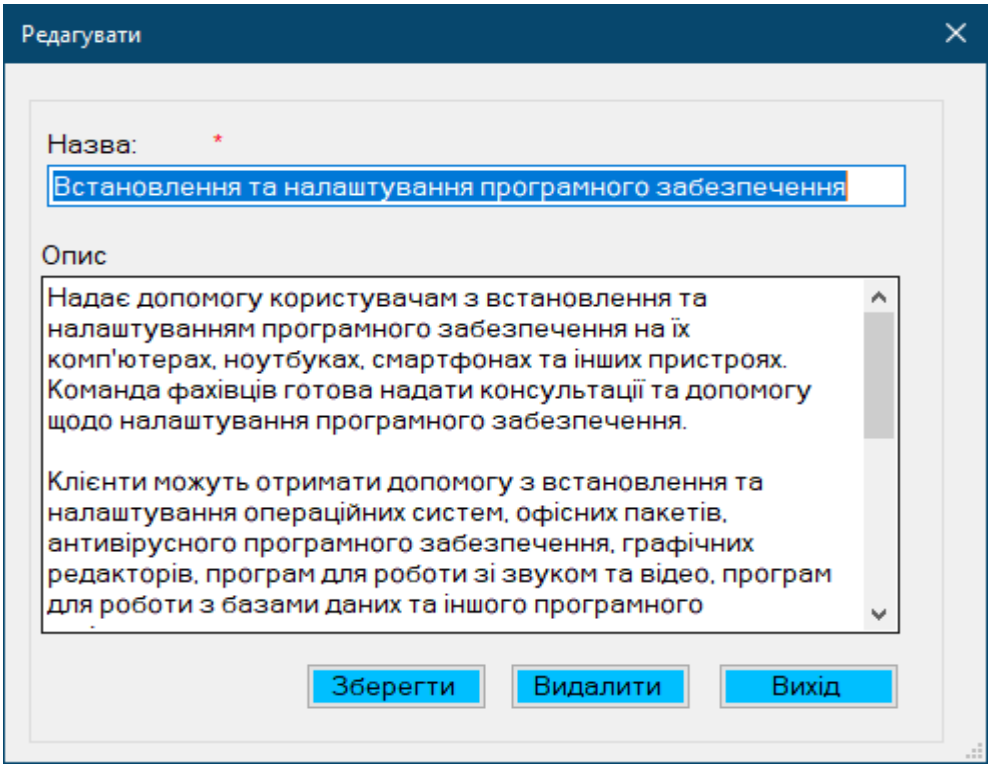


Рисунок 3.23 – Форма для редагування інформації вибраної теми

Для того, щоб провести навчання нейронної мережі, користувач повинен перейти до відповідного меню програми «Довідники» та вибрати опцію «Навчання нейронної мережі». Після цього на екрані з'явиться вікно, яке містить форму для проведення навчання нейронної мережі (див. рис. 3.24). У цій формі користувачу необхідно вибрати файл із збереженими даними для начання, після чого система автоматично проведе навчання та покаже його результати. Якщо результати навчання будуть задовільняти користувача, він може зберегти проведене навчання. Для цього йому потрібно задати назву для нейронної мережі, вибрати тему навчання та натиснути кнопку «Додати». Після збереження даних програма виведе повідомлення.

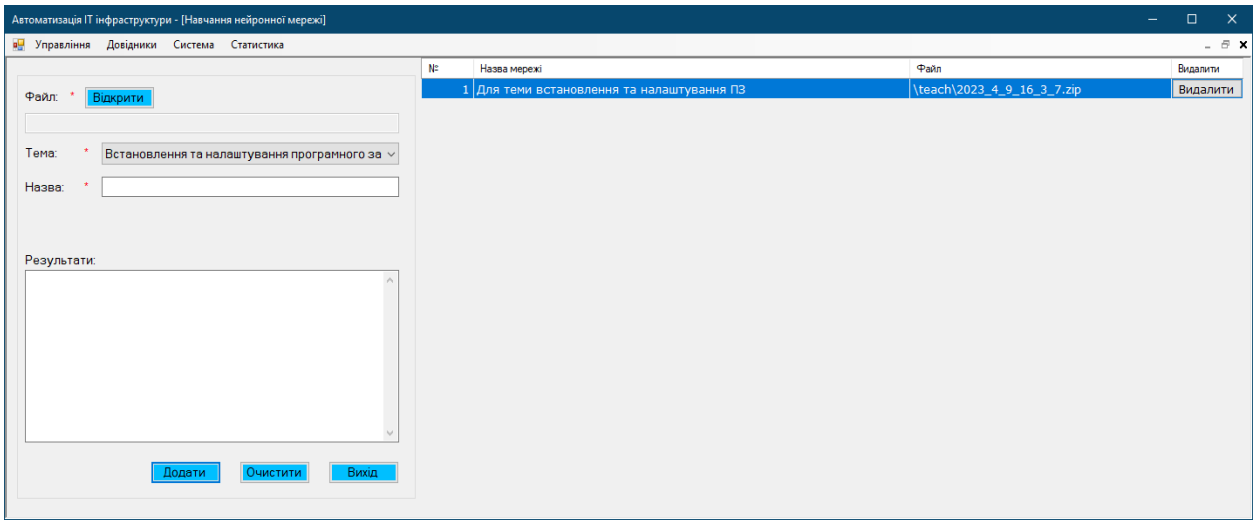


Рисунок 3.24 – Форма для проведення навчання нейронної мережі

Після збереження моделі навчання, користувач зможе перевірити її можливість надавання відповідей на запитання. Для цього йому необхідно перейти по меню програми «Управління» –> «Надання послуг» (рис. 3.25).

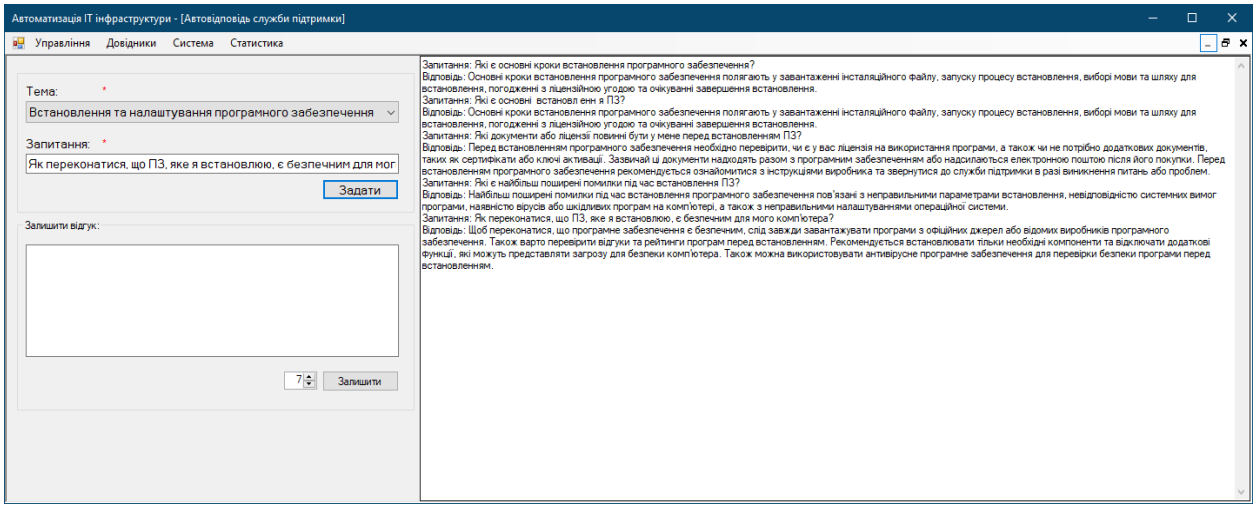


Рисунок 3.25 – Проведення процедури задавання запитань

Користувач може вибрати тему розмови та ввести своє запитання в відповідне поле. Ввівши запитання потрібно натиснути кнопку «Задати», після чого нейронна мережа зразу ж відповість на нього. При введенні запитання можуть допускатись помилки, та слід пам’ятати, що надавання відповідей, яких не було у наборі даних для навчання може бути не коректрим. Після відповіді користувач зможе її оцінити та залишити коментар.

Також, у програмі реалізовано формування статистики по результатах проведення запитань. Для цього можна скористатись пунктами меню «По темі запитань» та «По даті». У першому випадку буде фрормовано статистику по заданих запитаннях із вибраної теми та обчислено середній рейтинг відповіді по результатах оцінки користувачів (рис. 3.26).

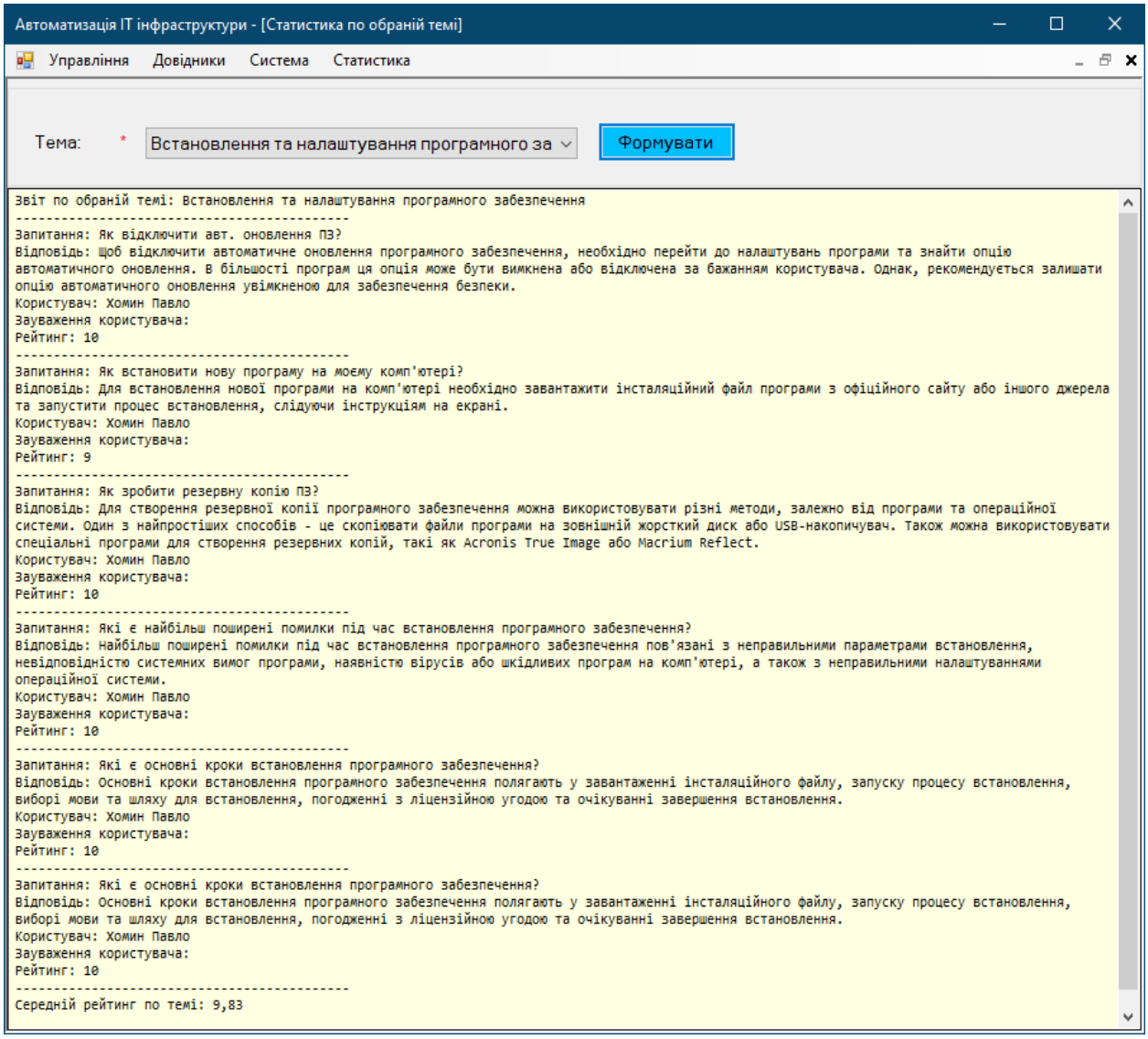


Рисунок 3.26 – Формування статистики по вибраній темі запитань

Скориставшись формою для формування статистики за вибраний період часу, будуть виведені всі записи по проведених діалогах користувачів.

Управління обліковими записами є важливою складовою багатьох систем, в тому числі і у додатку "Автоматизація IT інфраструктури". Користувач з роллю "Системний адміністратор" має повний доступ до управління обліковими записами системи.

Для того, щоб користувач з роллю "Системний адміністратор" міг управляти обліковими записами інших користувачів, він повинен виконати певні кроки. На екрані головного меню додатку він повинен обрати опцію

"Користувачі" (Меню «Система» - «Користувачі»), після чого відкриється вікно управління обліковими записами (рис. 3.27).

У цьому вікні користувач з роллю "Системний адміністратор" може переглядати список існуючих облікових записів користувачів, редагувати їхню інформацію, створювати нові облікові записи та видаляти старі. При редагуванні або створенні нових облікових записів, користувач має змогу встановити різні параметри, такі як ролі, права доступу та обмеження на роботу з додатком.

Важливо зазначити, що управління обліковими записами є дуже важливою функцією додатку, і потребує обережності та відповідальності в її використанні. Тільки відповідальний підхід до управління обліковими записами може забезпечити безпеку та надійність роботи системи.

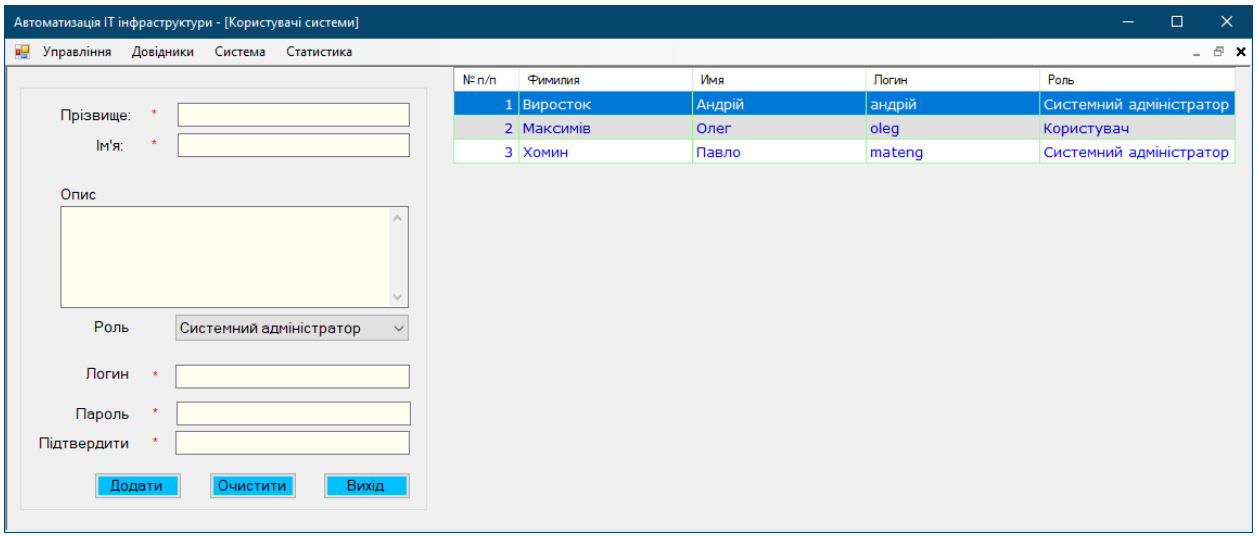
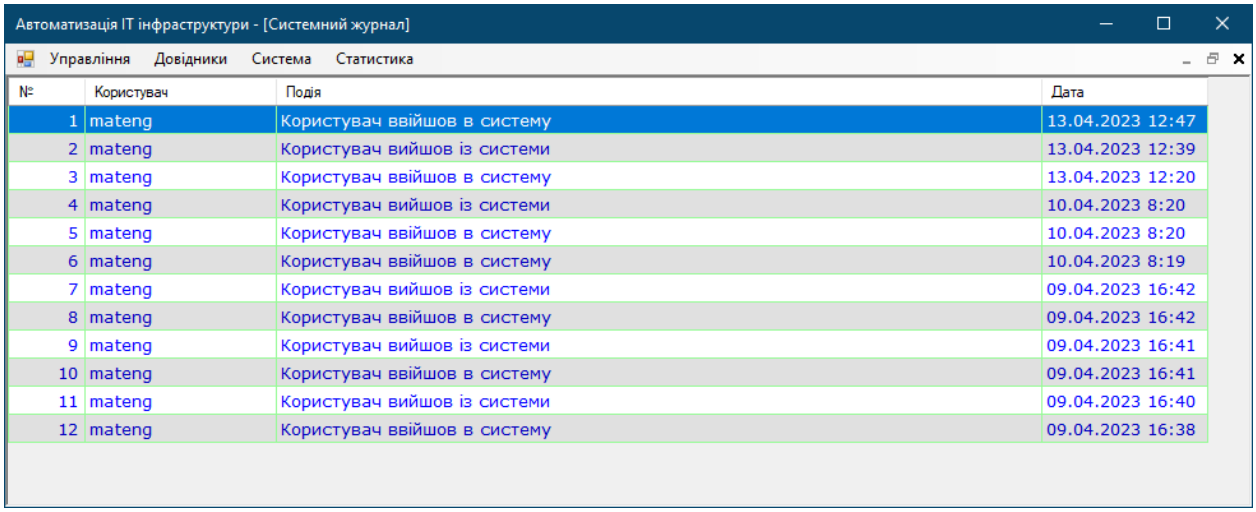


Рисунок 3.27 – Управління обліковими записами

В системі можна відстежувати активність користувачів та їх дії завдяки обліковому запису з правами адміністратора. Це можливо зробити за допомогою функції "Події ", яка знаходиться в меню "Система" (рис. 3.28).



№	Користувач	Подія	Дата
1	mateng	Користувач ввійшов в систему	13.04.2023 12:47
2	mateng	Користувач вийшов із системи	13.04.2023 12:39
3	mateng	Користувач ввійшов в систему	13.04.2023 12:20
4	mateng	Користувач вийшов із системи	10.04.2023 8:20
5	mateng	Користувач ввійшов в систему	10.04.2023 8:20
6	mateng	Користувач ввійшов в систему	10.04.2023 8:19
7	mateng	Користувач вийшов із системи	09.04.2023 16:42
8	mateng	Користувач ввійшов в систему	09.04.2023 16:42
9	mateng	Користувач вийшов із системи	09.04.2023 16:41
10	mateng	Користувач ввійшов в систему	09.04.2023 16:41
11	mateng	Користувач вийшов із системи	09.04.2023 16:40
12	mateng	Користувач ввійшов в систему	09.04.2023 16:38

Рисунок 3.28 – Події системи

Інформація в системному журналі може бути дуже корисною для виявлення проблем з безпекою та знаходження вразливостей в системі. Також, вона може бути використана для відстеження дій користувачів та їх контролю. Користувачі можуть переглядати журнал подій для відстеження дій інших користувачів в системі, зокрема з метою забезпечення безпеки в цифровому просторі.

Для зміни облікових даних, користувачі можуть скористатися меню "Персоналізація", де вони можуть змінити своє ім'я, пароль та інші особисті дані. Таким чином, користувач може змінити свої дані в будь-який момент, якщо це необхідно.

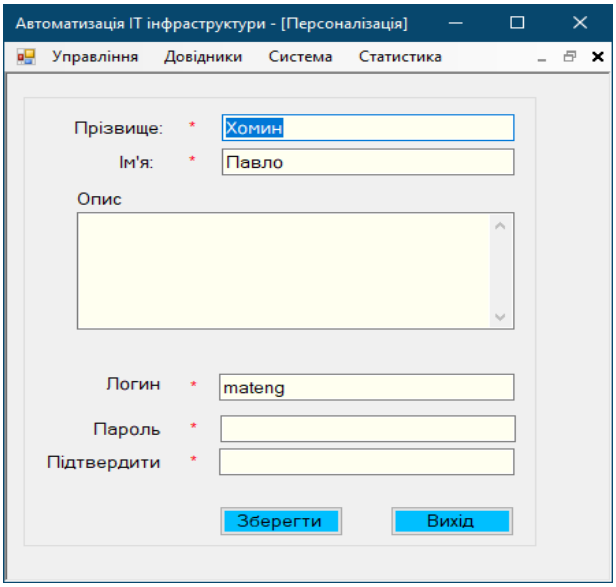


Рисунок 3.29 – Зміна даних персоналізації користувача

Щоб вийти з програми, користувач повинен перейти по меню "Управління" та вибрати опцію "Вихід". Це забезпечить безпечне та правильне закриття додатку та запобіжить втраті даних та можливих проблем з безпекою.

3.8 Опис процедури розгортання програмного забезпечення

Опис процедури розгортання програмного забезпечення у вигляді автономного застосунку можна розбити на дві основні частини: створення каталогу та копіювання застосунку, а також запуск програми на комп'ютері користувача.

1. Створення каталогу. На цільовому диску користувач повинен створити новий каталог, який буде використовуватися для розгортання програмного забезпечення. Назва каталогу може бути будь-якою, залежно від вимог користувача. Наприклад, користувач може створити каталог під назвою «Автоматизація IT інфраструктури».

2. Копіювання застосунку. Для розгортання застосунку необхідно скопіювати каталог «Debug» з проекту на цільовий диск, який був створений

на попередньому етапі. Цей каталог містить всі необхідні файли для запуску програми.

3. Запуск програми. Після копіювання каталогу «Debug», користувач може запустити програму на своєму комп'ютері. Це можна зробити одним зі стандартних способів в операційній системі Windows: подвійним клацанням лівою кнопкою миші на ярлику програми, викликом контекстного меню з вибором пункту «Відкрити» або натисканням кнопки «Пуск» на панелі завдань та подальшим вибором пункту «Усі програми» та подвійним клацанням лівою кнопкою миші на ярлику програми. Після запуску програма буде готова до використання.

3.9 Висновок

В даному розділі було описано процес вибору засобів розробки, включаючи вибір архітектурного шаблону, мови програмування та СУБД. Проектування та розробка бази даних, архітектура проекту і розробка модулів були проведені згідно вимог завдання на кваліфікаційну роботу. Було розроблено модель навчання та застосовано нейронну мережу для автоматичної відповіді на запитання користувачів.

Експериментальна перевірка та тестування показали, що розроблений продукт може ефективно вирішувати поставлені завдання, забезпечуючи достатній рівень точності відповідей на запитання. Для забезпечення зручного користування продуктом, було розроблено інструкцію користувача, а також опис процедури розгортання програмного забезпечення на комп'ютері користувача у вигляді автономного застосунку.

Таким чином, в рамках кваліфікаційної роботи було реалізовано функціональний продукт, який може ефективно використовуватися для автоматизації обробки запитань користувачів у сфері IT інфраструктури.

ВИСНОВКИ

У ході проведення дослідження було розглянуто кілька важливих аспектів автоматизації ІТ інфраструктури підприємства за допомогою машинного навчання та нейронних мереж і розроблено систему ~~для~~ підтримки прийняття рішень при роботі ІТ інфраструктури підприємства.

Спочатку було проведено проектування системи підтримки прийняття рішень при експлуатації ІТ інфраструктури підприємства. Досліджено поняття ІТ-інфраструктури, її завдання та особливості, акторів та функцій системи, а також проаналізовано аналогічні програмні рішення. Це допомогло встановити мету та завдання дослідження та чітко сформулювати постановку задачі.

Далі було здійснено перехід до вивчення методів та моделей прогнозування даних служби підтримки користувачів. Було проведено аналітичний огляд існуючих методів розв'язання задачі, таких як багатошаровий персептрон (MLP), узагальнені регресійна нейронна мережа (GRNN) та радіально-базисна мережа (RBN). Було вивчено структуру нейронної мережі та метод навчання нейронної мережі за допомогою бібліотеки ML .NET. Також було розроблено алгоритм процесу аналізу та прогнозування на основі бібліотеки ML .NET.

На наступному етапі було проведено розробку програмного забезпечення. Насамперед, було визначено засоби розробки, вибрано архітектурний шаблон, мову програмування та СУБД. Здійснено проектування та розробку бази даних, а також архітектури проекту. Проведено розробку модулів проекту, створено модель навчання та застосування нейронної мережі. На етапі експериментальної перевірки та тестування було проведено тестування розробленої моделі та перевірено границі застосовності системи. Це дозволило виявити можливі слабкі місця та відпрацювати стратегії оптимізації.

Після того, як модель була успішно протестована, розроблено інструкцію користувача, що допоможе ефективно впровадити систему на підприємстві. Останнім етапом стала процедура розгортання програмного забезпечення, яка включала в себе налаштування системи та підготовку до впровадження.

У результаті проведених досліджень було розроблено прототип автоматизованої системи підтримки прийняття рішень для IT інфраструктури підприємства, яка базується на машинному навчанні та нейронних мережах. Застосування такої системи дозволить автоматизувати обробку запитів користувачів. Впровадження цієї системи може знизити час очікування користувачів та підвищити ефективність роботи служби підтримки.

Таким чином, можна стверджувати, що спроектована система підтримки прийняття рішень при функціонуванні IT інфраструктури підприємства за допомогою машинного навчання та нейронних мереж відповідає висунутим вимогам та може стати важливим інструментом для оптимізації роботи IT-відділів різних підприємств. Завдяки успішному виконанню завдань та досягненню мети дослідження, можна вважати, що робота має значний практичний потенціал та може сприяти розвитку IT-інфраструктури на підприємствах різних галузей та рівнів. Основні переваги впровадження розробленої системи включають:

– підвищення продуктивності IT-підрозділів завдяки автоматизації рутинних процесів та зменшенню часу на обробку запитів;

– забезпечення більш точного прогнозування та превентивного ремонту обладнання, завдяки використанню алгоритмів машинного навчання та нейронних мереж;

– зменшення витрат на підтримку та розвиток IT-інфраструктури завдяки оптимізації ресурсів та підвищенню ефективності роботи співробітників.

Система автоматизації ІТ інфраструктури підприємства за допомогою машинного навчання та нейронних мереж може бути використана у таких галузях, як:

– фінансові установи: банки, страхові компанії, інвестиційні фонди. Застосування системи дозволить оптимізувати процеси обслуговування клієнтів, вирішення проблем та моніторингу ІТ-інфраструктури;

– телекомунікаційні компанії: провайдери інтернету, мобільні оператори, компанії зв'язку. Підтримка прийняття рішень при функціонуванні ІТ інфраструктури допоможе краще розуміти потреби користувачів та відповідно реагувати на виникаючі проблеми;

– органи державної влади та місцевого самоврядування. Застосування системи може поліпшити роботу ІТ-служб, що підтримують електронні сервіси для громадян та бізнесу;

– освітні установи: університети, коледжі, школи. Використання розробленої системи допоможе автоматизувати обробку запитів від студентів та викладачів, а також полегшити роботу адміністративного персоналу;

– медичні установи: лікарні, клініки, діагностичні центри. Впровадження системи підтримки прийняття рішень при функціонуванні ІТ інфраструктури сприятиме підвищенню якості обслуговування пацієнтів та оптимізації роботи медичного персоналу;

– промислові підприємства: заводи, фабрики, дослідницькі центри. Застосування системи підтримки прийняття рішень при експлуатації ІТ інфраструктури на промислових підприємствах дозволить поліпшити роботу ІТ-служб, що підтримують технологічні процеси, та забезпечити вчасне виявлення та усунення проблем, пов'язаних з роботою обладнання;

– роздрібна торгівля та електронна комерція: магазини, гіпермаркети, онлайн-платформи. Підтримка прийняття рішень при функціонуванні ІТ інфраструктури в цих галузях сприятиме швидкому

вирішенню питань, пов'язаних з операціями, логістикою та роботою з клієнтами;

– транспортні компанії: авіаперевізники, залізничні компанії, автоперевізники. Використання системи дозволить оптимізувати роботу IT-відділів, забезпечуючи надійність інформаційних систем та підтримку користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ковальчук С.М., Гончаренко Ю.В. Інформаційні системи та технології в управлінні організаціями: Навчальний посібник. - К.: Видавництво "Сімейний лікар", 2019. - 512 с.

2. Волощук В.П., Григор'єва Л.М. ІТ-інфраструктура компанії: структура, компоненти та роль в сучасному бізнесі. - К.: Ліра, 2021. - 408 с.

3. Tanenbaum A.S., Van Steen M. Distributed Systems: Principles and Paradigms. - 2nd Edition. - New Jersey: Prentice Hall, 2006. - 704 p.

4. Білоконенко В.І., Мельниченко О.С. Організація та управління ІТ-інфраструктурою: методи, техніки та підходи. - Львів: Астролябія, 2020. - 372 с.

5. Morris T., Wu S., Joseph A. Enterprise Systems for Management. - 2nd Edition. - New York: Pearson, 2012. - 480 p.

6. Шевченко С.І., Костюк О.М. Інформаційно-комунікаційна інфраструктура підприємства: поняття, архітектура та управління. - Харків: ХНУ імені В. Н. Каразіна, 2018. - 320 с.

7. Patterson D.A., Hennessy J.L. Computer Organization and Design: The Hardware/Software Interface. - 5th Edition. - San Francisco: Morgan Kaufmann, 2013. - 800 p.

8. Малиновський Є.О., Лобач М.В. ІТ-сервіси для управління корпоративними ресурсами: поняття, класифікація та практичне застосування. - Одеса: ОДУ імені І.І. Мечникова, 2019. - 456 с.

9. Laudon K.C., Laudon J.P. Management Information Systems: Managing the Digital Firm. - 15th Edition. - New York: Pearson, 2017. - 672 p.

10. Савчук В.А., Дубровіна Л.П. Основи інтеграції ІТ-інфраструктури в стратегічне управління підприємствами. - Тернопіль: Економічна думка, 2020. - 384 с.

11. Краснопьоров О.М., Черненко В.І. Інформаційна безпека в IT-інфраструктурі: технології, стандарти та рекомендації. - Дніпро: Пороги, 2017. - 400 с.

12. Stallings W. Data and Computer Communications. - 10th Edition. - New York: Pearson, 2013. - 912 p.

13. Литвинов О.Ю., Ярошенко І.В. Управління IT-інфраструктурою: методики, процеси та кращі практики. - Вінниця: Вінницький політехнічний інститут, 2018. - 368 с.

14. White T. Hadoop: The Definitive Guide. - 4th Edition. - Sebastopol: O'Reilly Media, 2015. - 768 p.

15. Захарченко В.А., Шевців І.М. Хмарні технології в IT-інфраструктурі сучасних підприємств: концепції, застосування та переваги. - Черкаси: Брама-Україна, 2021. - 416 с.

16.Fitzpatrick J., McGurren P. Zendesk: Руководство по внедрению и настройке системы обслуживания клиентов. - Москва: Питер, 2020. - 224 с.

17. Stafford J. Zendesk for Dummies: A Comprehensive Guide to Streamlining Customer Support with Zendesk. - Hoboken: John Wiley & Sons, 2019. - 360 p.

18. Brown M., Fitzgerald R. HappyFox: A Practical Guide to Optimizing Customer Support with HappyFox Help Desk Software. - Austin: Independent Publishing, 2021. - 220 p.

19. Шаповал О.І., Костенко В.П. Впровадження та налаштування HappyFox для підтримки клієнтів: Практичний довідник. - Київ: Український видавничий дім, 2022. - 248 с.

20. Громов О.В., Савельєв І.С. Freshdesk: Керівництво з впровадження та налаштування системи підтримки клієнтів. - Харків: Комп'ютерна література, 2020. - 240 с.

21. Kapoor A. Mastering Freshdesk: A Comprehensive Guide to Streamline Customer Support and Optimize Help Desk Operations. - New York: Apress, 2021. - 280 p.

22. Румельхарт Д. Е., МакКлеланд Дж. Л. Распараллеленные распределенные представления. М.: Издательство «Мир», 1988. – 512 с.

23. Хайкин С. Нейронные сети: полный курс. 2-е изд. М.: Вильямс, 2006. – 1104 с.

24. Лугер Дж. Ф. Искусственный интеллект: современный подход. М.: Издательский дом «Вильямс», 2006. – 960 с.

25. Горбань А. Н., Кечин С. В., Ладыженская Т. А. Багатошаровий персептрон: Теорія та практика. Новосибірськ: Наука, 2002. – 382 с.

26. Терехін В. В., Шматков С. О. Використання багатошарового персептрона в задачах розпізнавання образів: Навчальний посібник. Харків: ХНУ імені В. Н. Каразіна, 2011. – 175 с.

27. Співак Г. М., Малишев В. В. Узагальнена регресійна нейронна мережа: теорія та практика. Львів: Львівська політехніка, 2009. – 256 с.

28. Мастеровой В. А., Березовський М. О. Методи аналізу даних на основі узагальненої регресійної нейронної мережі. Москва: Фізматліт, 2013. – 248 с.

29. Калганов О. В., Мельник І. В. Прогнозування та оптимізація за допомогою узагальненої регресійної нейронної мережі. Київ: Видавництво "Наукова думка", 2011. – 320 с.

30. Лебедєв В. Є., Гринько Ю. О. Нейронні мережі та узагальнена регресійна модель: застосування в економіці та управлінні. Санкт-Петербург: Наука, 2016. – 275 с.

31. Поліщук В. Ф., Калінін О. М. Узагальнена регресійна нейронна мережа в задачах прогнозування виробництва: Навчальний посібник. Харків: ХНУ імені В. Н. Каразіна, 2015. – 210 с.

32. Байсеркеев Б. К., Рахметов Т. Р. Радіально-базисні мережі в задачах обробки інформації: Навчальний посібник. Алмати: Казахський національний університет імені Аль-Фарабі, 2010. – 180 с.

33. Смолянінов І. П., Мельник Л. Г. Радіально-базисні мережі в задачах розпізнавання образів. Москва: Фізматліт, 2008. – 240 с.

34. Вітковський В. М., Міщенко Т. В. Радіально-базисні нейронні мережі для прогнозування економічних показників: Навчальний посібник. Київ: Видавництво "Наукова думка", 2014. – 300 с.

35. Михайлов В. О., Бондаренко С. О. Радіально-базисні мережі в задачах обробки сигналів та систем. Санкт-Петербург: Наука, 2012. – 268 с.

36. Паламарчук С. В., Жадан О. М. Радіально-базисні мережі та їх застосування в задачах класифікації: Навчальний посібник. Харків: ХНУ імені В. Н. Каразіна, 2016. – 220 с.

37. Шут Х., Левчук Л. В. ML .NET: Практичний посібник для розробників. Київ: Видавництво "Наукова думка", 2020. – 340 с.

38. Гудфеллоу Я., Бенджіо Й., Курвіль В. Глибоке навчання: Навчальний посібник. Київ: Видавництво "Наукова думка", 2018. – 500 с.

39. Франконе Ч., Пател А. Microsoft ML .NET: Машинне навчання для .NET розробників. Москва: ДМК Прес, 2019. – 280 с.

40. Захарченко В. П., Куліков М. О. Основи нейронних мереж та машинного навчання: Навчальний посібник. Харків: ХНУ імені В. Н. Каразіна, 2017. – 380 с.

41. Вінсент П., Ларочель Х., Бенджіо Й. Представлення навчання: Прогнозування машинного навчання. Санкт-Петербург: Пітер, 2016. – 420 с.

42. Фаулер М. Шаблони корпоративних застосунків: Навчальний посібник. Київ: Видавництво "Наукова думка", 2018. - 480 с.

43. Ньюман С. Мікросервіси: Проектування, розробка, розгортання. Київ: Видавництво "Наукова думка", 2019. - 350 с.

44. Басс Л., Клементс П., Казман Р. Архітектура програмного забезпечення: Системний підхід. Харків: ХНУ імені В. Н. Каразіна, 2016. - 576 с.

45. Річтер Дж., Троелсен Е. CLR via C#: Програмування на платформі Microsoft .NET Framework 4.5 на мові C#. Київ: Видавництво "Наукова думка", 2016. - 896 с.

46. Васильєв В.І., Павленко Ю.В. Інформатика. Основи програмування на Java. Київ: Видавництво "Наукова думка", 2017. - 300 с.

47. Васильєв А.Н. Чистий код на C++. Київ: Видавництво "Наукова думка", 2019. - 400 с.

48. Петкова М., Шевчук В. Microsoft SQL Server: Адміністрування, програмування, оптимізація. Київ: Видавництво "Наукова думка", 2018. - 600 с.

49. Фейнман С. Oracle Database: Адміністрування, проектування та розробка. Одеса: Видавництво "Наукова книга", 2017. - 550 с.

50. Роббінс С., Форбс Дж. Microsoft Access: Розробка баз даних, звіти, аналітика. Львів: Видавництво "Літера", 2019. - 480 с.