НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

"На правах рукопису"
УДК *004.89*

ДО ЗАХИСТУ ДОПУЩЕНО
Завідувач кафедри

_____ Олександр РОЛІК

“ ____ ” _____ 2023 р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за освітньо-науковою програмою

«*Інформаційне забезпечення робототехнічних систем*»

зі спеціальності 126 «*Інформаційні системи та технології*»

на тему:

«Система моніторингу та оцінювання психологічного стану людини»

Виконала:
студентка VI курсу, групи **ІК-11мн**
Сергієнко Поліна Анатоліївна _____

Керівник:
доцент, к.т.н
Крилов Євген Володимирович _____

Рецензент:
завідувач кафедри СП д.т.н., проф.,
Мухін Вадим Євгенійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2023 року

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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-наукова програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Олександр РОЛІК

«__» _____2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Сергієнко Поліні Анатоліївні

1. Тема дисертації «Система моніторингу та оцінювання психологічного стану людини»

науковий керівник дисертації Крилов Євген Володимирович, к.т.н., доцент, затверджені наказом по університету від «20» березня 2023 р. № 1275с

2. Строк подання студентом дисертації “ 12 ” травня 2023 р.

3. Об’єкт дослідження: процес підтримки оцінювання психологічного стану людини з використанням опитувальників.

4. Предмет дослідження: алгоритми для оцінки та класифікації психологічного стану людини

5. Перелік завдань, які потрібно розробити: аналіз предметної області, аналіз існуючих рішень, формування датасету, побудова MLP нейронної мережі для задачі класифікації, оцінка ефективності навчання мережі, вибір технологі

ного стеку, розробка системи для оцінки психологічного стану людини, перевірка коректності функціонування.

6. Перелік графічного (ілюстративного) матеріалу

Схема бази даних для зберігання інформації про спеціалістів, діаграма бази даних для зберігання опитувальників, діаграма розгортання системи, архітектурна схема системи, функціонування системи моніторингу та оцінки психологічного стану людини, схема нейронної мережі, діаграма варіантів використання.

7. Орієнтовний перелік публікацій

Стаття в науковому журналі категорії Б “Телекомунікаційні та інформаційні технології” №4, назва статті “Оцінка психологічного стану людини з використанням нейронних MLP мереж”.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання “ 31 ” січня 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Опис предметної області і огляд існуючих рішень	14.02	
2	Формування функціональних і нефункціональних вимог до системи	20.02	
3	Визначення математично-технічного базису	27.02	
4	Формування датасету, вибір базових структур	09.03	
5	Процес навчання нейронної мережі, порівняння моделей	27.03	
7	Розробка і тестування системи для комунікації з користувачем	10.04	
8	Оформлення документації	17.04	
9	Подання роботи на попередній захист	20.04	
10	Подання роботи на основний захист	17.05	

Студент _____ *Поліна СЕРГІЄНКО*

Науковий керівник _____ *Євген КРИЛОВ*

РЕФЕРАТ

Магістерська дисертація: 101 сторінку, 39 рисунків, 5 таблиць, 1 додаток, 21 джерело.

Актуальність. Потреба в психологічній допомозі на сьогоднішній день в Україні є доволі гострою, при цьому важливо виконувати оцінки за різними параметрами, проте існуючі системи на разі є вузькопрофільними і не оцінюють стан людини комплексно.

Метою магістерської дисертації є створення системи для оцінки психологічного стану людини, яка дозволила би класифікувати стан людини за шкалою від 1 до 5, стала у нагоді спеціалістам у сфері психології та була би корисною в підтримці оцінки психологічного стану людини. Для досягнення поставленої мети поставлено наступні задачі:

- аналіз предметної області;
- аналіз існуючих рішень;
- формування датасету;
- побудова MLP нейронної мережі для задачі класифікації;
- оцінка ефективності навчання мережі;
- вибір технологічного стеку;
- розробка системи для оцінки психологічного стану людини;
- перевірка коректності функціонування.

Об’єкт дослідження: процес підтримки оцінювання психологічного стану людини з використанням опитувальників.

Предмет дослідження: алгоритми для оцінки та класифікації психологічного стану людини.

Методи дослідження, застосовані у даній роботі, базуються на методах емпіричного дослідження (аналіз існуючих рішень, вивчення наукових матеріалів, що стосуються задачі класифікації, порівняння), методи пошуку технічних рішень (морфологічний аналіз, узагальнений евристичний метод), методи машинного навчання та експертних оцінок.

Наукова новизна отриманих у магістерській дисертації результатів полягає в удосконаленні способу оцінки психологічного стану людини за рахунок розробки системи, яка направлена на роботу з дорослими людьми, є більш широкопрофільною, показує свою ефективність в процесі підтримки оцінки психологічного стану дорослої людини.

Публікація результатів дисертації. За результатами роботи було опубліковано статтю в науковому журналі категорії Б “Телекомунікаційні та інформаційні технології” №4 (, назва статті “Оцінка психологічного стану людини з використанням нейронних MLP мереж”.

НЕЙРОННІ МЕРЕЖІ, MLP, ЗАДАЧА БАГАТОКЛАСОВОЇ КЛАСИФІКАЦІЇ, ОЦІНКА ПСИХОЛОГІЧНОГО СТАНУ ЛЮДИНИ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, МІКРОСЕРВІСНА АРХІТЕКТУРА, БАЗИ ДАНИХ.

ABSTRACT

Master's thesis contains 101 pages, 39 figures, 5 tables, 1 appendices, and 21 sources.

Topicality. The need for psychological help today in Ukraine is quite acute, while it is important to carry out assessments according to various parameters, however, the existing systems are currently narrowly profiled and do not comprehensively evaluate the human condition.

The goal of the master's thesis is to create a system for assessing the psychological state of a person, which would allow classifying a person's state on a scale from 1 to 5, would be useful to specialists in the field of psychology and would be useful in supporting the assessment of a person's psychological state. To achieve the goal, the following tasks were set:

- analysis of the subject area;
- analysis of existing solutions;
- dataset formation;
- construction of an MLP neural network for the classification task;
- evaluation of network learning effectiveness;
- choosing a technological stack;
- development of a system for assessing a person's psychological state;
- checking the correct functioning.

The object of the study: the process of supporting the assessment of a person's psychological state using questionnaires.

The subject of research: algorithms for estimating and classification of a person's psychological state.

The research methods used in this work are based on empirical research methods (analysis of existing solutions, study of scientific materials related to the classification task, comparison), methods of finding technical solutions (morphological analysis, generalized heuristic method), machine learning methods and expert evaluations.

The scientific novelty of the results obtained in the master's thesis lies in the improvement of the method of assessing the psychological state of a person due to the development of a system that is more broad-based and shows its effectiveness in the process of supporting the assessment of the psychological state of an adult.

Publication of thesis results. Based on the results of the work, an article was published in the category B scientific magazine "Telecommunications and Information Technologies" No. 4, the title of the article is " Evaluation of human psychological state using mlp neural networks".

NEURAL NETWORKS, MLP, MULTI-CLASS CLASSIFICATION PROBLEM, ASSESSMENT OF HUMAN PSYCHOLOGICAL STATE, CLIENT-SERVER ARCHITECTURE, MICROSERVICE ARCHITECTURE, DATABASES.

ЗМІСТ

ВСТУП 10
1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ 14
1.1 Важливість психологічної допомоги 14
1.2 Загальний огляд загострення потреби отримання психологічної допомоги 15
1.3 Параметри, які визначають психологічний стан 16
1.4 Огляд існуючих рішень 16
1.4 Постановка задачі 21
Висновки до розділу 23
2 ВИМОГИ ДО СИСТЕМИ. ВИЗНАЧЕННЯ МАТЕМАТИЧНО-ТЕХНІЧНОГО БАЗИСУ 24
2.1 Вимоги до системи 24
2.1.1 Бачення 24
2.1.2 Користувачі. Функціональні вимоги 26
2.1.3 Нефункціональні вимоги до системи 27
2.2 Математично-технічний базис 27
2.2.1 Нейронні мережі 28
2.2.2 Класифікація нейронних мереж 28
2.2.3 Принципи навчання 30
2.2.4 Типи нейронних мереж 35
Висновки до розділу 39
3 ВИЗНАЧЕННЯ БАЗОВИХ СТРУКТУР 40
3.1 Підготовка датасету 40
3.2 Вибір типу нейронної мережі 42
3.3 Вибір моделей MLP 43
3.3.1 Функції активації 44
3.3.2 Функція втрат 47
3.3.3 Оптимізатор 49
3.3.3 Побудова моделей 50
3.4 Інструменти для тренування 53
3.5 Процес навчання 55
3.6 Підвищення ефективності навчання 61

Висновки до розділу 65
4 ПОБУДОВА СИСТЕМИ 67
4.1 Вибір типу архітектури 67
4.1.1 Вибір мессенджера 67
4.1.2 Вибір загальної архітектури 69
4.2 Архітектура бекенд-сервісів 71
4.3 Вибір СКБД 77
4.4 Проєктування БД 79
4.5 Архітектура на рівні сервісу 81
Висновки до розділу 84
5 СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 86
5.1 Вибір технічного стеку 86
5.2 Тестування 90
5.3 Інструкція користувача 93
Висновки до розділу 97
ВИСНОВКИ 98
ПЕРЕЛІК ПОСИЛАНЬ 100

ПЕРЕЛІК ВАЖЛИВИХ СКОРОЧЕНЬ І ТЕРМІНІВ

AI — Artificial intelligence.

API — Application Programming Interface.

DNN — Deep Neural Network.

HTTP — HyperText Transfer Protocol.

MLP — multilayer perceptron.

MVP — minimum viable product.

REST — representational state transfer.

ReLU — rectified linear unit.

SQL — structured query language

БД — база даних

ПЗ — програмне забезпечення

СКБД — система керування базами даних.

ВСТУП

На сьогоднішній день проблема своєчасної психологічної допомоги стоїть дуже гостро через певні зовнішні соціально-економічні та політичні фактори в країні. Протягом останніх років наша країна зіткнулася з низкою викликів, зокрема триваючим військовим конфліктом, політичною нестабільністю, пандемією COVID-19, економічними труднощами. Усі ці події мали значний вплив на психічне здоров'я людей.

Багато хто стикнувся з питання отримання кваліфікованої психологічної допомоги, яку, на жаль, в обставинах, в яких опинилися люди, було неможливо отримати. Тому можливість надання підтримки, порад для покращення стану, а також можливість швидкого знаходження доступного кваліфікованого спеціаліста стоїть є надзвичайно актуальною. Отже, перш за все система, яка зможе в будь-яких умовах швидко оцінити поточний стан пацієнта та надати швидкі поради до того, як з'явиться можливість оффлайн консультації з потрібним спеціалістом, є надзвичайно корисною.

Друга найбільш важлива і актуальна функція аналізу психологічного стану людей – це отримання цінної інформації стосовно поширеності психологічних розладів у людей. Такий застосунок може бути корисним для контролю ефективності поточних служб охорони здоров'я. Як результат отримання подібної інформації, можна сформувати необхідні заходи для покращення психологічного стану людей в країні загалом, запродавши певні нововведення, реформи тощо. Тому можливість надання підтримки, порад для покращення стану, а також можливість швидкого знаходження доступного кваліфікованого спеціаліста стоїть є надзвичайно актуальною.

Як бачимо, аналіз психологічного стану людей, особливо в Україні, є актуальною та важливою темою з огляду на сучасну економічну та соціально-політичну ситуації. Застосунок або бот-месенджер може надавати користувачам зручний і конфіденційний спосіб звернутися за підтримкою без необхідності фізичного відвідування спеціаліста з психічного здоров'я. Така опція буде

особливо корисною для тих регіонів, де подібні послугу з охорони здоров'я є недоступними або доступними не для всіх жителів, особливо в віддалених районах, сільській місцевості або областях, де лишилася невелика кількість спеціалістів.

Ще однією важливою перевагою застосунку може бути ціна. Не всі можуть дозволити собі психологічні послуги. І в такому разі застосунок може бути більш доступним варіантом або ж навіть безкоштовним.

Мобільний застосунок або система, доступ до якої можна отримати, використовуючи будь-який зручний гаджет може надати більше анонімності, оскільки на перших етапах аналізу не буде потребувати комунікації з живою людиною. Така можливість є значною перевагою для людей, для яких характерна сором'язливість або для тих, хто стурбований стигматизацією, пов'язаною зі зверненням за допомогою з питань психічного здоров'я.

Більш того подібна програма може виявити певні проблеми або розлади психологічного стану на ранній стадії, оскільки дозволяє провести оцінку стану в будь-який час. І це може значно полегшити перебіг розладу та запобігти прогресуванню певних симптомів.

Слід зазначити, що для багатьох людей важливе власне керування своїм власним психічним станом, не доручаючи це іншим людям. І в цьому випадку застосунок буде надзвичайно корисним, оскільки зможе рекомендувати певні методики і ресурси, необхідні для керування власним психологічних здоров'ям, що може дати їм можливість брати активну роль у власному благополуччі.

Подібного роду система для оцінки психологічного стану людини може бути корисною не лише для людей, які потребують допомоги чи хочуть впевненитися, що не мають проблем, а для психологів та інших дотичних до цієї сфери спеціалістів. Оцінка психологічного стану може допомогти в розумінні емоційного, когнітивного та поведінкового функціонування людини. Це дає можливість зробити більш обґрунтовані висновки про її потреби, проблеми та можливості. Дані, зібрані системою можуть бути використані для більш точної діагностики, планування терапевтичних втручань і моніторингу прогресу під час лікування.

Таким чином, система оцінки психологічного стану може стати цінним доповненням до цих методів і допомогти забезпечити більш об'єктивну оцінку.

Оскільки програма може бути корисною у вирішенні багатьох проблем, комплексів людей, стати своєчасною підтримкою у разі потреби, допомогти контролювати ефективність соціальних служб на місцевому чи навіть державному рівні, вона є надзвичайно актуальною в наш час та вартує детального вивчення і постійного розвитку і розширення.

Метою дослідження є створення системи для оцінки психологічного стану людини, яка би стала у нагоді спеціалістам сфери психологічної допомоги і дозволила би визначаити та класифікувати стан людини. Для досягнення мети необхідно виконати наступні задачі:

- виконати детальний аналіз предметної області;
- виконати огляд існуючих досліджень, який стосуються тематики аналізу психологічного стану людей;
- описати основний функціонал, враховуючи всі потреби користувачів та розподіл прав на користування застосунком згідно з роллю;
- дослідити методи аналізу психологічного стану людини;
- підготувати дані, які можна використовувати у якості тестових;
- обрати необхідний технічний стек, описати базові алгоритми для вирішення поставленої задачі, описати модель для аналізу стану людини;
- описати загальну архітектуру системи, враховуючи можливість комунікації з користувачем;
- описати архітектурні підходи до створення, визначивши основні інструменти та технології;
- виконати тестування системи для переконання у відповідності системи початковим вимогам, перевірити швидкість опрацювання запитів та витримування навантаження.

Практичне значення отриманих результатів: система може використовуватися людьми, які мають проблеми, пов’язані з психологічним станом або слідкують за своїм психологічним здоров’ям. Також система може

використовуватися психологами та дотичними до цієї сфери спеціалістами з метою підтримки стосовно оцінки психологічного стану людини. В подальшому, система може бути застосована на державному рівні для збору даних, формування певних звітів, подальшого аналізу і оцінки загального стану населення і впливу зовнішніх факторів на психологічний стан людей.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Перед початком роботи важливо розібратися в нюансах процесу оцінки психологічного стану людини, визначити, чому це важливо, що впливає на психологічний стан людини. Оскільки проблематика доволі актуальна і тема є важливою на сьогоднішній день, ця сфера розвивається і уже існують певні рішення, проте всі вони відрізняються своє. цільовою задачею, аудиторію, на яку направлені, реалізацією. В роботі було досліджено ряд існуючих систем та виконано аналіз отриманих результатів для формування вимог до розроблюваної системи.

1.1 Важливість психологічної допомоги

Психологічна допомога дуже важлива для людей, оскільки вона може покращити психічне здоров’я, підвищити добробут і загальну якість життя. Перш за все допомагає людям навчитися справлятися з труднощами та певними життєвими викликами, такими як труднощі у стосунках, фінансова напруга, стрес на роботі. Через подібні труднощі (зовнішні фактори) у людей можуть виникати неприємні прояви розладів: панічні атаки, ступор, нестерпний страх, плач, неконтрольована агресія, нервове тремтіння, психологічне виснаження тощо. У таких випадках аби не дати можливості психологічному виснаженню прогресувати, перш за все, варто зафіксувати наявність проблеми і отримати кваліфіковану психологічну допомогу або на перших етапах – інструкцію, що робити аби полегшити собі життя.

В подальшому, психологічна допомога може принести помітний позитивний результат. Людина може більш чітко реагувати у стресових ситуаціях, ефективніше впоратися з різними непередбачуваними ситуаціями. Загалом людина після роботи зі спеціалістами помічає покращення загального стану здоров’я і не відчуває ознак депресії, пост-травматичних симптомів або постійного переживання. Окрім цього, як правило, спостерігається поліпшення стосунків з рідними, з друзями і з

найближчим оточенням, оскільки людина краще розуміє свої потреби, відчуття, емоції. І також до переваг можна віднести зменшення ризиків ментальних проблем в майбутньому, оскільки людина чітко розуміє свій «звичайний» стан і стан, коли відбуваються певні зміни.

1.2 Загальний огляд загострення потреби отримання психологічної допомоги

Останнім часом в Україні і в світі загострилася проблема отримання психологічної допомоги. На це вплинула велика кількість факторів.

Одним з таких факторів стала пандемія COVID-19, яка призвела до значного збільшення потреби у психологічній допомозі, оскільки спалах хвороби спричинив низку стресових факторів: багато тих, хто втратив постійне місце роботи, опинився в ситуації фінансової нестабільності. Соціальна ізоляція ще більше вплинула на психологічне здоров'я людини, і на той час доволі важко було визначити, що людина перебуває в депресивному стані і отримати відповідну допомогу.

Страх заразитися вірусом, занепокоєння про близьких і невпевненість у майбутньому – все це сприяло підвищенню рівня стресу. Заходи соціального дистанціювання призвели до того, що багато людей почуваються ізольованими та самотніми. За даними психологічних досліджень пандемія призвела до зростання домашнього насильства, оскільки багато людей змушені були залишатися поруч зі своїми кривдниками.

Для людей в Україні значним фактором, який загострив потребу, отримання психологічної допомоги, є триваючий воєнно-політичний конфлікт. Кожен день люди відчувають страх за себе і близьких, ті, хто виїхав за кордон відчувають сум за домівкою і звичним способом життям. Більшість стикнулися з проблемами фінансової нестабільності. Через певні руйнування комунікацій частина українців мали обмежений доступ до психологічних послуг, а ще частина просто не може собі дозволити оплатити послуги кваліфікованих спеціалістів.

Саме тому онлайн-сервіси для моніторингу психологічного стану є вкрай важливими, оскільки дозволяють підтримувати стан громадян в тонусі. Головна задача розроблюваної системи – вчасно виявити проблему і допомогти у її рішенні: запропонувати порядок дій, які може послабити певну симптоматику та порадити спеціаліста, який задовольнить всім критеріям клієнта.

1.3 Параметри, які визначають психологічний стан

Психологічний стан людини – це певна характеристика особи, яка показує рівень її переживань або інших факторів, які впливають на сприйняття зовнішнього світу. На психологічний стан людини можуть впливати страх, тривожність, депресивність, самооцінка, емоційне вигорання тощо.

Більшість з цих проблем визначаються під час діалогу зі спеціалістом у процесі запитань і відповідей, де людина може самооцінити свій поточний стан. Людина може оцінити свій власний настрій та емоційний стан, відповідаючи на питання про свої емоції, настрій, рівень тривожності та інші параметри. Це може бути зроблено за допомогою анкет, опитувань або інших психологічних інструментів, що дозволяють людині самостійно визначати свій настрій та емоційний стан. А потім спеціаліст аналізує результати анкетування.

Подібний формат «розмови» можна перенести в формат відповідей на питання, які ставить система. Після чого система може сама спробувати оцінити рівень психологічної стабільності.

1.4 Огляд існуючих рішень

На сьогоднішній день уже існують певні програми, які можуть допомагати людині в тій чи іншій мірі підтримувати чи покращувати свій психологічний стан. Застосунки, які будуть описані нижче, є найбільш використовуваними станом на сьогодні.

1.4.1 MindShift

Даний застосунок було розроблено канадськими спеціалістами. Основна його спеціалізація – це вирішення проблем, пов’язаних з підвищеною тривожністю. Основною соціальною групою серед користувачів програми є молодь і підлітки.

Інтерфейси програми наведено на рисунку 1.1. Серед функціоналу можна визначити:

- програми для зниження рівня стресу: медитації, дихальні вправи, вправи для боротьби зі страхами;
- моніторинг сну;
- щоденник, що контролює процес подолання тривожності.

Застосунок показує свою ефективність. Цей процес контролюють і перевіряють науковці.

Дана програма є безкоштовною, оскільки була запущена у тестовій версії. Серед недоліків можна виділити те, що програма доступна лише у англомовній версії, що унеможливлює використання людьми, які нею не володіють. Також бачимо, шо програма спеціалізується на вузькому спектрі проблем та є корисною лише після усвідомлення людини, що вона має проблему з тривожністю. На етапі спроби зрозуміти, чи все так у житті людини, застосунок буде безкорисним.

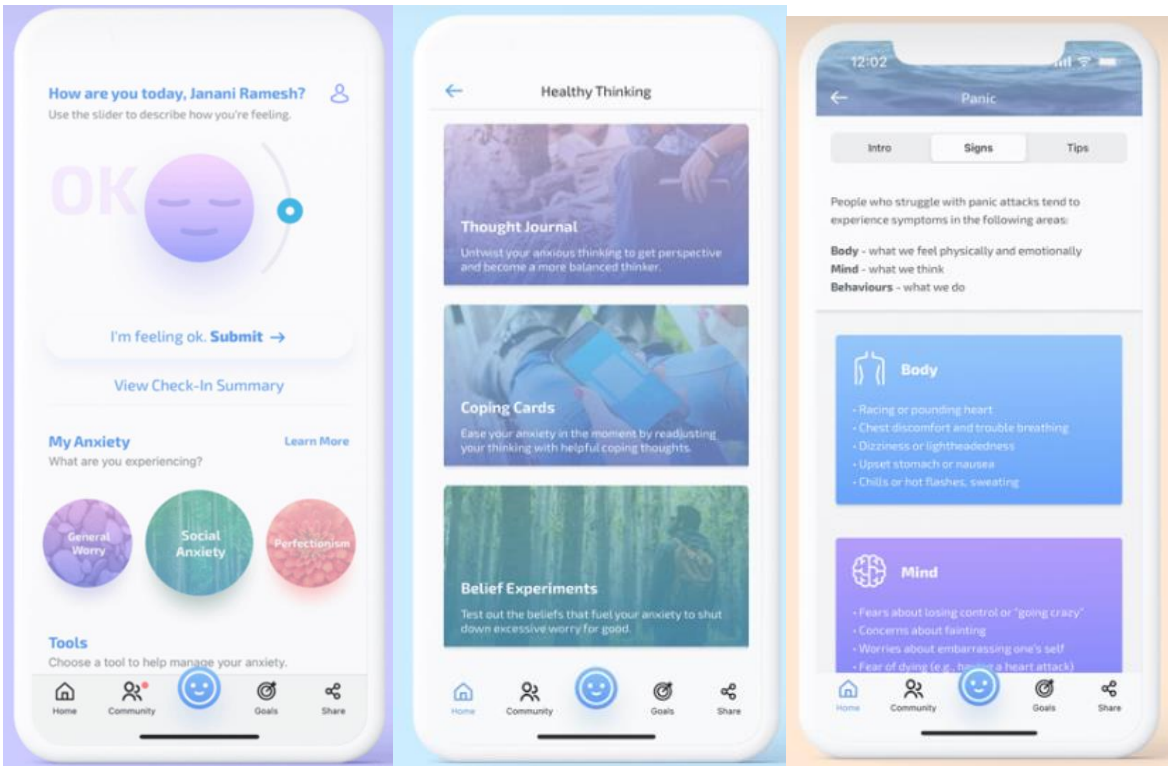

Рисунок 1.1 – Інтерфейс програми Mindshift

1.4.2 Woebot

Дана програма працює у форматі чат-бота, тобто спілкується з користувачем через месенджер. Основна задача застосунку – цікавитися, як у користувача справи та як самопочуття. Бот звертає увагу на формулювання, якими оперує людина і намагається показати, що будь-які почуття є нормою. Найбільш ефективною програма є при роботі з депресивними станами. Інтерфейси програми наведено на рис. 1.2.

Серед мінусів можна виділити комунікацію виключно англійською мовою. Аналогічно дана програма не передбачає можливості роботи в кооперації з психологом. Але варто звернути увагу, що повноцінно подібного роду застосунок не може замінити повністю терапію з живим спеціалістом. Самостійної роботи з проблемою не завжди буває достатньо. Хоча корисні статті можуть наштовхувати на думку про звернення до фахівців.

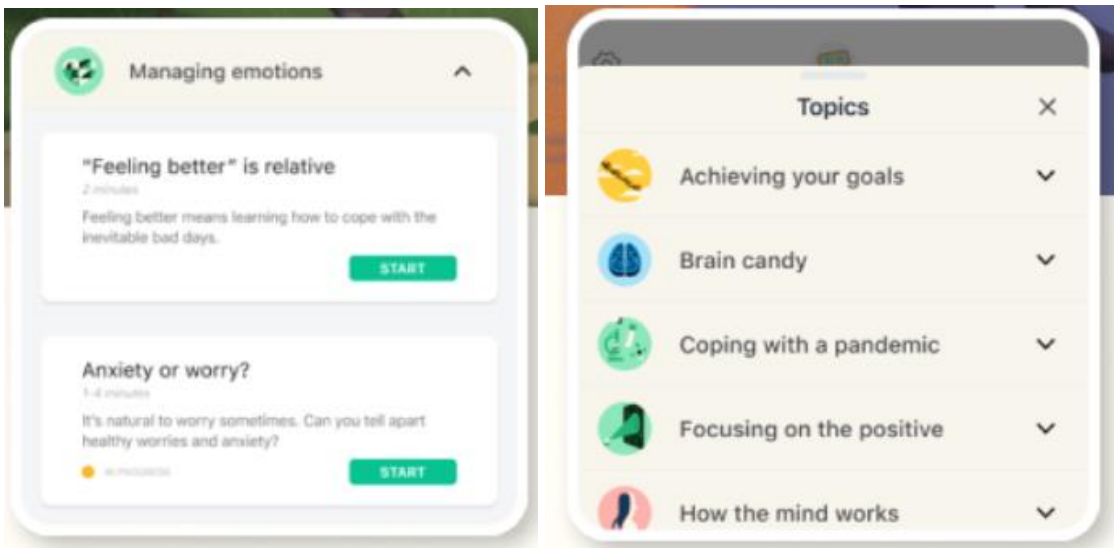

Рисунок 1.2 – Інтерфейс програми Woebot

1.4.3 T2 Mood Tracker

Дана програма призначення для того, щоб кожного дня моніторити настрій людини й контролювати його зміни. Застосунок пропонує нотувати, тобто записувати події, які трапилися з користувачем протягом дня, відмічати реакцію на кожну з подій, і в результаті програма формує графіки залежності настрою від певних тригерів. Найбільш чітко додаток відслідковує появу тривожного стану або депресії. Інтерфейс застосунку наведено на рис. 1.3. Окрім цього, система нотифікує психолога про важливі зміни в житті користувача. І таким чином, дозволяє працювати над дійсно актуальними питаннями.

Проте дана програма не дає поради, як швидко можна допомогти собі самостійно в критичному стані, доки не відбудеться консультація з психологом. Окрім того, іноді важко зробити перший крок і звернутися по допомогу через нюанси і переконання людей, які уже згадувалися вище.

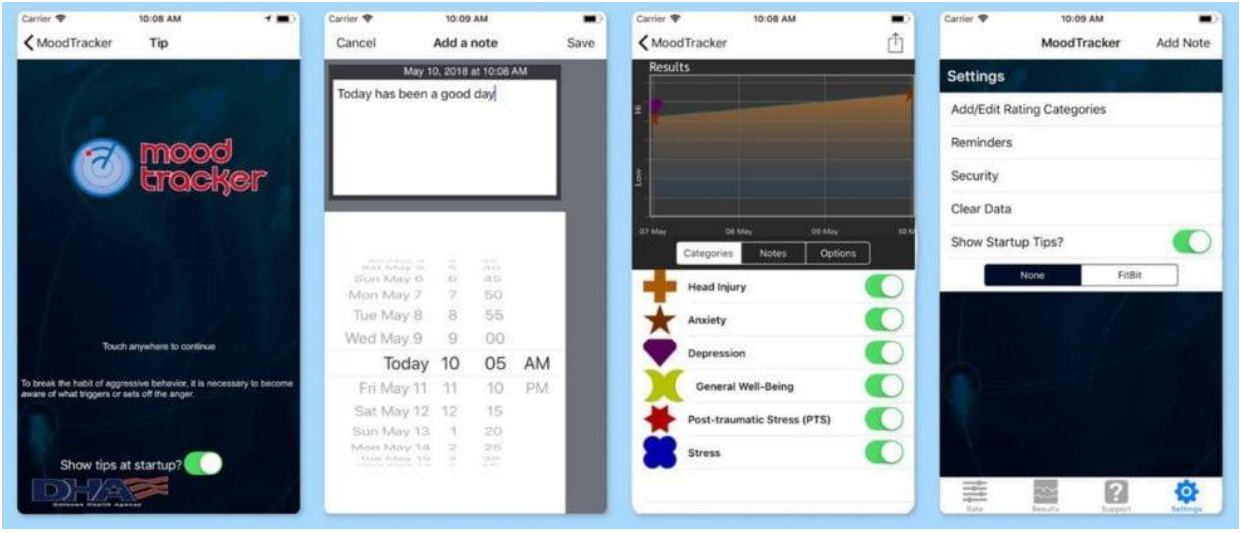

Рисунок 1.3 – Інтерфейс програми T2 Mood Tracker

1.4.4 Daylio

Дана програма являє собою щоденник емоцій. Вона має свою шкалу настроїв (емоцій) і багато видів різної активності. Користувач просто обирає, чим саме він займається та які емоції отримує в процесі. Потім застосунок аналізує звички, підказує, які справи є неприємними для користувача і пропонує їх зробити якомога раніше зранку. Основна задача цього застосунку – показати , які саме активності приносять задоволення. Зовнішній вигляд інтерфейсів застосунку наведено на рисунку 1.4. Дана програма має українську версію.

Перевагою є можливість занотувати про свої почуття і зрозуміти, що саме було би добре скоригувати у звичному розпорядку дня. Проте доволі важко зрозуміти, як боротися яскраво вираженими неприємними відчуттями і психологічними розладами. Звісно, розуміння того, що приносить задоволення, а що навпаки, знижує ризик виникнення депресивних станів, але у разі виникнення складних симптомів важко зрозуміти, як вийти з такого стану.

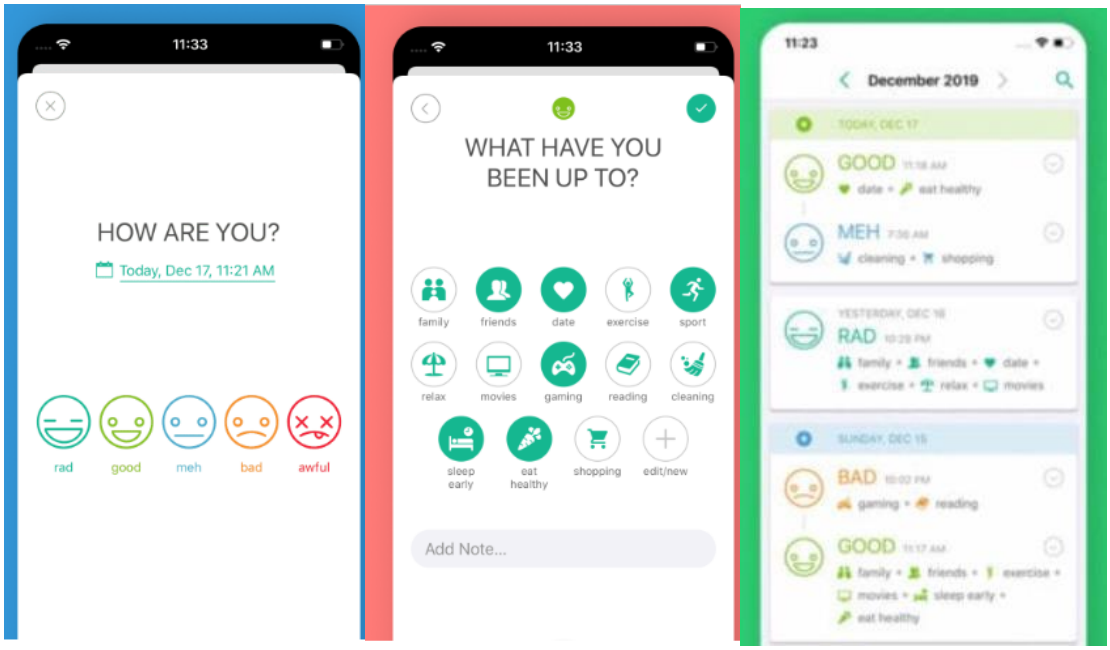

Рисунок 1.4 – Інтерфейс програми Daylio

1.4 Постановка задачі

Метою проекту є проаналізувати можливі процеси і алгоритми, необхідні для оцінки психологічного стану людини та розробити програму, яка зможе використовуватися на комерційній основі для психологічної допомоги людям, а також державними організаціями для відслідковування і покращення психологічного стану людей, а також для оцінки ефективності роботи соціальних служб, які займаються нормалізацією рівня життя і здоров'я громадян.

За результатами проведеного аналізу було визначено, які саме вимоги є критичними для додатку для покращення функціональності, яку надають уже існуючі системи:

- оцінка стану людини має виконуватися комплексно, включаючи різні показники;
- результатом роботи системи має бути визначена оцінка психологічного стану людини, що зазначається шкалою від 1 до 5, де 1 – людина терміново потребує допомоги, відповідно 5 – людина перебуває в стані спокою;
- інтуїтивний інтерфейс;

- підтримка української мови, локалізація;
- чіткий незалежний результат щодо поточного стану користувача;
- рекомендації щодо можливості впоратися самостійно з проблемою;
- пропозиції щодо своєчасної допомоги спеціалістів платформи.

Нефункціональні вимоги до системи:

- точність системи має сягати не менше 93%;
- обробка результатів опитування користувачів має відбуватися коректно;
- відповідь системи щодо результатів має бути швидкою, не більше 2 секунд;
- логіка системи в майбутньому має бути легко розширюваною з найменшими витратами;
- виключні ситуації мають бути коректно оброблені.

Для забезпечення надійності системи варто виконати наступні вимоги:

- цілісність та несуперечливість даних;
- безпека доступу до даних;
- достовірність даних;
- простота керування даними.

Розроблювана має вирізнятися від уже існуючих систем своєю унікальністю, якісно давати незалежну оцінку стосовно рівня психологічного стану людини, пропонувати спеціаліста, який здатний допомогти у ситуації, що склалася, а також бути легко розширюваним для нового функціонала, а також для використання нових нейронних мереж.

Отже, для досягнення поставленої мети необхідно:

- обрати математично-технічний базис, технічний стек та побудувати схеми системи саме для оцінки психологічного стану людини;
- перевірити ефективність роботи цієї системи за показниками відповідності реальним результатам та швидкодії;
- у разі отримання недостатньо ефективного результату, подумати стосовно оптимізації і покращення;

- описати архітектурний підхід до створення системи, яка буде безпосередньо взаємодіяти з користувачем та обрати технічний стек;
- виконати опис структури БД;
- реалізувати API для обробки запитів клієнта;
- провести тестування системи на віповідність початковим функціональним вимогам та перевірити системи щодо витримання навантаження та швидкості обробки запитів;
- у разі потреби виконати оптимізацію системи або окремих її частин.

Висновки до розділу

В рамках цього розділу було вивчено предметну область, проблеми, пов’язані з неможливістю отриманням психологічної допомоги. Було визначено найголовнішу мету систему – своєчасна коректна оцінка психологічного стану людини.

Виконано порівняльний аналіз уже існуючих рішень, визначено особливості, які можна покращати, а також функціонал, який є відсутнім на сьогоднішній день. Визначено основні вимоги до розроблювальної системи. Також було зроблено перелік задач, які необхідно виконати для побудови системи.

2 ВИМОГИ ДО СИСТЕМИ. ВИЗНАЧЕННЯ МАТЕМАТИЧНО-ТЕХНІЧНОГО БАЗИСУ

Наступним етапом після визначення проблематики в рамках існуючих систем є створення вимог до розроблюваної системи, яка має покращити або внести щось нове. Перш за все має бути сформавано загальне бачення системи, визначено основних користувачів системи, сформульовано функціональні і нефункціональні вимоги.

Після визначення чітких вимог важливим етапом є визначення математично-технічного базису, підбір технологій, які можуть бути корисними і ефективними для розробки системи, яка може допомагати в оцінці психологічного стану людини.

2.1 Вимоги до системи

Етап визначення вимог перед розробкою програмного забезпечення є критично важливим для успішного процесу розробки програмного продукту. Основною метою визначення меж системи для уникнення нескінченного розширення проєкту, побудувати більш чіткий план роботи над системою.

2.1.1 Бачення

Одним з основних етапів роботи над розробкою системи є формування бачення проєкту. Воно відображає бажане становище, яке потрібно досягнути після завершення проєкту. Бачення допомагає визначити, яким чином проект буде задовольняти потреби користувачів і які основні характеристики мають бути реалізовані.

Оскільки на сьогоднішній день питання своєчасного визначення стану психологічного самопочуття людини стоїть надзвичайно гостро і є важливим питанням, як і для самої людини, так і для підприємств, де може працювати людина або для держави загалом. Відповідно з’являється потреба у загальному додатку,

який зможе якісно оцінити, наскільки людина в нормі, а також надати поради, у разі потреба та порекомендувати фахівця, якщо проблема доволі глибока.

У ході аналізу вже існуючих рішень були визначені наступні проблеми:

- відсутність безкоштовної версії або відсутність підтримки українською мовою;
- моніторинг зміни стану людини в залежності від активностей, якими займається людина протягом дня;
- направлені на більш молоду категорію населення;
- неможливість оцінки стану людини за широким спектром показників;
- допомагають справитися лише з однією відомою проблему і не працюють на етапі визначення причини поганого самопочуття.

Ці проблеми унеможливлюють користування додатком від етапу тестування стану до закінчення терапії, особливо враховуючи, що людей можуть турбувати все нові і нові під питання під час роботи на одним уже конкретним.

Наслідками цих проблем є:

- необхідність встановлення декількох застосунків;
- багато часу на заповнення даних в різних застосунках;
- незручність використання іноземною мовою;
- самостійних пошук спеціалістів і опис проблеми.

Успішним вирішенням усіх вище описаних проблем буде

- створення нової система, яка зможе оцінити стан людини;
- швидкий процес оцінки стану людини з похибкою менше 10%;
- правильне проектування алгоритмів, баз даних та бекенд-сервісів;
- обґрунтований вибір методів міжсервісної комунікації;
- створення інтуїтивно зрозумілого користувацького інтерфейсу.

2.1.2 Користувачі. Функціональні вимоги

Програма має 4 основні ролі користувачів: пацієнт, лікар (психолог, психоаналітик, психотерапевт тощо), адміністратор. Відповідно до своєї ролі програма надає наступні можливості:

Пацієнт, психолог, адміністратор:

- привітання пацієнтів, які звертаються за допомогою вперше за день;
- бажання гарного дня після завершення роботи з програмою;
- нагадування про заплановану консультацію.

Пацієнт:

- проходження опитувальника для оцінки психологічного стану;
- отримання рекомендацій за результатами опитувальника;
- знайти у разі необхідності спеціаліста, який займається проблемами у визначеній області;
- щоденне швидке опитування щодо прогресу лікування та зміни морального стану;
- нотифікації про необхідність пройти опитування.

Психолог:

- отримання запитів на терапію від пацієнтів;
- перегляд заповнених анкет та згенерованих результатів, можливість їх редагування;
- моніторинг динаміки стану клієнта на основі щоденних опитувань.

Адміністратор:

- додавання в систему нових лікарів, редагування їх профілів;
- контроль роботи лікарів: отримання звітів про зустрічі;
- моніторинг стану пацієнтів під час проходження терапії.

Додаткові вимоги:

- система має працювати на базі мессенджера (Telegram бот), мати чіткі і зрозумілі команду українською мовою;

- всі клієнти мають бути автоматично зареєстровані в системі при першому зверненні;
- незареєстровані користувачі можуть лише ознайомитися з інформацією про систему, її призначення і функціонал. Діаграма прецедентів наведена в додатках (додаток А).

2.1.3 Нефункціональні вимоги до системи

Для забезпечення надійності системи, в якості атрибуту якості системи визначено наступні вимоги.

Точність оцінки психологічного стану людини має становити не менше 90%.

В тестовому режимі очікується, що система матиме 25 психологів в БД за різними профілями. Одночасно система має підтримувати 2-3 людини. Очікувана кількість користувачів системи в рамках тестування – 1000. В подальшому кількість користувачів може бути збільшена, отже необхідно врахувати потребу в легкому масштабуванні.

Всі запити в систему мають оброблятися за час не більший 2с. З підвищенням навантаження на систему не критичне уповільнення часу обробки запитів до 3с.

Система має бути доступною у порівнянні з “downtime” у 98% випадків. У разі виникнення технічної помилки, система має бути стабілізована за час, не більший 3 годин, оскільки потреба в допомозі може стати критичною.

Дані користувачів мають зберігатися протягом 2 років за замовчуванням. Інший період може бути налаштований користувачем.

Всі дані тестування користувачів в БД мають бути зашифровані.

Система має бути легко розширюваною новим функціоналом в майбутньому.

2.2 Математично-технічний базис

Найважливішою частиною системи – є підсистема, яка має проаналізувати стан людини на основі проведеного опитування. Зазвичай у реальному житті

подібного роду аналіз проводиться психологами мануально, враховуючи відповідь на кожне з питань. На виконання такого процесу необхідно витратити велику кількість часу. Даний процес може бути автоматизовано, використовуючи загальний сучасний підхід для автоматичного аналізу подібної інформації – нейронні мережі.

2.2.1 Нейронні мережі

Нейронні мережі є потужними інструментами для аналізу та обробки даних, і вони можуть бути використані для розпізнавання паттернів у поведінці людини або для прогнозування її психологічного стану на основі зібраних даних.

Нейронна мережа - це складна система обробки інформації, яка моделює роботу нервової системи тварин та людей. Вона складається з великої кількості взаємопов'язаних "нейронів", кожен з яких виконує певну обчислювальну функцію. Нейрони з'єднані між собою за допомогою зв'язків, які передають інформацію від одного нейрона до іншого [1].

У нейронній мережі, навченій на певному наборі даних, нейрони можуть виконувати різні обчислювальні операції, такі як розпізнавання зображень, класифікація тексту, аналіз мови, рекомендації та багато іншого. Для навчання нейронних мереж застосовуються алгоритми машинного навчання, які змінюють ваги зв'язків між нейронами, щоб максимізувати точність передбачень мережі на основі вхідних даних.

2.2.2 Класифікація нейронних мереж

Існує безліч різних типів нейронних мереж, кожен з яких має свої особливості та застосування в певних областях.

В залежності від типу архітектури, можна виділити 2 великі типи нейронних мереж: прямі і рекурентні. Прямі мережі характеризуються тим, що сигнали передаються тільки в одному напрямку, від вхідних до вихідних нейронів. В

рекурентних нейронних мережах сигнали можуть передаватися в обидва напрямки, що дозволяє використовувати історію попередніх сигналів при обробці нових даних [2]. Схема класифікації нейронних мереж за типом архітектури наведено на рис.2.1.

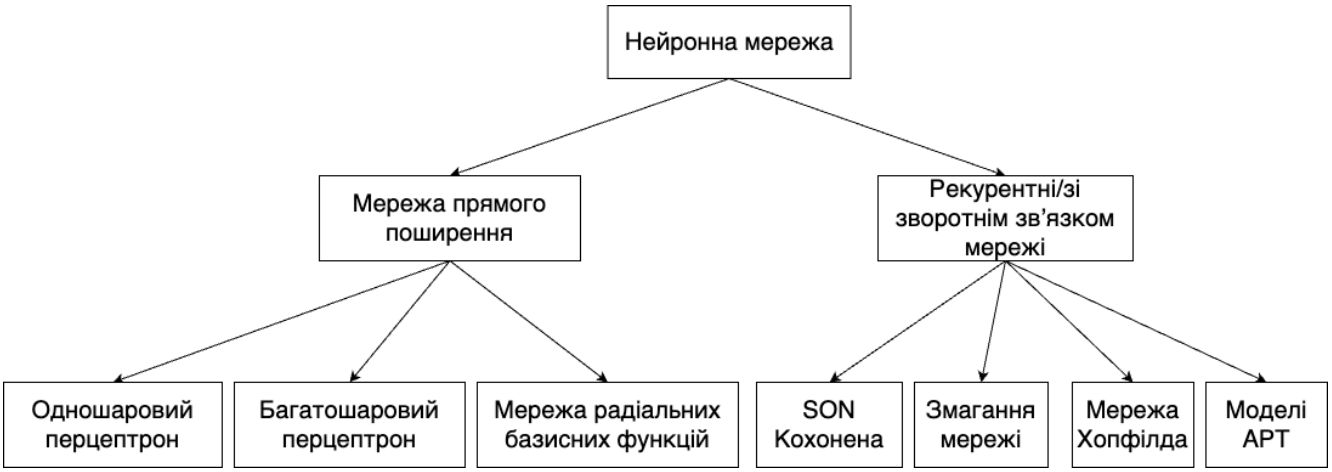

Рисунок 2.1 - Класифікація нейронних мереж за типом архітектури

За структурою виділяють 2 типи нейронних мереж: повнозв'язні і неповнозв'язні (рис. 2.2). В повнозв'язній структурі кожен нейрон у попередньому шарі пов'язаний з кожним нейроном у наступному шарі. Всі вихідні значення з попереднього шару передаються на вхід кожного нейрону в наступному шарі. Неповнозв'язні нейронні мережі характеризуються тим, що лише певна підмножина нейронів у попередньому шарі передає свої значення на вхід кожного нейрону в наступному шарі [3]. Такі мережі можуть мати різні типи з'єднань між шарами, такі як розріджені з'єднання, з'єднання з певною ймовірністю або з'єднання на основі певних правил.

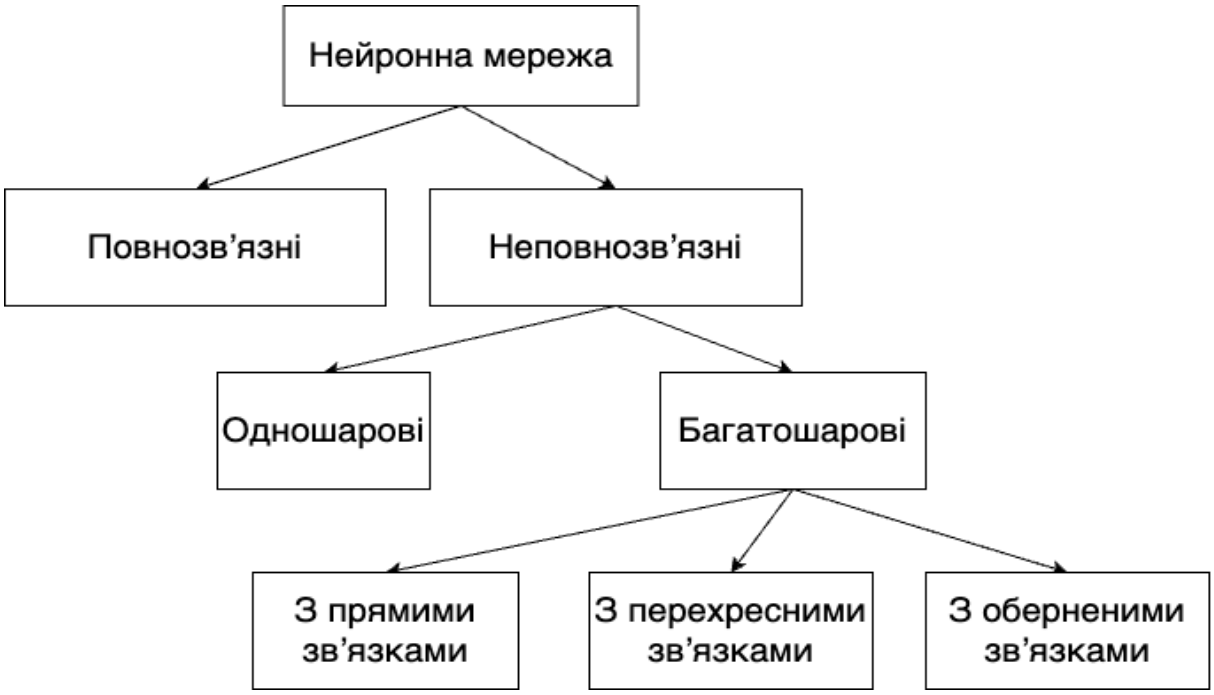

Рисунок 2.2 - Класифікація нейронних мереж за структурою

Також нейронні мережі відрізняються за своєю топологією. Можна виділити 3 основні типи: повнозв'язні, багатошарові і слабозв'язні. Якщо існує велика кількість нейронів, то зазвичай вони об'єднуються в шари. Кожен з цих шарів має свій власний набір нейронів, при чому будь-яку кількість, вона не залежить від інших шарів. У слабозв'язаних нейронних мережах зазвичай нейрони розташовуються у вузлах гексагональної або прямокутної решітки.

2.2.3 Принципи навчання

Навчання нейронної мережі - це процес налаштування ваг та параметрів мережі з використанням набору вхідних даних, щоб модель могла вчиняти передбачення або виконувати завдання, необхідні для вирішення конкретної проблеми.

Нейронні мережі навчаються за допомогою тренувального набору даних, який містить в собі вхідні дані та відповідні мітки чи значення. Сам процес навчання виконується за алгоритмом наведеним нижче.

Етап 1. Ініціалізація. За певними правилами ініціюються параметри мережі.

Етап 2. Forward pass. Вхідні дані проходять через мережу, виконуючи послідовні операції, такі як лінійні комбінації та активаційні функції, щоб отримати передбачення моделі.

Етап 3. Вибір лінійних комбінації і активаційних функцій – основних елементів нейронної мережі.

Лінійною комбінацією можна назвати вираз, який являє собою суму елементів, перемножених на певні коефіцієнти. Математично це може бути виражено наступним чином: z = w1*x1 + w2*x2 + … + wn*xn + b.

Функція активації визначає вихідне значення нейрона в залежності від результату виваженої суми входів та порогового значення. Функція активації застосовується з метою введення нелінійності в рішення моделі. Функція активації приймає вхідний сигнал і видає вихідний сигнал, який передається на вхід наступного шару мережі. Функції активації можуть бути різних типів, таких як сигмоїдна, гіперболічний тангенс (tanh), ReLU (Rectified Linear Unit), Leaky ReLU, softmax та багато інших [4]. Функції активації дозволяють нейронним мережам моделювати складні не-лінійні залежності між вхідними даними, що робить їх потужними інструментами для рішення різноманітних завдань машинного навчання.

Етап 4. Визначення loss consumption. На цьому етапі порівнюються передбачення моделі з вихідними мітками та обчислюється значення функції втрати, яка вказує на рівень помилок моделі. Якщо похибка мережі є невисокою, то можна вважати, що мережа навчена. В іншому випадку переходять до наступного етапу навчання.

Етап 5. Оновлення (коригування) ваг мережі. Необхідно оновити параметри мережі щоб зменшити значення функції втрат та покращити точність моделі.

Цей процес повторюється декілька разів, на багатьох епохах. Епоха – це одна повна ітерація навчального процесу, коли весь навчальний набір даних проходить через мережу один раз. Зазвичай, більше епох веде до кращої збіжності моделі, але також може збільшувати час навчання [5]. Вибір оптимальної кількості епох - це

компроміс між ресурсами обчислення та швидкістю навчання, залежно від конкретної задачі та даних.

Розрізняють навчання з вчителем і без.

2.2.3.1 Типи навчання

Процес навчання з вчителем полягає у тому, що алгоритм навчається на основі набору прикладів даних, в яких вже є відомі правильні відповіді. Зразок відповідей подається на входи мережі, після чого обробляється в структурі мережі і обчислюється вихід мережі, що потім порівнюється з відповідним значенням цільового вектора, який є очікуваним (бажаним) виходом мережі. Після чого за певним правилом розраховується похибка і коригуються вагові коефіцієнти зв'язків всередині мережі. Вектори навчального набору подаються послідовно, похибки обчислюються, відбувається коригування вагів, допоки похибка в навчальній матриці не досягне допустимого рівня.

Загалом, навчання з вчителем може широко використовуватися для задач класифікації, коли є потреба розділити об’єкт за певним сетом відомих ознак. Також такий алгоритм може бути корисним для задач регресії. Регресія дуже схожа на класифікацію, проте замість категорії передбачається певне число.

Процес навчання без вчителя полягає в тому, що модель навчається на основі набору даних, в якому немає відомих правильних відповідей або міток. Такий підхід часто використовується, як метод аналізу даних.

Навчання без учителя може бути корисним у задачах кластерізації, коли необхідно розділити об’єкти за невідомою ознакою, коли є потреба в узагальненні, для пошуку певних правил чи асоціацій, наприклад, для пошуку певних закономірностей в замовленнях.

Ще одним типом навчання є навчання з підкріпленням. Основна ідея такого підходу полягає у тому, що програма вчиться досягати своєї мети в середовищі, яке є невизначеним і подібний сценарій є схожим на гру.

Для знаходження правильного рішення система використовує метод проб і помилок. За правильні рішення вона отримує «заохочення» - певні бали, а за неправильні – певне покарання, наприклад штрафні бали. Основна мета агента отримати максимальну кількість балів. Агентом в такому сценарії називають сутність, яка приймає рішення вибору дій в середовищі на основі введених станів та отриманих винагород або покарань [6].

Навчання з підкріпленням може застосовуватися у багатьох різних сферах, таких як автоматизація виробництва, робототехніка, фінанси, реклама, ігри та багато інших. Наприклад, в робототехніці навчання з підкріпленням може використовуватися для навчання роботів виконувати різноманітні завдання, такі як переміщення, навігація, рухи тощо. В автоматизації виробництва воно може застосовуватися для оптимізації процесів виробництва та управління ресурсами. У фінансовому секторі навчання з підкріпленням може використовуватися для прийняття рішень щодо управління фінансовими портфелями, купівлі-продажу цінних паперів та оптимізації ризиків. У рекламі воно може використовуватися для оптимізації рекламних кампаній та вибору оптимальних стратегій рекламного взаємодії. У галузі ігор навчання з підкріпленням застосовується для розробки ігрових агентів, які можуть самостійно вчитися та вдосконалювати свої навички гри.

2.2.3.2 Правила навчання

Існують 4 типи правил навчання нейронних мереж:

- корекція за помилкою;
- навчання Больцмана;
- правило Дельта;
- навчання методом змагання.

Правила навчання Больцмана використовуються для навчання ваг (параметрів) моделі Больцмана шляхом апроксимації ймовірностей входу і виходу моделі. Основна ідея полягає в тому, що ваги нейронів навчаються шляхом

максимізації правдоподібності вхідних даних, враховуючи взаємодію між нейронами.

Дельта-правило є простим правилом навчання, використовуваним у багатьох моделях штучних нейронних мереж. Це правило використовується для навчання ваг нейронів на основі різниці між прогнозованим виходом нейрона і очікуваним виходом. У процесі навчання ваги оновлюються згідно з градієнтним спуском, змінюючи їх в напрямку, що максимізує правильність прогнозів.

Корекція за помилкою використовується для внесення необхідних змін у ваги нейронів, щоб зменшити помилку між прогнозованим виходом мережі та очікуваним виходом. Процес корекції заснований на градієнтному спуску, де ваги нейронів оновлюються в напрямку, протилежному до градієнта (похідної) функції втрати [7]. Зазвичай використовується алгоритм зворотного поширення помилок, який дозволяє передавати помилку з виходу до входу мережі та оновлювати ваги.

Процес корекції за помилкою може бути розглянутий у кількох кроках.

Крок 1. Прямий прохід: вхідні дані подаються на вхід мережі, і виходи розраховуються на кожному шарі до досягнення вихідного шару.

Крок 2. Розрахунок помилки: порівнюються прогнозовані виходи мережі з очікуваними значеннями і обчислюється величина помилки.

Крок 3. Зворотнє поширення помилки: помилка поширюється назад від вихідного шару до вхідного шару, розраховуючи помилки на кожному шарі і враховуючи ваги зв'язків між нейронами.

Крок 4. Оновлення ваг: ваги нейронів оновлюються згідно з градієнтним спуском, змінюючи їх значення в напрямку, який зменшує помилку.

Навчання методом змагання (англ. competitive learning) є одним з методів навчання без вчителя, використовуваним у штучних нейронних мережах. Цей підхід базується на конкуренції між нейронами у мережі за право відповідати на певний стимул.

У методі змагання використовуються безліч нейронів, організованих у вхідний шар та приховані шари. Кожен нейрон зв'язаний зі всіма нейронами

попереднього шару, але тільки один нейрон в прихованому шарі може активуватися та відповідати на конкретний стимул.

Процес навчання методом змагання зазвичай включає етапи описані нижче.

Етап 1. Ініціалізація ваг. На цьому етапі ваги нейронів встановлюються випадковим чином або за певними правилами.

Етап 2. Вибір вхідного стимулу: подається один зразок даних або стимул на вхід мережі.

Етап 3. Розрахунок активації: обчислюється активація кожного нейрона на основі вхідних даних та ваг.

Етап 4. Визначення переможця. Нейрон з найбільшою активацією вважається переможцем і активується.

Етап 5. Оновлення ваг: ваги переможцевого нейрона та його сусідів оновлюються, щоб збільшити чутливість до подібних стимулів у майбутньому.

Етап 6. Повторення процесу. Етапи 2-5 повторюються для кожного зразка даних у навчальному наборі.

Навчання методом змагання може застосовуватися для кластеризації даних, виявлення шаблонів, розпізнавання візуальних ознак та інших завдань без вчителя.

2.2.4 Типи нейронних мереж

Узагальнено, процес оцінки психологічного стану людини за шкалою від 1 до 5 є задачею багатокласової класифікації, тобто результатом роботи мережі має бути номер (оцінка) до якого належить людина з таким станом. На практиці задача класифікації найкраще вирішується при використанні навчання з учителем, тобто при формуванні тестового набору вхідних даних з визначеним спеціалістами вихідним результатом.

Для задач багатокласової класифікації можуть використовуватися різні типи нейронних мереж.

Перцептрон з багатьма шарами (MLP). В загальному, такі нейронної мережі складаються з декількох зв’язаних послідовно нейронів. У цій структурі є 3 шари:

вхідний, прихований шар (використовується для внутрішнього представлення даних та виконання обчислень, може складатися з 1 або кількох шарів нейронів), [8] вихідний шар (рис. 2.3) MLP можуть використовуватися для задач класифікації об’єктів, (наприклад, класифікації електронних листів,, класифікації текстових даних на категорії), регресії (наприклад, передбачення попиту на товари), розпізнавання тощо.

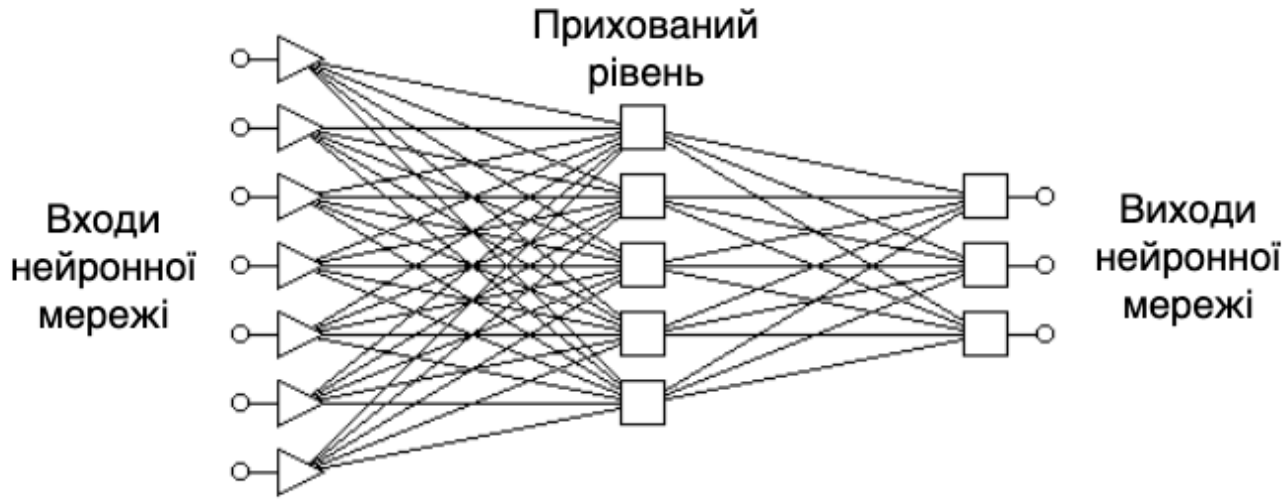

Рисунок 2.3 – Структура багатошарового персептрону

Згортокові нейронні мережі (CNN) є типом мереж, які використовують алгоритму глибокого навчання для класифікації зображень, візуальних елементів. Глибоке навчання дозволяє автоматично витягувати приховані ознаки з великої кількості даних, що дозволяє моделям представляти складні функції апроксимації даних. Згортковий тип нейронних мереж використовує принципи лінійної алгебри, принципи роботи з матрицями для виявлення шаблонів в зображеннях.

Основними компонентами структури CNN є вхідні дані, згортковий шар (використовуються фільтри для виявлення особливостей даних), функції активації, пулінговий шар (використовується для зменшення розмірів зображення, зберігаючи при цьому важливі ознаки) та повнозв’язний шар (з'єднує виходи попередніх шарів в один вектор-дескриптор, який потім використовується для класифікації або регресії) [9]. Схему наведено на рис. 2.4.

Рисунок 2.4 - Модель згорткової нейронної мережі

Рекурентні нейронні мережі (RNN) – це мережи, в яких вхідні дані передаються, як в прямому (вхід-вихід), так і в зворотному напрямках. Завдяки такій структурі нейрони можуть виконувати свої функції повторно. Таким чином інформація може проходити через нейронну мережу декілька разів. Подібного роду петлі, що забезпечуть зворотній зв’язок дають можливість зберігати інформацію.

Основними структурними компонентами RNN є вхідні дані, рекурентний шар (має рекурентні зв’язки що дозволяють передавати інформацію від попередніх часових кроків в майбутні; кожен часовий крок має власні ваги, які оновлюються під час тренування моделі), функція активації, вихідний шар [10]. На рис.2.5 наведено можливі комбінації структури RNN. Зеленим кольором позначена сама мережа, червоним - вхідні дані, а синім - відповідно вихідні.

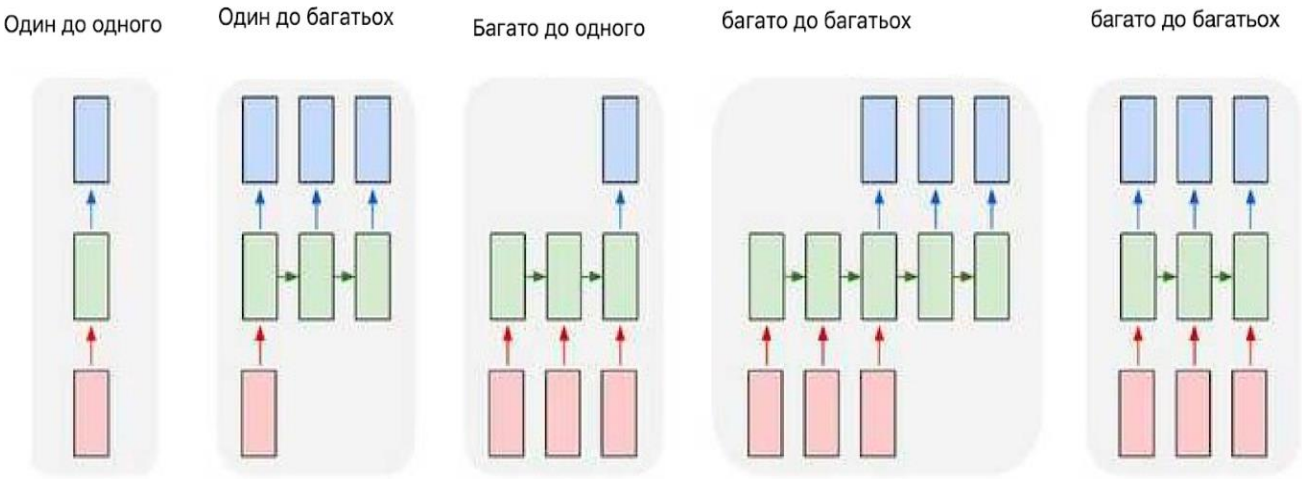

Рисунок 2.5 - Модель RNN

Широкого використання RNN набувають в задачах побудови мовних моделей та в задачах, де важливою є обробка послідовних даних або моделювання залежностей в часі.

Автокодувальник - це архітектура нейронної мережі, яка складається з двох частин: кодувальника (encoder) і декодера (decoder). Кодувальник приймає вхідні дані і перетворює їх в менший розмір, що називається "код" або "проекція". Декодер приймає цей код і відтворює вхідні дані на виході. Внутрішнє представлення (код) між кодувальником і декодером є складною функцією згортки, яка вивчається під час навчання автоенкодера. Одна з головних особливостей автоенкодерів полягає в тому, що вони навчаються без учителя, тобто без міток або класифікації даних. Навчання автоенкодера полягає в тому, щоб мінімізувати різницю між вхідними даними та їх реконструкцією на виході декодера [11]. Це досягається за допомогою функції втрати (loss function), яка порівнює вхідні дані з їх реконструкцією, і оптимізаційним алгоритмом, який оновлює ваги автоенкодера, щоб зменшити цю втрату. Автокодувальники мають широку сферу застосування: генерація нових даних, рекомендаційні задачі, обробка природніх мов тощо.

Висновки до розділу

В рамках цього розділу було сформовано бачення проекту, визначено основних акторів системи, функціональні та нефункціональні вимоги.

Також було проведено аналіз технологій, які можуть використовуватися для аналізу людини за результатами опитувальників. Було визначено, що сучасним способом є використання нейронних мереж. Описано способи класифікації нейронних мереж за їх типом архітектури, топологією та структурою. Вивчено та описано різні підходи до навчання мереж (з вчителем, без вчителя, з підкріпленням), розкрито поняття глибокого навчання. Проаналізовано різні типи нейронних мереж, їх особливості, основні структурні елементи та їх цільове призначення.

3 ВИЗНАЧЕННЯ БАЗОВИХ СТРУКТУР

Після огляду можливих базових структур можна переходити до безпосереднього вибору засобів для реалізації розроблюваної нейронної мережі. Першим кроком має бути збір та підготовка даних. Після чого можна виконати вибір типу нейронної мережі і розглянути моделі, які потенційно можуть вирішити задачі. Обрані моделі мають бути оцінені та порівняні з точки зору ефективності. У разі досягнення недостатнього рівня точності, варто визначити порядок дій для покращення кінцевого результату.

3.1 Підготовка датасету

Перед початком побудови нейронної мережі необхідно визначитися з форматом вхідних та вихідних даних. В загальному, основні кроки формування датасету для розроблюваної системи наведені нижче.

Крок 1. Визначення змінних. Потрібно визначити, які змінні впливають на психологічний стан людини. В даному випадку це будуть результати тестів на психологічні характеристики.

Крок 2. Збір даних. На цьому етапі безпосередньо проводиться опитування людей.

Крок 3. Обробка даних. Після збору даних потрібно їх обробити, привести у відповідний формат, за потреби виконати нормалізацію.

Крок 4. Розбиття на тренувальний та тестовий набори.

Для виконання поставленої мети, людям пропонується пройти тест, який складається з певної кількості питань. Після цього професійних психолог поставить свою оцінку кожному з опитуваних людей. Таким чином буде сформовано датасет. Як було зазначено вище, змінними для розроблювальної системи будуть результати тестування. Кожна відповідь на питання являє собою вибір найбільш підходящого варіанта, який представлено користувачам у вигляді текстового рядку, який потім конвертується в систему балів (від 0 до 10).

Опитувальник включає в себе питання, що стосуються різних змін психологічного стану людини. Наприклад, анкета на основі шкали депресії може включати запитання про настрій, сонливість, інтерес до різних діяльностей і т.д. Деякі питання являють собою питання з існуючих психологічних тестів, наприклад, тест на оцінку депресії Бека чи з популярних тестів на переживання стресу. Питання з тесту Бека в нативному варіанті мають 4 варіанти відповідей, оцінюються за чотирма балами від 0 до 3, але в нашому випадку бали конвертуються в 10-бальну шкалу. Подібного роду опитування можна проводити онлайн або в форматі паперових анкет. В нашому варіанті опитування проводилося посередництвом електронної форми.

Таким чином, аналізуючи відповіді на різні питання, можна зробити висновок стосовно психологічного стану людини. Отже, розроблювана нейронна мережа може натренуватися на подібному датасеті, а саме виконувати задачу мультикласифікації, і в подальшому спростити роботу професійних психологів, оцінюючи людей за тестом, звертаючи увагу на різні параметри, на які, модливо, спеціаліст не зверне увагу. Фактично, нейронна мережа має знайти залежність між результатами тесту та оцінкою спеціаліста. І якщо така дійсно буде існувати, а вона має існувати, то результатом тренування зокрема стануть генералізовані критерії оцінювання створеного тесту.

В запропонованому опитуванні прийняли участь 2100 осіб. Отже можна сформувати датасет, де inputs.shape = (2100, 40), тобто 2100 вхідних датапоінтів зі 40 відповідями на питання. а outputs.shape = (2100,), тобто одна оцінка від психолога відповідно для кожної людини [12]. Розподіл кількості оцінок для кожного стану за оцінкою спеціалістів наведено на рис. 3.1

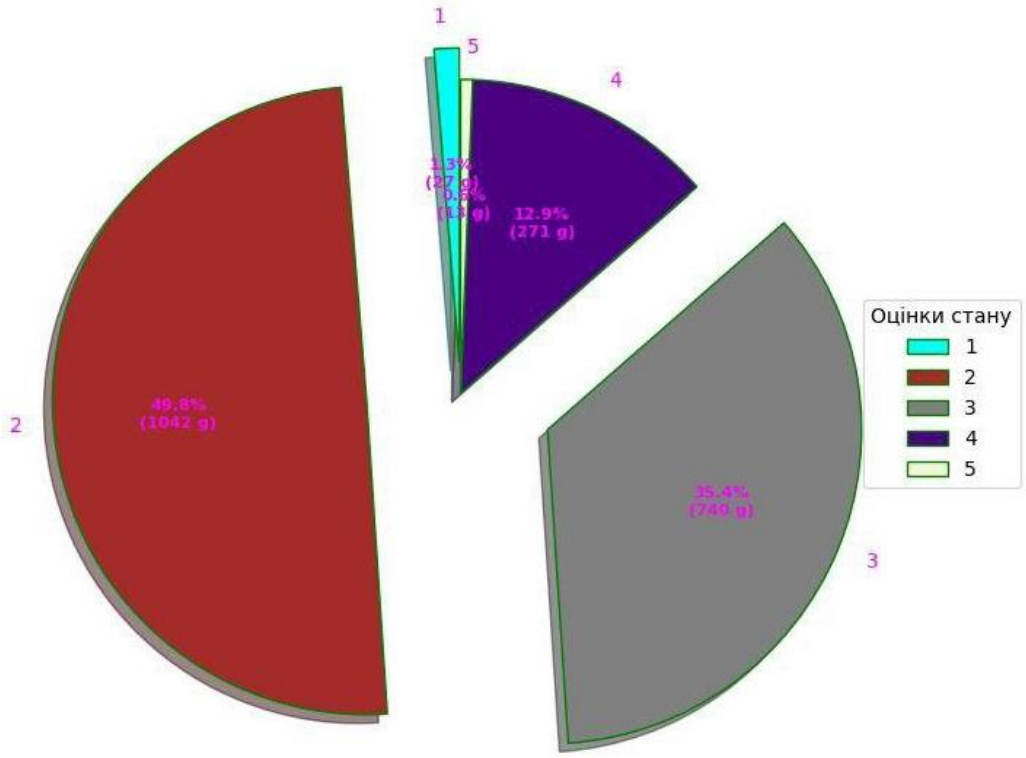

Рисунок 3.1 – Розподіл кількості оцінок для кожного з станів

Наступним кроком є розподіл отриманих даних на тренувальний, валідаційний та тестувальний набори. Тренувальна вибірка використовується суто для навчання мережі. Валідаційна вибірка буде використовуватися для оцінки ефективності моделі під час тренування. Як правило, вона допомагає визначити оптимальні значення гіперпараметрів моделі, які забезпечують найкращі результати. Тестувальна вибірка не буде приймати участь в тренувальному процесі і необхідна для фінальної перевірки.

Дані будуть розоділені на тренувальний, валідаційний та тестувальний набори наступним чином: 1600, 400, 100 відповідно.

3.2 Вибір типу нейронної мережі

В розділі 2 було описано різні типи нейронних мереж, які є ефективними при вирішенні задач класифікації. На остаточний вибір типу нейронної мережі впливають, окрім типу задачі, ще розмір даних, архітектура даних, вимоги до

продуктивності, доступні ресурси. Більш того для точного визначення найбільш ефективного рішення варто провести певний ряд експериментів з різними моделями та спробувати визначити найбільш підходящий варіант.

Вхідними даними для нейронної мережі будуть результати проведено опитування.

Відповідно до теоретичного опису, зазначеного в розділі 2 для вирішення задачі оцінки психологічного стану людини найбільше підходить MLP, оскільки такий тип нейронних мереж призначений для задач, де ми хочемо якийся об’єкт віднести до одного з попередньо визначених класів. Зазвичай при цьому, кількість класів відповідає кількості вихідних нейронів. В нашому випадку буде 5 класів.

3.3 Вибір моделей MLP

Загальна модель багатошарового перцептрону включає в себе різні варіації архітектури DNN (мереж прямого поширення), такі як:

- проста MLP - базова архітектура DNN, яка має вхідний шар, один або більше прихованих шарів та вихідний шар; кожен вузол в прихованих та вихідних шарах використовує функцію активації для обчислення вихідного сигналу [13];
- MLP з різними функціями активації; крім стандартних функцій активації, таких як сигмоїдна, тангенс гіперболічний (tanh) та лінійна функція активації, існують також інші варіанти, такі як ReLU (Rectified Linear Unit), Leaky ReLU, PReLU (Parametric ReLU) та інші;
- MLP з різними варіаціями вагової ініціалізації; ваги вузлів можуть бути ініціалізовані різними способами, такими як випадкова ініціалізація, нормалізована ініціалізація, Xavier ініціалізація та інші;
- MLP з різними методами регуляризації; для запобігання перенавчанню можуть бути використані різні методи регуляризації, такі як L1 регуляризація, L2 регуляризація (відома також як weight decay), Dropout, Batch Normalization та інші;

- MLP з різними оптимізаторами; для оптимізації процесу навчання можуть бути використані різні методи оптимізації, такі як стохастичний градієнтний спуск (SGD), момент (Momentum), RMSprop, Adam, Adagrad та інші.

Для визначення найбільш ефективної моделі нейронної мережі проведемо практичне порівняння. Розглянемо наступні типи моделей: Unet-like, базова MLP, MLP з ReLu активацією.

3.3.1 Функції активації

Функції активації приймає вихідний сигнал з попереднього осередку і перетворює його на деяку форму, яку можна використовувати як вхідні дані для наступного нейрона. В наших моделях буде застосовано наступні функції активації: Sigmoid() та Tanh(), ReLU відповідно. Характеристики обраних функцій наведено нижче.

Сигмоїдальна функція активації є нелінійною функцією, яка приймає будь-яке дійсне число як вхід і відображає його в діапазоні від 0 до 1. Математичний вираз і графік наведено на рис. 3.2 Така функція широко використовується в задачах класифікації, оскільки прагне привести значення до однієї зі сторін кривої (наприклад, до верхнього при x=5 і нижньому при x=-5). Проте при наближенні до кінців сигмоїди значення Y мають тенденцію реагувати на зміни в X. Це означає, що градієнт у таких областях набуває маленьких значень. А це може призводити до проблем із градієнтом зникнення. Отже, при наближенні активаційної функції до майже горизонтальної частини кривої з обох боків, градієнта мале або зникає. І як результат, нейромережа відмовляється навчатися далі або робить це вкрай повільно.

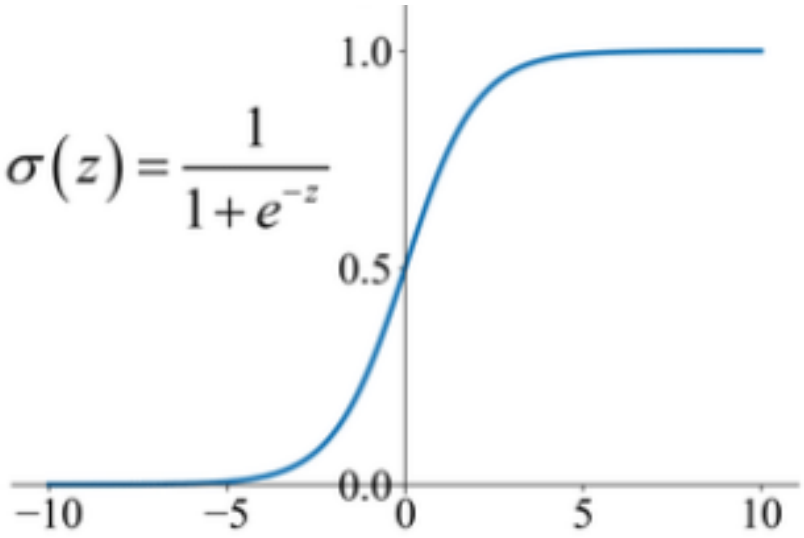

Рисунок 3.2 - Сигмоїдальна функція активації

Гіберболічний тангенс – ще одна нелінійна функція активації, яка приймає дійсне число як вхід і відображає його в діапазоні від -1 до 1. Математичне вираження і графік зображено на рис. 3.3 Така функція має характеристики схожі до сигмоїдної, проте варто відмітити, що градієнт більший. Ця функція зберігає більше інформації про вхідні дані через свій діапазон значень, що може поліпшити здатність нейронної мережі до навчання складних залежностей.

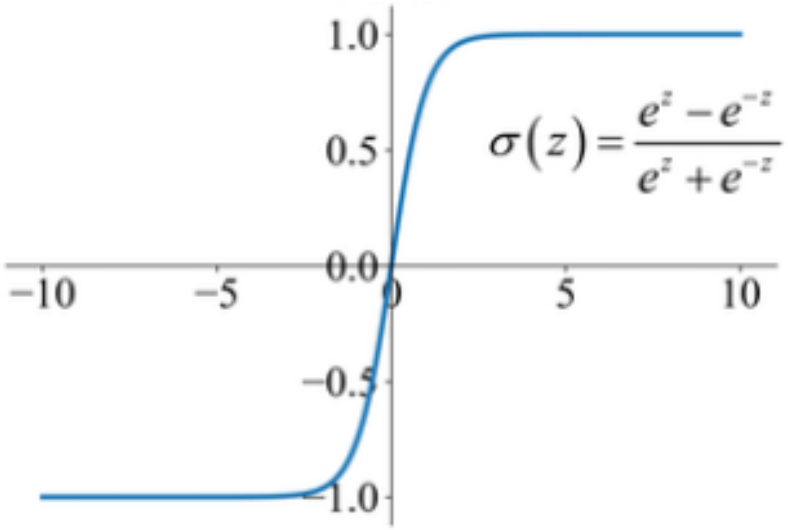

Рисунок 3.3 - Функція активації - гіперболічний тангенс

ReLu – нелінійна функція активації, область значень якої знаходиться в межах [0,inf). Математичне вираження і графік наведено на рис. Одна з головних переваг ReLU полягає в тому, що вона уникне проблеми зі зникненням градієнта, яка може виникнути зі функціями активації sigmoid і tanh [14]. При позитивних значеннях вхідного сигналу, ReLU має постійний та максимально великий градієнт, що сприяє ефективному навчанню моделі. Математичне вираження і графік зображено на рис. 3.4 Проте частина ReLu є горизонтальною лінією (для негативних значень X), а отже градієнт в цій частині дорівнює 0. Отже, ваги не будуть коригуватися під час спуску, а це означає, що нейрони, що знаходяться в такому стані, не будуть реагувати на зміни в вхідних даних. Таке явище називається проблемою вмираючого ReLu (Dying ReLu problem). Через цю проблему деякі нейрони просто вимикаються, роблячи значну частину нейромережі пасивної.

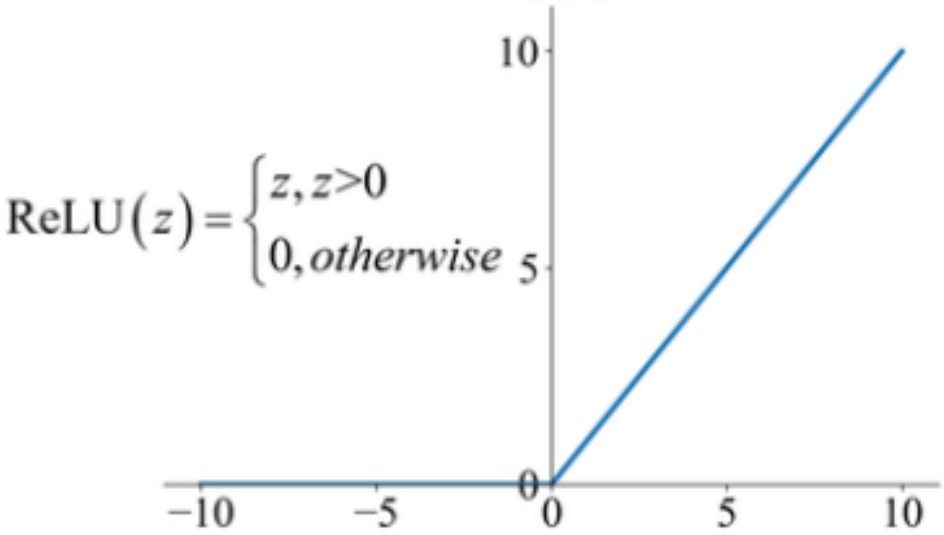

Рисунок 3.4 - Функція активації ReLU

Також існують інші види активаційних функцій, наприклад, softmax, Leaky ReLU, ELU, SELU та інші.

3.3.2 Функція втрат

Необхідно визначитися з типом функції втрат. Загалом, Функція втрат (англ. loss function) - це математична функція, яка вимірює рівень помилки або розбіжності між прогнозованими значеннями моделі і справжніми значеннями цільової змінної. В залежності від типу моделі, можуть бути обрані різні типу функцій, деякі з них описано нижче.

Mean Squared Error (MSE, середнє квадратичне відхилення), даний алгоритм обчислює середнє квадратичне значення різниці між прогнозованими і фактичними значеннями; широко використовується для задач регресії. Дана функція розраховується в декілька етапів.

Етап 1. Від кожного спостереження віднімається відповідне прогнозоване значення, щоб виміряти відхилення.

Етап 2. Кожне відхилення підноситься до квадрату, щоб уникнути впливу від'ємних значень.

Етап 3. Сума квадратів відхилень ділиться на загальну кількість спостережень (n), щоб отримати середнє значення.

Етап 4. Остаточне значення MSE представляє середньоквадратичну відстань між прогнозами і реальними значеннями.

Binary Cross-Entropy (бінарна перехресна ентропія) - функція втрат , яка використовується для бінарної класифікації, де модель прогнозує ймовірність належності до одного з двох класів. Вона оцінює розрив між прогнозованою і фактичною ймовірністю.

Categorical Cross-Entropy (категоріальна перехресна ентропія): Ця функція втрат використовується для багатокласової класифікації, коли модель прогнозує ймовірності належності до кількох класів. Вона вимірює розрив між прогнозованими ймовірностями та фактичними значеннями класів. Процес розрахунку наведено нижче.

Крок 1. Кожне значення вектора істинних міток класу (y_true) множиться на відповідне значення прогнозованої ймовірності класу (y_pred).

Крок 2. Використовується логарифм для обчислення натурального логарифма кожного добутку.

Крок 3. Отримані значення сумуються разом і помножуються на -1, оскільки використовується мінус перед формулою.

Kullback-Leibler Divergence (дивергенція Кульбака-Лейблера), функція втрат використовується для порівняння двох розподілів ймовірностей. Вона широко використовується у задачах генеративних моделей, наприклад, у варіаційному автокод. Процес розрахунку дивергенції розписано нижче.

Крок 1. Обчислюється відношення між ймовірностними розподілами P і Q для кожної події x.

Крок 2. Використовується логарифм для обчислення натурального логарифма цих відношень.

Крок 3. Отримані значення множаться на відповідні ймовірності P(x).

Крок 4. Сума цих добутків дає кінцеву KL-дивергенцію.

В нашому випадку, в якості функції втрат буде використано CrossEntropyLoss. Функція має наступний вигляд

$$H(P,Q) = -\sum_x P(x)\, log\, Q(x), \quad (1)$$

де *P(x)* – розподіл вірних відповідей,

Q(x) – розподіл імовірностей передбачень.

Вона вимірює різницю між передбачуваним розподілом ймовірностей і фактичним розподілом ймовірностей [15]. Значення функції втрати буде максимальним, якщо фактичний клас y не співпадає з передбачуваним класом, тобто коли p_i є найбільшою для неправильного класу. Використання cross-entropy loss допомагає покращити точність моделі.

3.3.3 Оптимізатор

Також, варто обрати оптимізатор, тобто метод, який використовується для налаштування параметрів моделі машинного навчання з метою мінімізації функції втрат. Оптимізатор використовується під час процесу навчання моделі для оновлення ваг і знаходження найкращих значень параметрів. Він працює шляхом обчислення градієнтів (похідних) функції втрат відносно параметрів моделі і використання цих градієнтів для коригування ваг.

Різні типи оптимізаторів можуть використовувати різні стратегії для оновлення параметрів. Але існують і більш розширені оптимізатори, які комбінують різні методи для більш ефективного навчання моделі. Для наших моделей в якості оптимізатора використаємо AdamW, модифікацію алгоритму оптимізації Adam (Adaptive Moment Estimation) яка включає додаткову L2-регуляризацію ваг моделі під час навчання.

L2-регуляризація - це метод регуляризації, який використовується для зменшення перенавчання (overfitting) і поліпшення загальної універсальності моделі машинного навчання [16]. Вона полягає в додаванні штрафу до функції втрат, що базується на квадратичній нормі ваг моделі.

Adam адаптивно налаштовує швидкість навчання для кожного параметра на основі статистики градієнту, що дозволяє ефективно переміщатися вздовж кривої функції втрат. Загальний алгоритм онисано нижче.

Крок 1. Обчислення градієнту за допомогою зворотнього поширення (backpropagation).

Крок 2. Ініціалізація початкових значень для швидкості навчання, експоненціальних коефіцієнтів спуску та квадратів градієнтів.

Крок 3. Оновлення швидкості навчання для кожного параметра, використовуючи оцінки першого та другого моментів градієнтів.

Крок 4. Оновлення параметрів нейронної мережі за допомогою нової швидкості навчання.

3.3.3 Побудова моделей

Розглянемо схеми моделей, що були побудовані на основі визначених кроків, описаних вище. Схеми наведено на рис. 3.5, 3.6 та 3.7 .

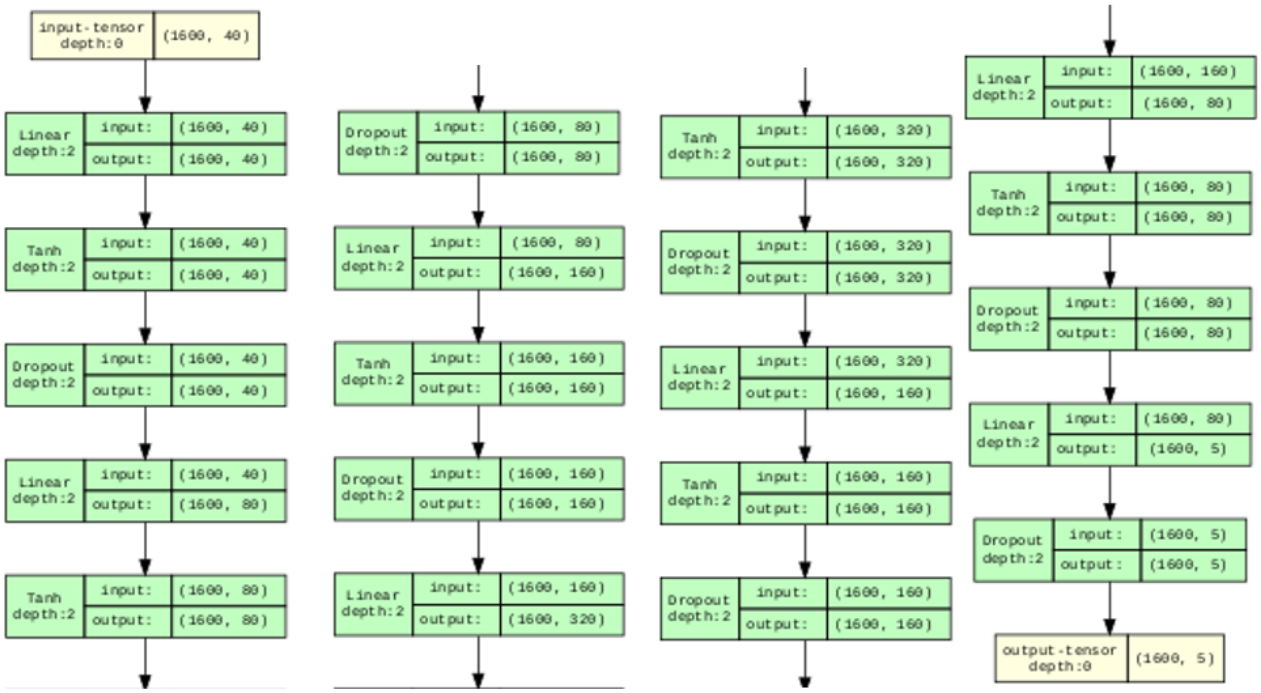

Рисунок 3.5 – Схема базової MLP

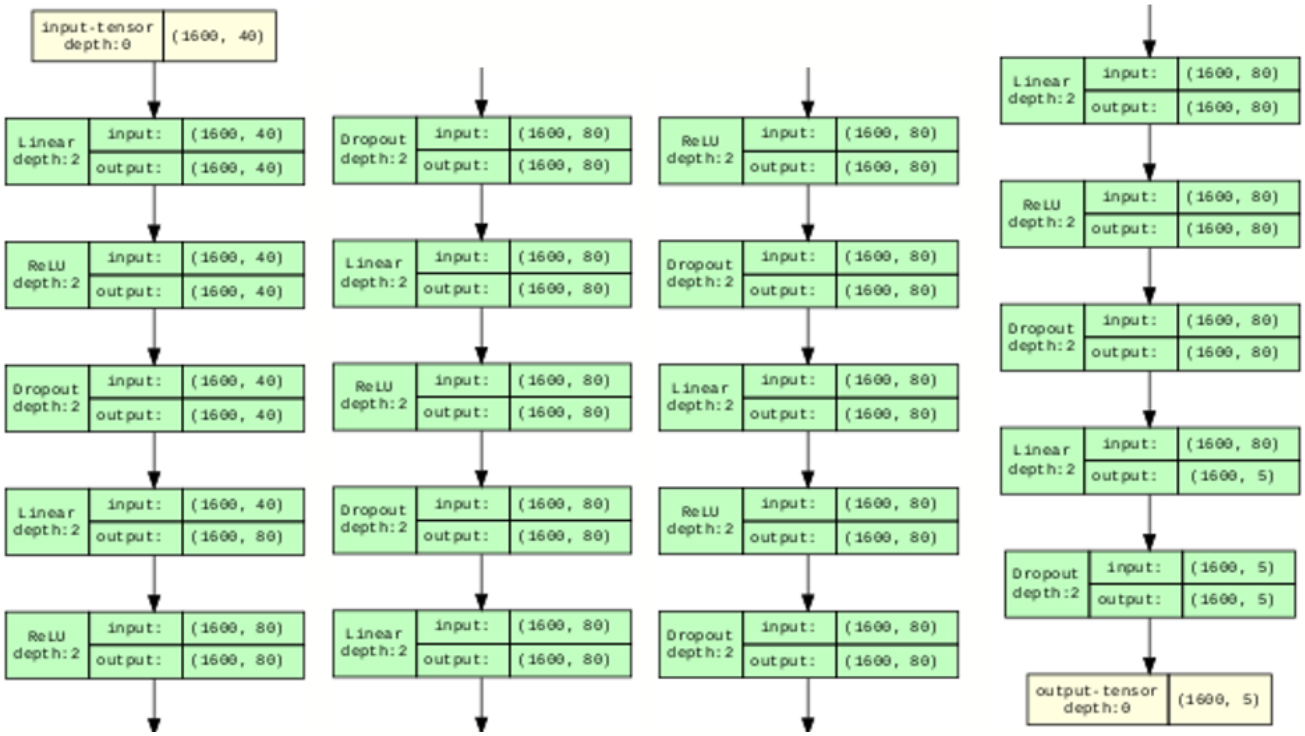

Рисунок 3.6 – Схема MLP з ReLu активацією

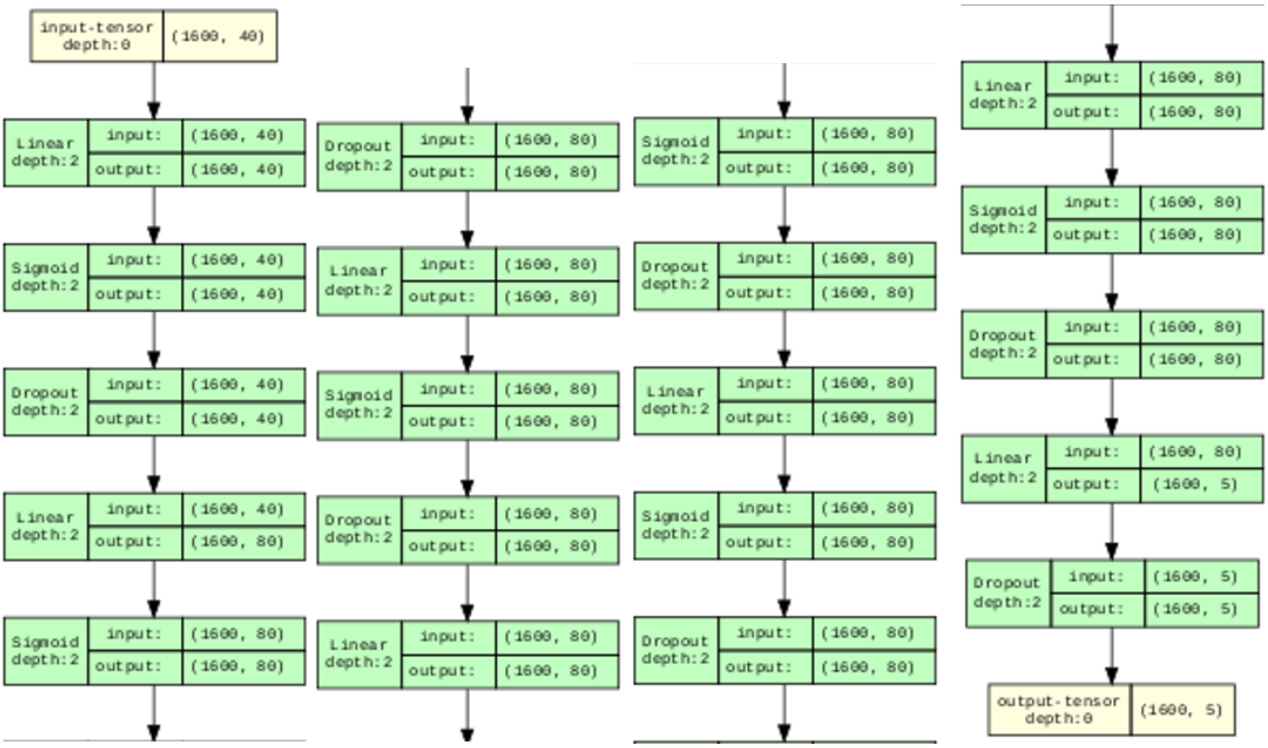

Рисунок 3.7 – Схема моделі Unet-like

Підсумки з параметрами для кожної з моделей наведено на рис. 3.8, 3.9 та 3.10 відповідно. Можна помітити, що обидві базові MLP мають однакову кількість параметрів тренування, а саме 48656, у той час як U-shape MLP має 209156 параметрів, але при цьому для вказаної кількості опитаних людей оперативна пам'ять для такої моделі в два рази більша ніж для аналогів.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1           [-1, 1800, 40]           1,640
              Tanh-2           [-1, 1800, 40]               0
           Dropout-3           [-1, 1800, 40]               0
            Linear-4           [-1, 1800, 80]           3,280
              Tanh-5           [-1, 1800, 80]               0
           Dropout-6           [-1, 1800, 80]               0
            Linear-7          [-1, 1800, 160]          12,960
              Tanh-8          [-1, 1800, 160]               0
           Dropout-9          [-1, 1800, 160]               0
           Linear-10          [-1, 1800, 320]          51,520
             Tanh-11          [-1, 1800, 320]               0
          Dropout-12          [-1, 1800, 320]               0
           Linear-13          [-1, 1800, 160]          51,360
             Tanh-14          [-1, 1800, 160]               0
          Dropout-15          [-1, 1800, 160]               0
           Linear-16           [-1, 1800, 80]          12,880
             Tanh-17           [-1, 1800, 80]               0
          Dropout-18           [-1, 1800, 80]               0
           Linear-19            [-1, 1800, 6]             486
          Dropout-20            [-1, 1800, 6]               0
================================================================
Total params: 134,126
Trainable params: 134,126
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.27
Forward/backward pass size (MB): 34.77
Params size (MB): 0.51
Estimated Total Size (MB): 35.56
----------------------------------------------------------------
```

Рисунок 3.8 - Параметри базової MLP моделі

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1           [-1, 1800, 40]           1,640
              ReLU-2           [-1, 1800, 40]               0
           Dropout-3           [-1, 1800, 40]               0
            Linear-4           [-1, 1800, 80]           3,280
              ReLU-5           [-1, 1800, 80]               0
           Dropout-6           [-1, 1800, 80]               0
            Linear-7           [-1, 1800, 80]           6,480
              ReLU-8           [-1, 1800, 80]               0
           Dropout-9           [-1, 1800, 80]               0
           Linear-10           [-1, 1800, 80]           6,480
             ReLU-11           [-1, 1800, 80]               0
          Dropout-12           [-1, 1800, 80]               0
           Linear-13           [-1, 1800, 80]           6,480
             ReLU-14           [-1, 1800, 80]               0
          Dropout-15           [-1, 1800, 80]               0
           Linear-16           [-1, 1800, 80]           6,480
             ReLU-17           [-1, 1800, 80]               0
          Dropout-18           [-1, 1800, 80]               0
           Linear-19            [-1, 1800, 6]             486
          Dropout-20            [-1, 1800, 6]               0
================================================================
Total params: 31,326
Trainable params: 31,326
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.27
Forward/backward pass size (MB): 18.29
Params size (MB): 0.12
Estimated Total Size (MB): 18.69
----------------------------------------------------------------
```

Рисунок 3.9 – Параметри моделі з ReLU активацією

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1            [-1, 1800, 40]           1,640
           Sigmoid-2            [-1, 1800, 40]               0
           Dropout-3            [-1, 1800, 40]               0
            Linear-4            [-1, 1800, 80]           3,280
           Sigmoid-5            [-1, 1800, 80]               0
           Dropout-6            [-1, 1800, 80]               0
            Linear-7            [-1, 1800, 80]           6,480
           Sigmoid-8            [-1, 1800, 80]               0
           Dropout-9            [-1, 1800, 80]               0
           Linear-10            [-1, 1800, 80]           6,480
          Sigmoid-11            [-1, 1800, 80]               0
          Dropout-12            [-1, 1800, 80]               0
           Linear-13            [-1, 1800, 80]           6,480
          Sigmoid-14            [-1, 1800, 80]               0
          Dropout-15            [-1, 1800, 80]               0
           Linear-16            [-1, 1800, 80]           6,480
          Sigmoid-17            [-1, 1800, 80]               0
          Dropout-18            [-1, 1800, 80]               0
           Linear-19             [-1, 1800, 6]             486
          Dropout-20             [-1, 1800, 6]               0
================================================================
Total params: 31,326
Trainable params: 31,326
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.27
Forward/backward pass size (MB): 18.29
Params size (MB): 0.12
Estimated Total Size (MB): 18.69
----------------------------------------------------------------
```

Рисунок 3.10 – Параметри Unet-like моделі

3.4 Інструменти для тренування

Для навчання нейронних мереж можуть використовуватися різні мови програмування, такі як Python, Java, C++ та інші. Відповідно до кожної мови програмування існує ряд додаткових інструментів і бібліотек, які дозволяють ефективніше вирішувати задачі навчання нейронної мережі.

Такі мови програмування, як наприклад, Java та C# значно рідше використовуються для роботи з нейронними мережами, у порівнянні з Python. Перш за все це пов'язано з швидкістю виконання операцій. Java та C# є мовами програмування загального призначення, які працюють на високому рівні абстракції. Вони забезпечують безпеку типів та багато інших функцій, що збільшують накладні витрати на виконання програми, відповідно при інтенсивному

обчислюванні, такі мови програмування будуть повільніші. Крім того, ці мови програмування мають значно менше інструментів (бібліотек) для роботи з машинним навчанням. Отже, для навчання обраних моделей буде використано мову програмування Python.

Python – це високорівнева кросплатформенна мова програмування загального призначення з динамічною типізацією. Ця мова підходить для об’єктно-орієнтованого програмування, імперативного, функціонального та процедурного. Python має великий екосистему бібліотек, які включають багато інструментів для розв'язання різноманітних задач. Має широку спільноту розробників. Також Python дозволяє інтегрувати код, написаний з використанням інших мов програмування [17].

Для мови програмування Python використовуються багато різних бібліотек для роботи з нейронними мережами. Популярними є:

- TensorFlow - відкрита бібліотека для машинного навчання та глибокого навчання, розроблена компанією Google, ця бібліотека надає потужні інструменти для побудови та навчання нейронних мереж, а також має широкий спектр функцій для обробки даних та візуалізації результатів;
- Keras - високорівневий інтерфейс для роботи з нейронними мережами, який працює поверх TensorFlow, дозволяє легко визначати, навчати та оцінювати моделі нейронних мереж і має зручну API для швидкої розробки;
- PyTorch - фреймворк для глибокого навчання, розроблений компанією Facebook, надає гнучку та ефективну систему для розробки нейронних мереж, зокрема підтримку динамічного графу обчислень. Він також має широкий набір функцій для обробки даних та візуалізації результатів;
- Scikit-learn - бібліотека для машинного навчання, яка надає інструменти для класифікації, регресії, кластеризації та багатьох інших завдань. scikit-learn також має підтримку для нейронних мереж через модуль MLPClassifier;
- Theano - старіший фреймворк для глибокого навчання, який надає швидкість та оптимізації для обчислення графів нейронних мереж, дозволяє ефективно виконувати обчислення;

- MXNet - ще один популярний фреймворк для глибокого навчання, який надає швидкість та масштабованість, має багатофункціональний API для побудови нейронних мереж, підтримує різні типи моделей та оптимізатори, а також має підтримку графічних процесорів (GPU) для прискорення обчислень;
- Caffe - фреймворк для глибокого навчання, спеціалізований на швидкості та ефективності, він надає інструменти для побудови та навчання нейронних мереж, зокрема підтримку згорткових нейронних мереж, рекурентних нейронних мереж та мережі довготривалої пам'яті (LSTM).

Для навчання нейронних мереж та візуалізації результатів навчання буде обрано фреймворк PyTorch.

В якості середовища для розробки буде обрано інтегроване середовище розробки PyCharm, що надає розширені функції та інструменти для розробки, тестування та налагодження програм на Python.

3.5 Процес навчання

Перейдемо до процесу навчання трьох моделей.

Базова MLP показала наступний результат під час тренування, зображений на рис. 3.11.

```
Епоха: 0 | Втрати: 1.741, Точність: 29.31% | Втрати на валідації: 1.696, Точність на валідації: 39.75%
Епоха: 1000 | Втрати: 1.661, Точність: 32.06% | Втрати на валідації: 1.617, Точність на валідації: 39.75%
Епоха: 2000 | Втрати: 1.265, Точність: 44.31% | Втрати на валідації: 0.920, Точність на валідації: 75.75%
Епоха: 3000 | Втрати: 1.073, Точність: 54.56% | Втрати на валідації: 0.654, Точність на валідації: 87.00%
Епоха: 4000 | Втрати: 1.012, Точність: 57.38% | Втрати на валідації: 0.563, Точність на валідації: 88.00%
Епоха: 5000 | Втрати: 0.982, Точність: 58.25% | Втрати на валідації: 0.506, Точність на валідації: 90.50%
Епоха: 6000 | Втрати: 0.929, Точність: 60.38% | Втрати на валідації: 0.460, Точність на валідації: 90.25%
Епоха: 7000 | Втрати: 0.914, Точність: 61.62% | Втрати на валідації: 0.417, Точність на валідації: 94.00%
Епоха: 8000 | Втрати: 0.900, Точність: 61.81% | Втрати на валідації: 0.393, Точність на валідації: 94.50%
Епоха: 9000 | Втрати: 0.904, Точність: 61.56% | Втрати на валідації: 0.365, Точність на валідації: 95.75%
Епоха: 10000 | Втрати: 0.859, Точність: 63.81% | Втрати на валідації: 0.345, Точність на валідації: 96.00%
Епоха: 11000 | Втрати: 0.834, Точність: 63.75% | Втрати на валідації: 0.330, Точність на валідації: 96.00%
Епоха: 12000 | Втрати: 0.809, Точність: 66.56% | Втрати на валідації: 0.317, Точність на валідації: 96.00%
Епоха: 13000 | Втрати: 0.816, Точність: 65.25% | Втрати на валідації: 0.306, Точність на валідації: 96.00%
Епоха: 14000 | Втрати: 0.802, Точність: 67.19% | Втрати на валідації: 0.299, Точність на валідації: 96.00%
Епоха: 15000 | Втрати: 0.807, Точність: 66.69% | Втрати на валідації: 0.292, Точність на валідації: 96.00%
Епоха: 16000 | Втрати: 0.798, Точність: 67.12% | Втрати на валідації: 0.283, Точність на валідації: 96.00%
Епоха: 17000 | Втрати: 0.806, Точність: 64.88% | Втрати на валідації: 0.278, Точність на валідації: 96.00%
Епоха: 18000 | Втрати: 0.749, Точність: 68.25% | Втрати на валідації: 0.271, Точність на валідації: 96.00%
Епоха: 19000 | Втрати: 0.769, Точність: 67.88% | Втрати на валідації: 0.273, Точність на валідації: 95.25%
```

Рисунок 3.11 – Результати тренування базової MLP

Найкращим значенням для точності тренування базової MLP є 70.81, валідації – 96. Графік точності тренування та валідації моделі на 20 тис епох представлено на рис. 3.12. Як видно з графіка рівень точності є доволі низьким, варто перевірити, чи може бути покращена ефективність навчання, змінивши кількість питань, залишивши у вибірці найбільш вагомі питання.

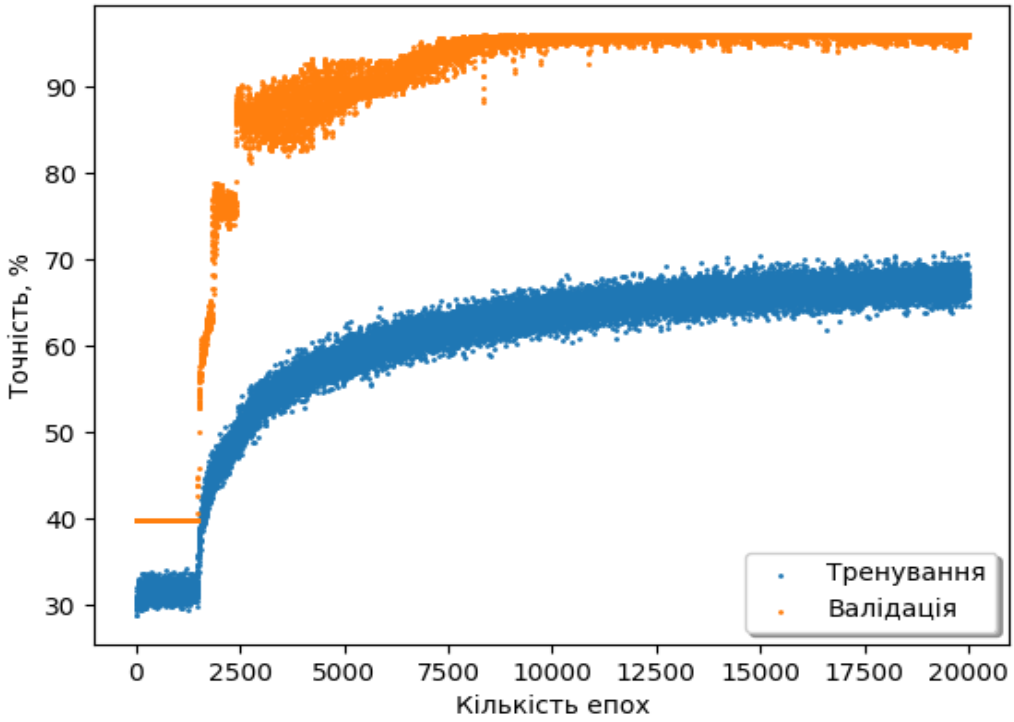

Рисунок 3.12 – Точність тренування та валідації для моделі базової MLP

При спробі зменшити кількість питань точність підвищилася. Результати наведено на рис. 3.13. Суттєва різниця між тренуванням і валідацією обумовлена високим показником dropout (ймовірністю обнулення нейронів в процесі навчання)

	Тренування	Валідація
10	70.81	96.0
20	75.00	97.0
30	80.06	96.0
40	83.50	84.5
50	85.89	75.5

Рисунок 3.13 – Результати тренування і валідації базової MLP з різною кількістю питань

Таке явище пов’язано з тим, що доволі невелика кількість опитаних людей не може покрити велику кількість початкових характеристик і доволі не тривіальну за кількістю параметрів моделі. Але з практичної точки малий обсяг питань не є дуже змістовним, тому є сенс перевірити інші моделі.

Розглянемо базову MLP з ReLU активацією. Результати тренування наведено на рис. 3.14.

```
Епоха: 0 | Втрати: 1.785, Точність: 17.62% | Втрати на валідації: 1.759, Точність на валідації: 51.00%
Епоха: 1000 | Втрати: 0.444, Точність: 80.56% | Втрати на валідації: 6.150, Точність на валідації: 54.25%
Епоха: 2000 | Втрати: 0.378, Точність: 80.31% | Втрати на валідації: 11.279, Точність на валідації: 54.75%
Епоха: 3000 | Втрати: 0.394, Точність: 80.50% | Втрати на валідації: 15.700, Точність на валідації: 52.50%
Епоха: 4000 | Втрати: 0.290, Точність: 84.31% | Втрати на валідації: 17.906, Точність на валідації: 50.00%
Епоха: 5000 | Втрати: 0.319, Точність: 82.62% | Втрати на валідації: 19.909, Точність на валідації: 50.50%
Епоха: 6000 | Втрати: 0.292, Точність: 84.31% | Втрати на валідації: 24.731, Точність на валідації: 51.00%
Епоха: 7000 | Втрати: 0.315, Точність: 82.75% | Втрати на валідації: 17.207, Точність на валідації: 50.50%
Епоха: 8000 | Втрати: 0.287, Точність: 83.75% | Втрати на валідації: 17.539, Точність на валідації: 50.00%
Епоха: 9000 | Втрати: 0.320, Точність: 82.94% | Втрати на валідації: 16.710, Точність на валідації: 50.50%
Епоха: 10000 | Втрати: 0.299, Точність: 82.25% | Втрати на валідації: 17.339, Точність на валідації: 50.50%
Епоха: 11000 | Втрати: 0.347, Точність: 80.94% | Втрати на валідації: 13.638, Точність на валідації: 50.75%
Епоха: 12000 | Втрати: 0.326, Точність: 82.44% | Втрати на валідації: 18.515, Точність на валідації: 50.50%
Епоха: 13000 | Втрати: 0.306, Точність: 82.75% | Втрати на валідації: 11.457, Точність на валідації: 49.50%
Епоха: 14000 | Втрати: 0.335, Точність: 79.75% | Втрати на валідації: 12.264, Точність на валідації: 49.75%
Епоха: 15000 | Втрати: 0.322, Точність: 81.44% | Втрати на валідації: 11.224, Точність на валідації: 49.75%
Епоха: 16000 | Втрати: 0.361, Точність: 80.19% | Втрати на валідації: 15.361, Точність на валідації: 50.00%
Епоха: 17000 | Втрати: 0.383, Точність: 79.06% | Втрати на валідації: 12.395, Точність на валідації: 48.25%
Епоха: 18000 | Втрати: 0.354, Точність: 81.31% | Втрати на валідації: 11.943, Точність на валідації: 48.00%
Епоха: 19000 | Втрати: 0.421, Точність: 78.00% | Втрати на валідації: 11.012, Точність на валідації: 46.75%
```

Рисунок 3.14 – Результати тренування базової MLP з ReLU активацією

Найкращим значенням тренування базової MLP з ReLU активацією – 87,38, валідації – 57. Графік точності тренування та валідації цієї моделі представлено на рис. 3.15.

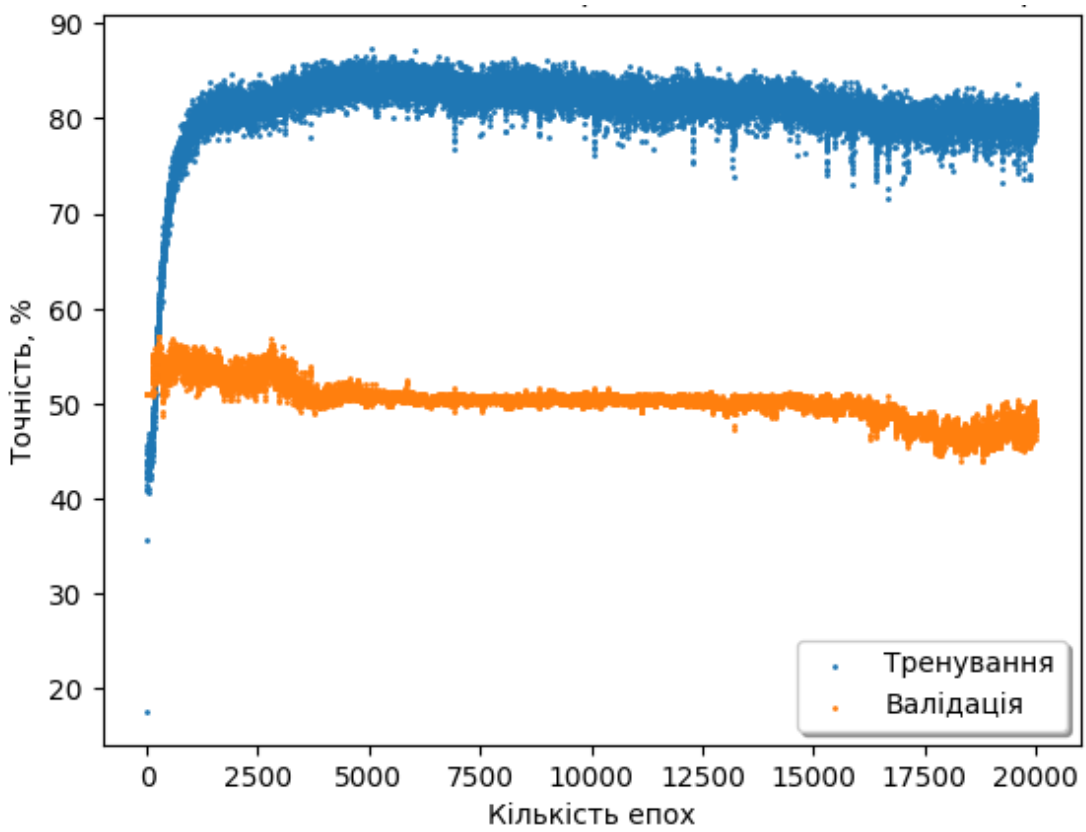

Рисунок 3.15 – Точність тренування та валідації для моделі MLP з ReLU активацією

При спробі зменшити кількість питань, точність змінювалася незначно. Результати наведено на рис.3.16. Можна помітити, що функція активації в даному випадку взагалі не була застосована. Є сенс розглянути іншу модель.

	Тренування	Валідація
10	81.44	40.00
20	86.81	45.75
30	87.94	58.50
40	83.94	52.50
50	87.38	57.00

Рисунок 3.16 – Результати тренування і валідації MLP з з ReLU активацією при різній кількості питань

Розглянемо Unet-like модель. Результати тренування наведено на рис. 3.17.

```
Епоха: 0 | Втрати: 1.786, Точність: 18.00% | Втрати на валідації: 1.664, Точність на валідації: 51.50%
Епоха: 1000 | Втрати: 0.891, Точність: 62.38% | Втрати на валідації: 1.060, Точність на валідації: 56.00%
Епоха: 2000 | Втрати: 0.796, Точність: 66.94% | Втрати на валідації: 1.055, Точність на валідації: 56.50%
Епоха: 3000 | Втрати: 0.454, Точність: 78.69% | Втрати на валідації: 0.453, Точність на валідації: 86.25%
Епоха: 4000 | Втрати: 0.400, Точність: 80.62% | Втрати на валідації: 0.503, Точність на валідації: 86.25%
Епоха: 5000 | Втрати: 0.357, Точність: 81.00% | Втрати на валідації: 0.515, Точність на валідації: 88.00%
Епоха: 6000 | Втрати: 0.337, Точність: 82.62% | Втрати на валідації: 0.772, Точність на валідації: 82.75%
Епоха: 7000 | Втрати: 0.289, Точність: 83.69% | Втрати на валідації: 0.595, Точність на валідації: 87.00%
Епоха: 8000 | Втрати: 0.320, Точність: 82.88% | Втрати на валідації: 0.607, Точність на валідації: 86.50%
Епоха: 9000 | Втрати: 0.266, Точність: 85.62% | Втрати на валідації: 0.586, Точність на валідації: 89.50%
Епоха: 10000 | Втрати: 0.342, Точність: 82.31% | Втрати на валідації: 0.654, Точність на валідації: 85.75%
Епоха: 11000 | Втрати: 0.229, Точність: 85.62% | Втрати на валідації: 0.559, Точність на валідації: 90.50%
Епоха: 12000 | Втрати: 0.279, Точність: 85.19% | Втрати на валідації: 0.559, Точність на валідації: 90.00%
Епоха: 13000 | Втрати: 0.236, Точність: 86.00% | Втрати на валідації: 0.584, Точність на валідації: 90.25%
Епоха: 14000 | Втрати: 0.251, Точність: 83.62% | Втрати на валідації: 0.603, Точність на валідації: 90.00%
Епоха: 15000 | Втрати: 0.242, Точність: 85.69% | Втрати на валідації: 0.630, Точність на валідації: 88.50%
Епоха: 16000 | Втрати: 0.218, Точність: 87.31% | Втрати на валідації: 0.636, Точність на валідації: 89.00%
Епоха: 17000 | Втрати: 0.245, Точність: 85.56% | Втрати на валідації: 0.610, Точність на валідації: 90.25%
Епоха: 18000 | Втрати: 0.211, Точність: 86.31% | Втрати на валідації: 0.606, Точність на валідації: 90.00%
Епоха: 19000 | Втрати: 0.282, Точність: 85.62% | Втрати на валідації: 0.790, Точність на валідації: 84.75%
```

Рисунок 3.17 – Результати тренування Unet-like моделі

Найкращими значеннями точності тренування є 88.88, валідації – 92. При цьому з самого початку значення тренування і валідації доволі сильно розходилися, але після 10 тисяч епох зійшлися. Графік наведено на рис. 3.18.

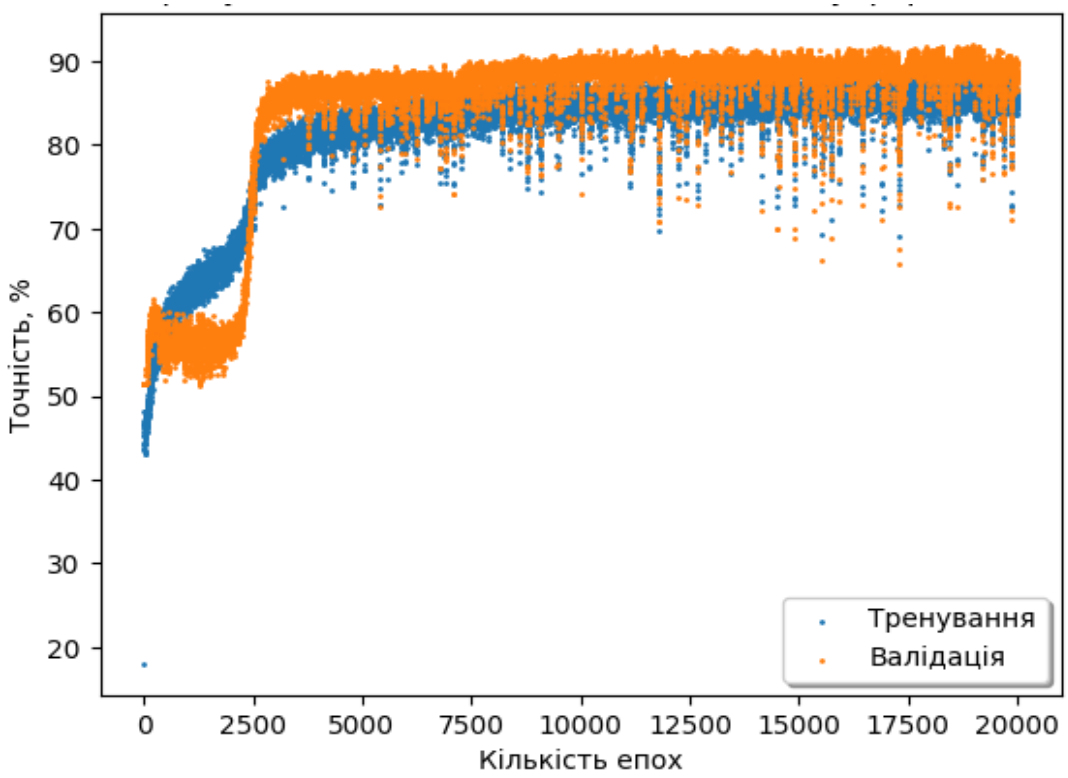

Рисунок 3.18 – Точність тренування та валідації для моделі Unet-like

Значення точності тренування і валідація при спробі змінити кількість питань наведено на рис. 3.19. Ця модель у порівнянні з іншими, значно краще поводить себе на більшій кількості питань, що має для нас сенс.

	Тренування	Валідація
10	86.75	100.00
20	87.38	97.75
30	87.31	92.00
40	88.88	92.00
50	88.31	83.75

Рисунок 3.19 – Результати тренування і валідації Unet-like моделі

Схему фінальної мережі наведено в додатку А.

На тестовій вибірці з найкращою кількістю питань, з якою початково формувався датасет (40 питань) була отримана точність 90%. Для завершення перевірки ефективності моделі побудовано confusion-матрицю, яка зобразить кількість правильних/неправильних класифікацій для кожного класу і загальні точність класифікації. Матрицю наведено на рис.3.20.

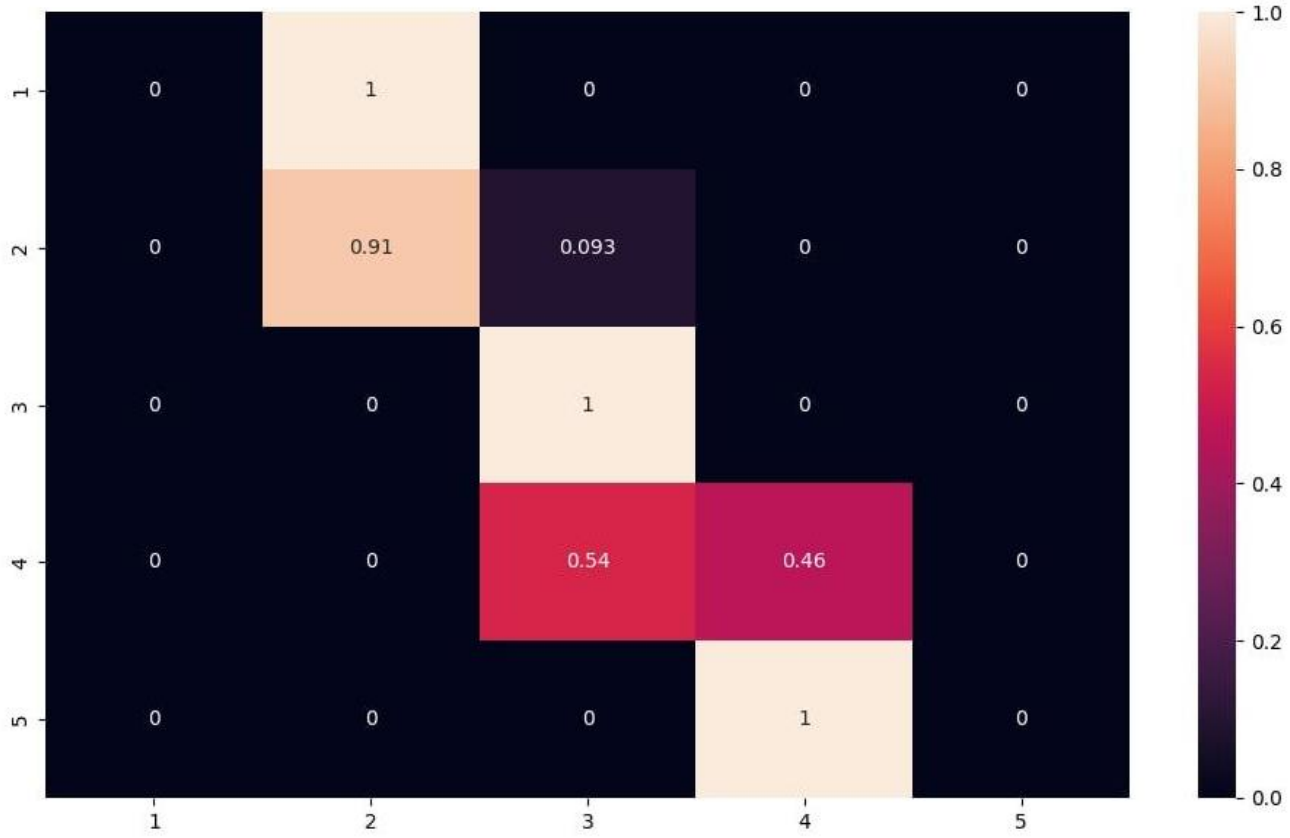

Рисунок 3.20 – Confusion-матриця для обраної моделі

З матриці можна зробити висновок, що у випадках, де кількість оцінок стану значуща (для двійок - 43 відсотки на тесті та майже 50 на тренуванні, для трійок - 41 на тесті та 35 на тренуванні), імовірність їх вгадування теж задовільна (91 відсоток для 2 і 100 для 3).

Оцінка 4 була поставлено значно меншій кількості людей (на тестовому наборі - 13, на тренуванні 12), і розроблювана мережа впоралась на 50%, що є, очевидно незадовільним. Те ж саме стосується і станів 1 та 5, які присутні у вибірці у 1 та 2 відсотках відповідно. Зокрема слід зауважити, що не зважаючи на високий середній відсоток вгадування, оцінка окремих класів має бути покращена.

3.6 Підвищення ефективності навчання

Можна спробувати покращити результат нейронної мережі, оскільки рівень точності варто підняти і процес тренування відбувався не достатньо швидше

(значна точність набувається починаючи зі трьох тисяч епох). Спробуємо застосувати один з класичних підходів для подібного роду випадків – нормалізацію початкових даних, тобто приведення значень, можливих у датасеті, до єдиного діапазону. До основних методів нормалізації відносять: scaling to a range, стандартизацію, normalization by decimal scaling, max normalization.

Scaling to a range – метод нормалізації, формулу для якого наведено нижче

$$x' = \frac{x - min}{\max - min}, \tag{2}$$

де x – початкове значення, x' – нормалізоване значення;

min та max – мінімальне та максимальне значення вхідних даних.

Такий підхід є корисним у випадках, коли чітко відома верхня і нижня межі даних і вони приблизно розподілені у такому діапазоні.

Ще одним з популярних підходів є стандартизація. Формула виглядає наступним чином

$$x' = \frac{x - mean}{std}, \tag{3}$$

де *x'* – нормалізоване значеня, *x* – початкове значення;

mean - середнє значення вхідних даних;

std - стандартне відхилення вхідних даних.

В такому випадку дані не будуть зосереджені в певному діапазоні. Таким чиноом дані будуть більш центровані навколо середнього значення.

Також можна відмітити метод нормалізацції by decimal scaling. Формула виглядає наступним чином:

$$x_normalized = \frac{x}{10^d}, \tag{4}$$

де *x_normalized* – нормалізоване значення, *x* - початкове значення точки даних; *d* - кількість десяткових знаків, необхідних для представлення максимального абсолютного значення в наборі даних.

Ділення кожної точки даних на відповідну степінь числа 10 зсуває та масштабує значення до десяткового діапазону, роблячи їх порівнянними і зменшуючи вплив викидів [18].

Також для нормалізації даних застосувують підхід max normalization. Він використовується для приведення значень даних до діапазону від 0 до 1. Формула виглядає наступним чином:

$$x' = \frac{x}{max}, \tag{5}$$

де *x'* – нормалізоване значеня, *x* – початкове значення;

max - максимальне значення вхідних даних

.

Такий метод може бути корисним, коли варто зменшити значення, щоб певна частина з них не домінувала над іншими.

Також існують методи логаріфмічної нормалізації, Robust нормалізація, Unit Vector нормалізація.

Для нормалізації даних для поставленої задачі буде обрано метод максимальної нормалізації. Як було зазначено, суть цього метода полягає у поділі значення кожного атрибуту (в нашому випадку, відповіді на питання) на максимальне значення (в нашому випадку – 10). Таким чином, ми зможемо нормалізувати дані до вибірки в діапазоні [0;1].

Після нормалізації даних отримали наступні показники ефективності навчання: найкраще значення для точності - 92.50, для валідації - 97.25 (рис. 3.21). З графіка можна побачити, що мінімум функції втрат досягається в районі 1500 епохи, що приблизно у два рази швидше за результат роботи мережі без початкової нормалізації даних. Окрім цього, точність на валідації зросла до 97.25%.

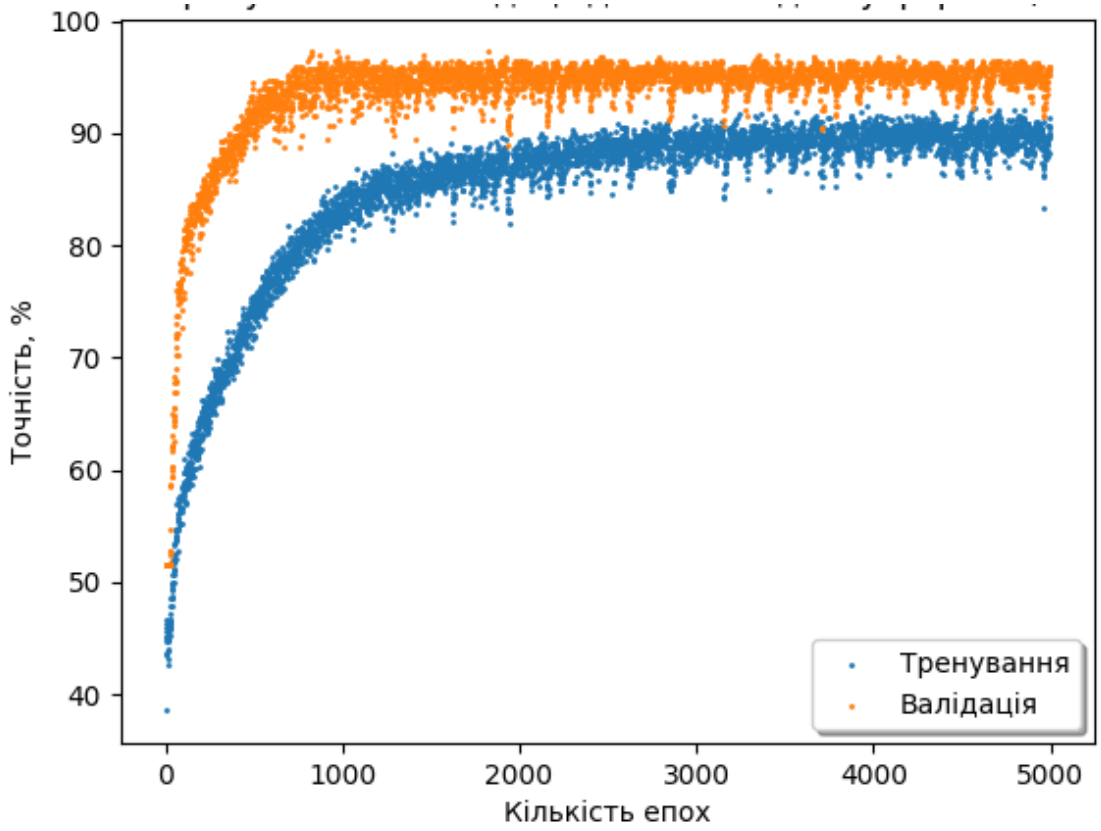

Рисунок 3.21 – Точність тренування та валідації для моделі Unet-like після нормалізації даних

Фінальна confusion-матриця (рис. 3.22) показує, що на тестовій вибірці точність становить трохи менше, проте сягає 93-100%. Підсумовуючи вищезазначене, нормалізація вхідних даних дала значний приріст точності створюваної нейронній мережі.

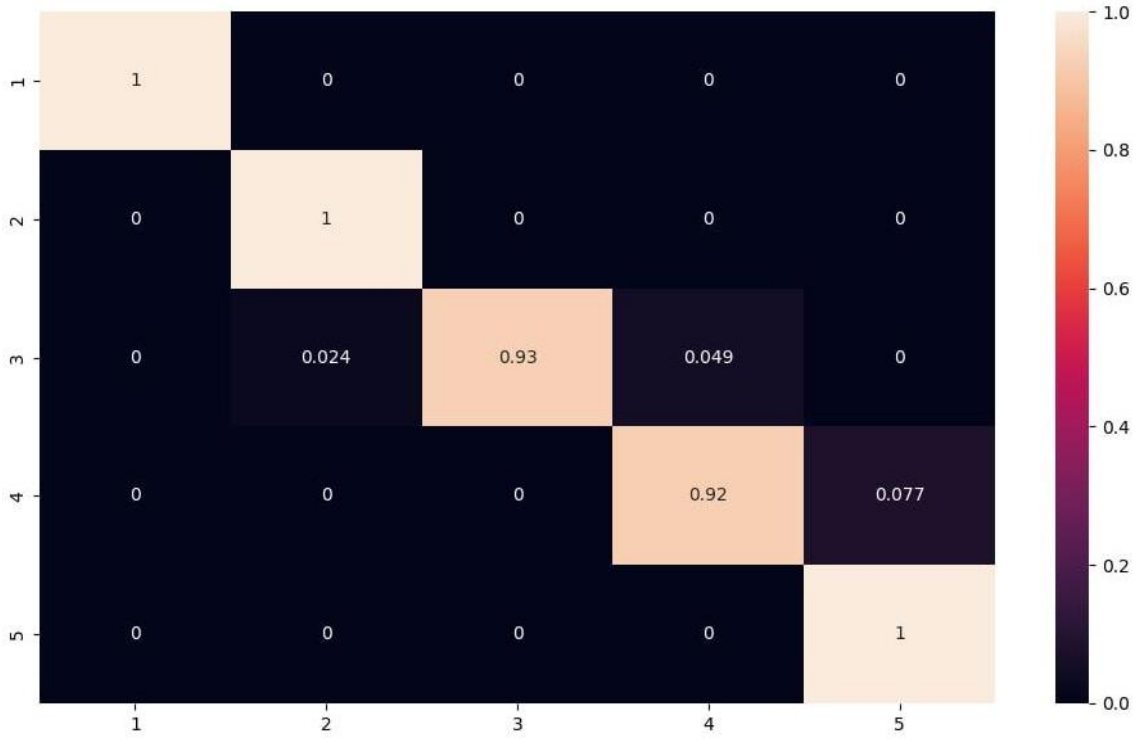

Рисунок 3.22 – Confusion-матриця після нормалізації даних

Судячи з матриці, середній відсоток вдалих передбачень зріс майже на 10%. Крім того розроблювана мережа більш вдало відпрацьовує у зонах малої густини, тобто навіть для тих класів, тобто оцінок, які представлені у недостатній кількості. Загалом, такий рівень точності можна вважати достатнім.

Різниця вгадування класу мережею обумовлена невеликою вибіркою для такого роду задачі. Якщо збільшити вибірку і змінити розподіл даних (рис. 3.1), додати більше відповідей, які попадуть в класи 4, 5, а також 1, то результати будуть отримані кращі.

Висновки до розділу

В розділі було зроблено огляд процесу формування датасету. Описано процес побудови нейронної мережі для оцінки психологічного стану людини у вигляді оцінок за 5-бальною шкалою. Описано процес вибору функції активації, функції втрат, оптимізатора. Виконано огляд декількох різних моделей MLP, проведено оцінки їх точності, а також наведено показники точності для моделей при спробі

зміни кількості початкових питань. Найкращим чином показала себе Unet-like модель з точністю оцінки близько 90%, 91% і більше на класах, представники яких доволі часто зустрічаються у вибірках, і близько 50% на зонах малої густини, що не є дуже високим показником ефективності і швидкості навчання загалом.

В роботі описані можливі підходи до нормалізації даних. Виконано нормалізацію вхідних даних шляхом ділення кожного значення атрибутів на максимальне значення, що дозволило збільшити точність нейронної мережі до 93%, на класах, які частіше зустрічаються у вибірці вище, загалом середній відсоток вдалих передбачень зріс майже на 10%. Було побудовано відповідні confusion-матриці.

Результати навчання мережі можуть бути покращені, якщо розширити вибірку більшою кількістю оцінок 4 і 5, а також більшою кількістю оцінок 1.

4 ПОБУДОВА СИСТЕМИ

Після визначення вимог і базових структур наступним етапом є розробка система. Першим кроком є визначення загальної архітектури системи, враховуючи всі визначені вимоги. Загалом, можна виділити 2 частини в архітектурі системи - клієнтський інтерфейс і серверна частина. Клієнтська частина відповідає за взаємодія з користувачем, а задача серверної частини полягає в обробці всіх даних і логіці програми.

Кожен компонент загальної системи має бути окремо проаналізований з метою визначення архітектурного підходу до його розробки.

4.1 Вибір типу архітектури

Вибір типу архітектури в першу чергу залежить від формату комунцікації системи з кінцевим користувачем, а також від навантаження на систему та від вимог, як варто розподіляти навантаження, тобто від нефункціональних вимог до розробки системи, що були визначені у розділі 2.

4.1.1 Вибір мессенджера

Розроблювана система з точки зору користувацького інтерфейсу має бути ботом у месенджері, тобто програмою, яка розроблена для автоматизованої комунікації з користувачами через месенджери, такі як Telegram, Viber, Facebook Messenger, WhatsApp і багато інших.

У порівнянні зі звичними сайтами мессенджер-боти мають певні переваги:

- не треба звикати до нового користувацького інтерфейса; відповідно і командам, які займаються розробкою програми, можуть не витрачати багато часу на розробку користувацького інтерфейсу;
- немає необхідності витрачати час на тестування фронтенд системи, що в принципі, в результаті дешевше;

- може використовуватися з будь-якого девайсу за наявності встановленого браузера чи застосунку до месенджера та доступу до інтернету;
- вони не обмежені жорсткими розмірами екрану веб-сторінок;
- більша вірогідність отримати важливу інформацію, оскільки користувачі користуються мессенджерами декілька разів на день.

Існують 3 типи ботів:

- rule-based;
- hybrid;
- AI.

Rule-based боти – це ті, які надають користувачеві вибір параметрів, які потрібно натиснути, щоб отримати відповідь на конкретний запит.

AI чат-боти, які використовують штучний інтелект, обробку природної мови (NLP) і машинне навчання, розуміють різноманітні ключові слова та фрази та навчаються на введенні відвідувачів. Ці боти з часом навчаються розуміти більше запитів і різні способи, як клієнти формулюють запитання.

Гібридні чат-боти поєднують ботів на основі правил і технологію AI. Вони виконують деякі завдання на основі правил, але вони також можуть визначити контекст і наміри користувача.

Для розроблюваної системи буде обрано Rule-Based чат-бот.

В якості базового месенджера за замовчуванням, для MVP розробки було обрано Telegram, один з найпопулярніших месенджерів, що надає можливість обмінюватися повідомленнями, зображеннями, відео, а також іншими файлами при підключенні до інтернету. Серед основного функціоналу цього месенджера можна виділити:

- приватні та групові чати: користувач має можливість надсилати повідомлення одному користувачу приватно або створювати групові чати з кількома учасниками;
- хмарне зберігання та синхронізація: можливість зберігання повідомлень, медіа-файлів та документів в хмарному сховищі, що дозволяє

отримувати доступ до них з будь-якого пристрою, більш того завдяки цьому досягається синхронізацію між пристроями;

- криптографічний захист: надійне шифрування для захисту ваших повідомлень та файлів, що передаються по мережі;
- самознищення повідомлень: можливість встановлення таймера для самознищення для повідомлень, щоб вони автоматично видалялися через заданий час після читання або відправки;
- голосові та відеовиклики: підтримка голосових та відеовикликів, що дозволяють зв'язуватися з іншими користувачами у режимі реального часу;
- канали та боти: створення публічних канали, де можете публікувати повідомлення для широкого загалу, а також використовувати ботів для автоматизації деяких завдань та отримання обробленої інформації.

4.1.2 Вибір загальної архітектури

Месенджер бот є точкою входу в систему, веб-клієнтом який буде надсилати запити до сервісів, що в свою чергу будуть обробляти інформацію і звертатися за даними до сховищ даних. Отже, архітектура системи є клієнт-серверною з веб-браузером. Загальну схему роботу клієнт-серверної архітектури наведено на рис. 4.1.

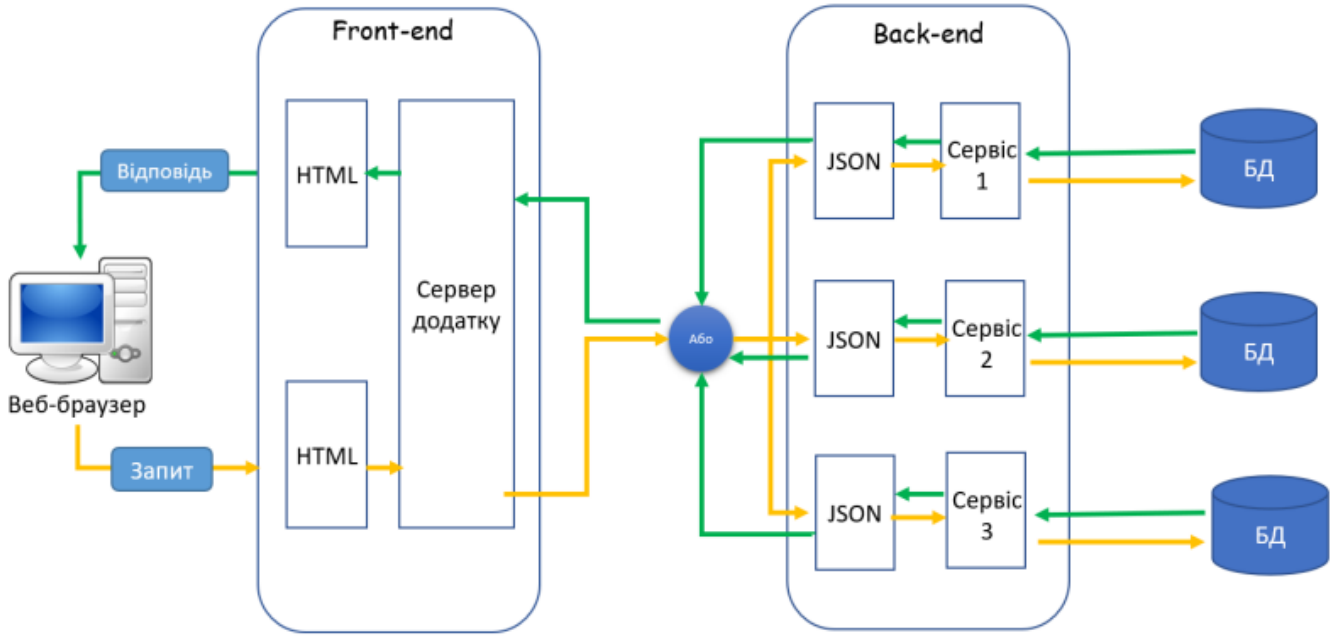

Рисунок 4.1 - Загальна схему функціонування клієнт-серверної архітектури

Перевагою такого типу архітектури є відсутність потреби у вивченні інтерфейсу прикладної програми. А отже це спрощує процес навчання користування програмою. Месенджер-бот – це програма, яку користувачі використовують декілька разів на день.

Також, як перевагу можна виділити незалежність від того, який девайс у користувача та яка операційна система на ньому встановлена, оскільки користувач взаємодіє лише з месенджером, який є доступним у веб-версії або у форматі мобільного застосунку, який зневажає на внутрішній устрій комп'ютера, видаючи інформацію у вигляді переписки. Програма доступна будь-якому користувачу в мережі Internet, у якого є права на доступ звертатися до них. Оскільки всі операції знаходяться на сервері (може бути будь-який сервер, що підтримує протокол HTTP), то зникає необхідність супроводження та модернізації частини додатків, які знаходяться на машинах-клієнтах. Така конфігурація здатна забезпечити роботу десятків тисяч, навіть мільйонів користувачів безперебійно.

Проте у цього типу архітектури є і недоліки. Оскільки весь додаток виконується на сервері, то при його відмові робота програма може повністю зупинитися. У разі виникнення проблем з бекенд-частиною, користувачу відобразиться візуальна частина з повідомленням про якусь проблему. Крім того для розподілення навантаження у разі великої кількості користувачів, логіку додатку буде розбито на декілька мікросервісів, що будуть комунікувати між собою через HTTP-протокол або через чергу повідомлень, що дає гарантію того, що повідомлення залишиться у черзі, поки його не буде прочитано сервісом. Останній тип комунікації може бути досить корисним при нотифікуванні користувачів.

Також серед недоліків можна виділити чітку залежність на месенджер. У разі виникнення якоїсь проблеми з месенджером, у користувача немає можливості отримати результати роботи програми.

4.2 Архітектура бекенд-сервісів

Розглянемо 2 підходи до створення архітектури системи: моноліт і мікросервіси. Монолітна архітектура являє собою підхід, при якому вся логіка і всі компоненти розробляються в рамках однієї системи. Такий підхід є простим з точки зору тестування, оскільки всі частини системи взаємодіють одна з одною безпосередньо в рамках одного сервісу. Оскільки все програмне забезпечення написане з використанням однієї технологічної платформи, програма є більш керованою. Проте такі системи складно масштабувати, оскільки всі компоненти програму залежать один від одного. Подібна архітектура буде хорошим вибором для невеликих проєктів, які не потребують подальшого розширення чи масштабування.

Ідея мікросервісного підходу полягає в тому, що програма складається з кількох невеликих самомтійних сервісів, які не є тісто зв’язаними між собою. Мікросервіси можуть комунікувати один з одним 2 способами: синхронним і асинхронним. При синхронному підході використовується технологія REST, тобто для виконання запиту і отримання відповіді використовується протокол HTTP. В даному випадку для нас важливим фактором є отримання відповіді після кожного запиту. В асинхронному підході отримання відповіді, тобто інформація щодо того, чи виконався запит чи ні – не є вкрай важливою. Ми передаємо дані в чергу повідомлень і очікуємо, що ці дані будуть оброблені [19].

Такий підхід є більш практичним для великих систем, які активно розробляються та повинні бути розширені найближчим часом. Кожен сервіс, як окрема логічна частина системи, яка може розроблятися окремою командою з використанням іншого сету технологій, запускається набагато швидше, оскільки є меншою. Кожен сервіс розгортається окремо, тобто при внесенні змін в один сервіс, інший продовжує працювати. Кожен сервіс може окремо масштабуватися, якщо очікується більше навантаження на певну частину системи. Кожен сервіс окремо тестується простіше, але можуть виникати проблеми на етапі тестування взаємодії всіх елементів системи.

З огляду предметної області та зважаючи на те, що система має бути легко розширювана новим функціоналом і містити різні за логікою частини: опитувальники, робота з психологами і проведення консультацій, керування користувачами. Відповідно краще обрати мікросервісний підхід.

Буде виділено наступні сервіси:

- сервіс користувачів (аутентифікації) - відповідає за інформацію, пов’язану з даними користувачів, перевіряє правильність введення даних для логіну в систему, а також перевіряє ролі та пермісії користувачів;
- сервіс психологів та дотичних спеціалістів - відповідає за інформацію пов’язану з психологами, їх спеціалізаціями та роботу з клієнтами;
- сервіс опитувальників - відповідає за роботу з опитувальниками для оцінки психологічного стану, формує оцнку шляхом інтеграції з нейронною мережею;
- сервіс нотифікацій - нотифікує користувача про важливі зміни.

Додатково до логічних (смислових) сервісів необхідно створити service registry та API Gateway.

При звичному, уже можна сказати, застарілому, способі розгортанні системи, сервіси розгортаються на відомих портах або хостах та можуть бути легко знайдені і виклекані один одним. Проте у сучасних системах кількість екземплярів кожного сервісу може змінюватися в залежності від потреб динамічно, а отже варто реалізовувати механізм, який буде виконувати запити до якогось конкретного екземпляру.

У цьому підході, замість того, щоб клієнтам бути свідомими про конкретні мікросервіси та їх розташування, використовується централізований компонент, відомий як сервер виявлення (discovery server), який відповідає за реєстрацію, виявлення та надання інформації про доступні мікросервіси. Загальне схема роботи наведена на рис. 4.2.

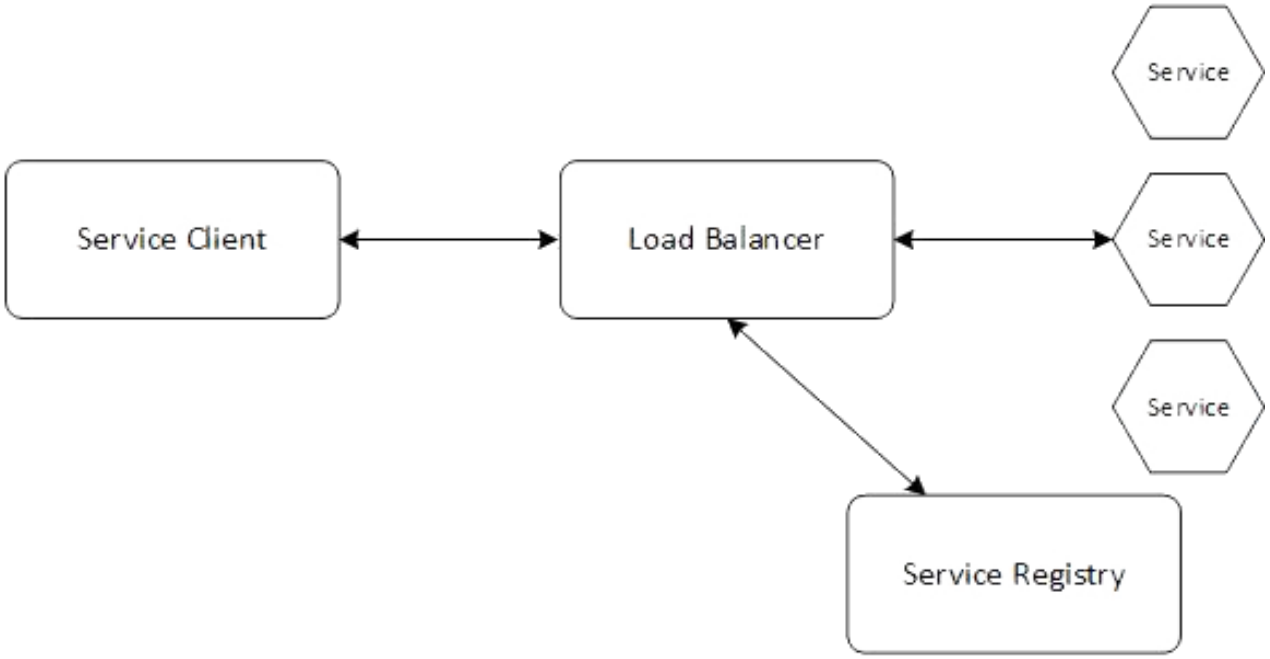

Рисунок 4.2 – Server-side discovery паттерн

Service Registry збирає інформацію про кожен з сервісів, де саме він розгорнутий. Цю інформацію можуть використовувати інші сервіси, наприклад, gateway для того, щоб виконати запит на конкретний сервіс.

Gateway є єдиною точкою входу в систему і його основна задача – передавати запити реальним бекенд-сервісам. У такому випадку фронтенд-частині не треба знати, де саме розгорнуто той чи інший сервіс. Більш того, якщо піднято декілька екземплярів сервісу, Gateway зможе визначити, на який саме екземпляр перенаправити запит з точки зору балансування навантаження.

Балансування навантаження може відбуватися різними способами:

- запит може бути надісланий сервісу з найменшою кількістю активних з’єднань, така методика відома під назвою RoundRobbin;
- запити можуть бути розподілені на основі хешу IP-адреси клієнта, щодозволяє кожному клієнту залишатись на одному і тому ж мікросервісі протягом сесії;
- балансувальник може враховувати стан мікросервісів, такі як завантаження процесора, доступність пам'яті тощо, і приймає рішення про розподіл запитів на основі цих метрик.

Для розробки системи буде використано одну з готових бібліотек для балансування навантаження між сервісами.

В нашому випадку система буде підтримувати 2 типи комунікації: синхронну і асинхронну. Асинхронний тип комунікації буде використовуватися для направлення нотифікацій в різних ситуаціях клієнту, оскільки саме факт (час) відправлення нотифікації не є критично важливим.

Усі сервіси, окрім сервіса нотифікацій будуть комунікувати між собою за допомогою REST, тобто використовуючи HTTP протокол У цьому випадку кожен сервіс може використовувати HTTP методи для відправки запитів до інших сервісів та отримання відповідей. Способи використання методів наведено в табл. 4.1. Колонка «ідемпотентність» показує, чи є метод ідемпотентний, тобто чи змінюється стан сервера після виконання операції. Практичною цінністю розуміння ідемпотентності методів є впевненість у тому, що операції можуть бути безпечно повторюваними без виникнення додаткових змін або побічних ефектів.

Таблиця 4.1 – Основні HTTP-методи

Метод	Ідемпотентність	Опис
GET	так	Використовується для отримання ресурсу з сервера. Клієнт надсилає запит до сервера, і сервер повертає відповідь, яка містить вміст запитаного ресурсу.

Кінець таблиці 4.1

		GET-запити повинні бути безпечними та не міняти стан сервера.
POST	ні	Використовується для створення нового ресурсу на сервері або виконання деяких дій засобами сервера. Клієнт надсилає дані до сервера, які потім обробляються та зберігаються. POST-запити можуть змінювати стан сервера.

PUT	так	Використовується для повного оновлення (заміни) існуючого ресурсу на сервері. Клієнт надсилає нові дані для заміни вмісту ресурсу на сервері. Також для оновлення ресурсу (часткового) може бути використано метод PATCH, який не є ідемпотентним
DELETE	так	Використовується для видалення ресурсу на сервері. Клієнт надсилає запит на видалення конкретного ресурсу, і сервер видаляє цей ресурс. DELETE-запити повинні бути обережними і підтримуватися на сервері з використанням відповідних механізмів авторизації та підтвердження.

На іншій стороні такого роду комунікаційного зв’язку є сервер, що обслуговує користувача, надаючи клієнту відповідь на його запит. У відповіді міститься код статусу (табл. 4.2) - це число, яке складається з трьох цифр, що визначає загальний результат роботи з отриманим запитом, та може вказувати клієнту на необхідні подальші дії, наприклад зміни даних для коректної подальшої обробки запиту та виведення повідомлення про це користувачу.

Кожна з груп статус-кодів HTTP має багато різних кодів. Отже, кожен клас статус-кодів включає десятки конкретних кодів, які вказують на певні стани або події під час обміну даними між клієнтом і сервером. Наприклад, з групи успішних запитів можна виділити такі статус-коди: 200 OK, 201 Created, 204 No content.

Таблиця 4.2 – Коди статусів відповідей

Код	Статус	Опис

1**	Інформативні	Повідомляють клієнта про прогрес процесу запиту та про те, що додаткові дії можуть бути необхідні для завершення запиту.
2**	Успішні	Запит клієнта був успішно оброблений і виконаний.
3**	Переспрямування	Для завершення запиту клієнт повинен зробити додаткові дії, щоб завершити запит, такі як перенаправлення на іншу адресу.
4**	Клієнтська помилка	Повідомляють помилку або неправильний запит з боку клієнта.
5**	Помилка сервера	Вказують на помилку або недоступність сервера під час обробки запиту

Комунікація з сервісом нотифікацій буде відбуватися за допомогою черги повідомлень. Як було зазначено раніше, комунікація через чергу повідомлень базується на асинхронному обміні повідомленнями між компонентами системи. Така черга є посередником, який отримує повідомлення від відправника і доставляє його отримувачеві. Це дозволяє забезпечити розсилку повідомлень між різними компонентами системи, незалежно від їх доступності та стану. Сервіси в нашому випадку буде надсилати повідомлення до такої системи черги повідомлень, а сервіс нотифікацій може підписатися на цю чергу та отримувати повідомлення. Такий підхід дозволяє асинхронну комунікацію між сервісами.

Основні типи комунікації включають P2P і Publish-Subscribe.

При P2P (Point-to-Point) типі комунікації повідомлення надсилаються від одного відправника до одного або кількох конкретних отримувачів. Кожне повідомлення має лише одного отримувача або групу отримувачів, які підписані на цей тип повідомлень. Це тип комунікації використовується, коли потрібно надіслати повідомлення конкретному одержувачу або групі одержувачів.

Publish-Subscribe тип комунікації полягає у тому, що повідомлення надсилаються від відправника, і вони розсилаються всім підписаним отримувачам,

які цікавляться цим типом повідомлень. Отримувачі можуть підписатися на певні категорії або теми повідомлень, і всі повідомлення, що належать до цих категорій або тем, будуть їм доставлені. Це дозволяє розподілити повідомлення між багатьма отримувачами, які цікавляться певною темою або категорією.

Для розроблюваної системи буде обрано тип Publish-Subscribe.

Загальну архітектурну схему мережі наведено в додатку А.

Програма має бути легко інтегровано з скриптами, що стосуються нейронної мережі, які написані на мові програмування Python. Слід зазначити, що тестування скриптів і сервісів має виконуватися, як окремо, так і в зв'язці.

Така реалізація має перевагу в невисокій зв'язності нейронних мереж та системи, яка взаємодіє безпосередньо з користувачем. Саме це може давати перевагу в використанні як окремо, так і разом, що дозволить легко вносити в зміну в обидві частини системи.

4.3 Вибір СКБД

Оскільки обрано мікросервісний тип архітектури, варто зазначити, що кожен з сервісів має мати свою окрему базу дану, обмежену мінімальною предметною областю, за яку відповідає сервіс. Загалом, основними вимогами до зберігання даних є зберігання інформації про користувачів системи і їх ролі, відповіді на питання і інформація про психологів і роботу з клієнтами.

На сьогоднішній день існують реляційні і нереляційні бази даних.

Найпоширенішим і домінуючим типом є реляційні бази даних. Вони зберігають всю інформацію у вигляді набору таблиць зі стовпчиками з заданими типами та рядками. Первинний ключ у таблицях допомагає унікально ідентифікувати кожен запис. Саме цей тип бази даних є найбільш популярним, оскільки є найбільш простим для користувачів, дозволяє досить лаконічно описувати основні операції, використовувати SQL (мову програмування для запитів, обробки та визначення даних, а також для контролю доступу). База даних такого типу має відповідати принципам ACID (атомарність, несуперечливість,

ізольованість та довговічність), що дозволяє забезпечити цілісність даних та зменшити вірогідність неочікуваної поведінки системи. Високий рівень цілісності досягається завдяки визначенню транзакцій (набору дій, що має виконатися повністю або не виконатися взагалі). Але може виникати проблема при паралельному виконанні транзакцій, яка може бути вирішена за допомогою застосування рівнів ізоляції транзакцій. Проте така модель має більш низьку швидкість доступу до даних та потребує більшого об'єму зовнішньої пам'яті. Дуже часто може з'явитися багато таблиць, в яких буде важко розібратися. Для пришвидшення деяких операцій можна використовувати індекси БД. Така БД буде прекрасним вибором для додатків, яким не потрібні будуть часті оновлення, масштабування та злиття даних. Найпопулярнішими СКБД є SQL Server, MySql, PostgreSql, Oracle, MariaDB тощо.

На другому місці по популярності є NoSql бази даних. Вони не використовують традиційну модель з таблицями та зв'язками, а використовують альтернативні способи зберігання та доступу до даних, що надає можливість зберігати неструктуровані або слабоструктуровані дані, а також маніпулювати ними. Такого типу БД не накладає жодних обмежень на типи даних. Окрім того в процесі розробки можна створювати нові типи даних у разі необхідності. NoSql бази даних краще масштабуються, легко підтримують операції реплікації - копіювання оновлених даних на інші сервери для забезпечення відмовостійкості та масштабованості системи. Щодо недоліків використання такого роду БД можна виділити обмежену ємність мови запитів у порівнянні з SQL, яка має довгу історію та безліч стандартів. Вона не потребує проектування моделі даних перед початком роботи, тому можна зіштовхнутися з певними труднощами в подальшій роботі. Найпопулярнішими СКБД є MongoDB, Redis, Cassandra, CouchDB тощо.

Після аналізу двох найпопулярніших типів зробимо висновок, що для поставленої задачі найкращим ріщенням буде вибір саме реляційної бази даних, оскільки дані мають чітку структуру та чітку залежність між ними, крім того не очікується частої зміни структури таблиць. Також з використанням реляційної БД

можна бути впевненими в транзакційній безпеці, а також існує багато механізмів щодо резервного копіювання даних, відновлення та обробки помилок.

Однією з проблем, з якими можна стикнутися при використання реляційної БД - це проблеми продуктивності при значному збільшення кількості записів в таблицях. Проте існують механізми для контролю та покращення продуктивності у разі потреби, коли неможливо зробити короткий період зберігання даних, такі як індексування, оптимізація запитів, партиціонування, кешування, оптимізація схеми бази даних та зв'язків між об'єктами, денормалізація, реплікація тощо.

У якості системи керування БД обрано PostgreSql, оскільки вона дозволяє підтримувати БД необмеженого розміру, дозволяє використовувати потужні та надійні механізми транзакцій і реплікації, надає можливість легкого розширення.

4.4 Проєктування БД

Під аналізу функціональної вимоги, яка стосуються процесу керування користувачами було виділено 2 основні сутності: Користувач та Роль (користувач, психолог, адміністратор). Загалом, процес аутентифікації та авторизації у випадку з використанням мессенджер-боту відбувається наступним чином:

- авторизація в системі месенджера;
- отримання дозволу - месенджер повертає вам деяку інформацію про користувача, наприклад, унікальний ідентифікатор користувача. Ви можете використовувати ці дані для подальших кроків, таких як авторизація;
- створення внутрішньо-системного облікового запису у разі у разі потреби;
- авторизація - надання доступу до певного функціонала.

Визначено наступні базові сутності та їх атрибути (табл.4.3)

Таблиця 4.3 - Базові сутності і атрибути для БД користувачів

Сутність	Атрибути
Користувач	Ідентифікатор, ідентифікатор в системі месенджера, ім'я, роль

Роль	Ідентифікатор, назва

Кожен користувач може мати декілька привілегій, особливо важливим це може бути при розширенні системи. Отже, відношення між двома сутностями має бути “один-до-багатьох”, що відповідно до правил нормалізації БД, вирішується через додаткову таблицю. Схема БД для сервіса користувачів представлена в додатку А.

Процес відповідей на питання передбачає зберігання повного набору відповідей для відслідковування історії та для надання інформації спеціалісту. Кожне з питань опитувальника має відповідну числову вагу для формування датасету, описаного в розділі 3. Базові сутності і їх атрибути наведено в табл. 4.4.

Таблиця 4.4 - Базові сутності і атрибути для БД опитувальників

Сутність	Атрибути
Опитувальник	Ідентифікатор, автор, назва
Питання	Ідентифікатор, текст, опитувальник
Варіанти відповідей	Ідентифікатор, питання, числова вага відповіді
Відповідь	Ідентифікатор, користувач, питання, відповідь, версія проходження (group id)

Оскільки кожен користувач має можливість періодично декілька разів проходити опитувальники з метою трекінгу зміни свого стану. Тому додано атрибут answer_group_id, який є однаковим для всіх питань, на які були надано відпвіді в рамках одного проходження опитувальника. За допомогою цього ідентифікатора можна зберігати інформацію щодо отриманої оцінки психологічного стану особи. Схема БД для сервіса опитувальників представлена в додатку Б.

Для керування даними про психологів і їх взаємодією з клієнтами необхідно зберігати наступну інформацію (табл. 4.5).

Таблиця 4.5 - Базові сутності і атрибути для БД психологів

Сутність	Атрибути
Психолог	Ідентифікатор,ідентифікатор користувача в системі, ім’я, ціна однієї консультації, рейтингова оцінка
Спеціалізація	Ідентифікатор, назва, інформація-опис
Клієнт	Ідентифікатор, ідентифікатор користувача в системі, ім’я
Консультація	Ідентифікатор, психолог, клієнт, дата і час, статус

В даній частини загальної системи інформація про ім’я користувачів (психолога чи клієнта) є надлишковою, оскільки може бути отримана шляхом запиту в інший сервіс. Оскільки 1 психолог може спеціалізуватися на декількох різних сферах, то зв’язок між сутностями Психолог і Спеціалізація - “багато-до-багатьох”. Відповідно до третьої нормальної форми подібного роду зв’язок вирішується через окрему таблицю.Схема БД для сервіса психологів представлена в додатку А.

4.5Архітектура на рівні сервісу

На рівні окремого сервісу структура компонентів може будуватися з використанням різних архітектурних шаблонів. Серед популярних шаблонів можна виділити: Layered Architecture (архітектура з рівнями), Component-Based Architecture (архітектура на основі компонентів), Domain-Driven Design (DDD), Hexagonal Architecture.

Суть паттерна Layered Architecture (архітектура з рівнями). полягає у тому, що сервіс розбивається на окремі рівні абстракції, такі як presentation, service layer, data access layer та інші. Кожен рівень має свою відповідальність і обробляє конкретні завдання.

При використанні Component-Based Architecture сервіс розбивається на набір компонентів, які виконують конкретні функції. Кожен компонент - це незалежний модуль який може бути перевикористаний у різних контекстах. Це надає можливість легко підтримувати та розширювати систему, замінюючи або додаючи компоненти у разі потреби.

Domain-Driven Design (DDD) зорієнтований на домен. Цей підхід спрямований на моделювання програми навколо доменних областей.. Головною ідеєю цього підходу є розуміння бізнес-домену та відображення цього розуміння в програмній моделі з відповідним розділенням компонентів.

Гексагональна архітектура, також відома як Ports and Adapters або Clean Architecture, базується на принципі розділення основної логіки додатку від залежностей та інфраструктури. Вона використовує інтерфейси (порти) для забезпечення комунікації з зовнішніми компонентами (адаптерами). Це дозволяє замінювати адаптери для зовнішніх ресурсів без впливу на основну логіку додатку.

Для проєктування сервісної архітектури було обрано Layered Architecture (рис. 4.3)

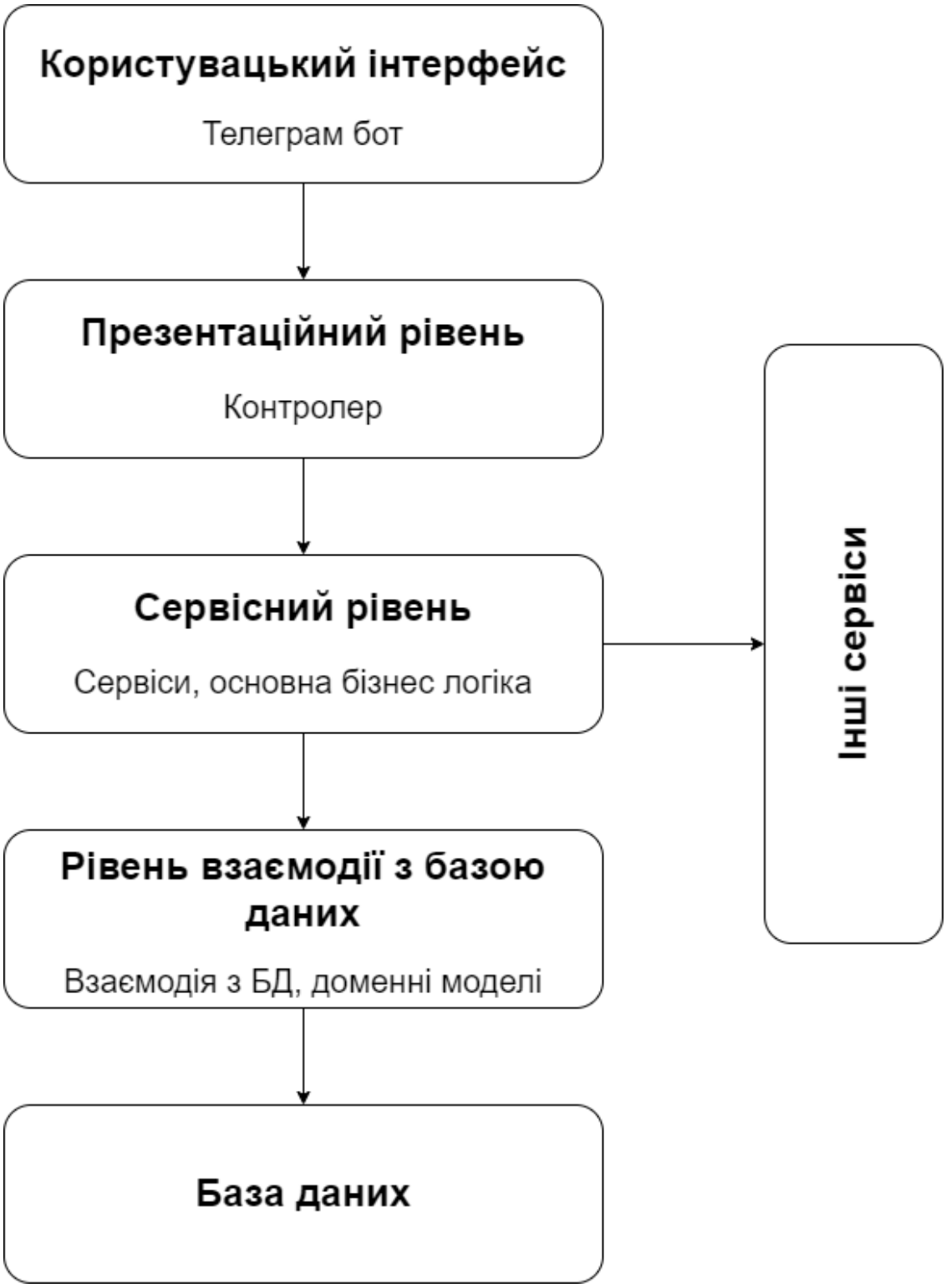

Рисунок 4.3 - Архітектурна схема сервісного рівня

З точки зору сервісу має бути чітка структура пакетів і відповідати рівням зазначеним вище. Отже, сервіси мають мати рівні сервісу (Service), контролера (Controller) та домена (Domain) і використовуватися для розділення

відповідальностей та забезпечення структурованості програми. Кожен з цих рівнів має свою функціональну роль у додатку.

Рівень домена включає класи, які представляють бізнес-логіку та об'єкти додатку. Це можуть бути класи, що описують сутності домену, такі як користувачі, продукти, замовлення і т. д. Рівень домена містить методи, які виконують операції з цими об'єктами, такі як збереження, видалення, оновлення тощо. Цей рівень забезпечує уніфікований доступ до даних і визначає правила бізнес-логіки.

Рівень сервісу містить бізнес-логіку, яка не належить безпосередньо до доменних об'єктів. Він містить сервіси або компоненти, які забезпечують виконання специфічних операцій над доменними об'єктами. Наприклад, в сервісному рівні можуть бути реалізовані методи для обробки бізнес-правил, координації декількох доменних об'єктів, виконання складних операцій тощо. Цей рівень відповідає за керування виконанням бізнес-логіки та взаємодію з рівнем домена

Рівень контролера (Controller) обробляє вхідні HTTP-запити, виконує необхідні дії для підготовки даних та відповідає на запити результатами. Він відповідає за маршрутизацію запитів і взаємодію з рівнем сервісу для отримання необхідних даних або виконання певних операцій. Результатом роботи контролера може бути відправка відповіді клієнту, перенаправлення на іншу сторінку або виклик іншого контролера.

Рівень представлення (View): Варто згадати, що додатки на основі Java, зазвичай, також включають рівень представлення (View), який відповідає за відображення даних інтерфейсу користувача. Цей рівень займається візуалізацією інформації та взаємодіє з користувачем.

Висновки до розділу

В розділі було описано вибір технічних засобів для реалізації клієнтської частини та бекенд-сервісів. Було обрано використовувати месенджер бот, як точку входу в систему, було обрано Telegram в якості месенджера для базової розробки.

Було обрано клієнт-серверну архітектуру, виконано огляд використання мікросервісного і монолітного підходів. Для вирішення поставленої проблеми було обрано використати мікосервісний тип, виділено основні сервіси: ті, що безпосередньо стосуються предметної області і службові. Завдяки такому типу архітектури можна ефективного розподіляти навантаження між компонентами системи і масштабувати окремі частини в залежності від задачі.

Обрано систему керування базами даних PostgreSql. Виділені базові сутності і атрибути БД, а також створено схеми для кожного сервіса відповідно.

5 СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Відповідно до визначеної архітектури системи має бути розроблена система для оцінки психологічного стану людини, що базується на кдієнт-серверні мікросервісній архітектурі. Для того переш за все мають бути визначені технічні засоби для реалізації для кожного сервісу.

Також мають бути визначені підходи та інструменти для тестування, щоб переконатися, що система функціонує коректно та відповідає початковим вимогам.

Як результат розробки системи, користувачам має бути надано інструкції, як користуватися системою.

5.1 Вибір технічного стеку

Для реалізації бекенд-сервісів буде використано мову програмування Java. Java – це популярна мова програмування, яка була розроблена компанією SunMicrosystems. Серед властивостей і перевагцієї мови можна виділити:

- об’єктно-орієнтованість;
- простота;
- безпека;
- надійність (виявлення помилок на ранніх стадіях, динамічна перевірка виключних ситуацій, що можуть призвести до помилок)
- кросплатформеність;
- багатопотоковість;
- динамічність (можливість додавання нових методів до бібліотек, не завдаючи шкоди);
- підтримка раніше написаного коду;
- garbage collector;

Для розробки веб-застосунків на мові програмування Java використовується Java EE, а саме Java Servlets, які використовуються для обробки запитів від клієнтів та генерації відповідей. Проте зараз більш широко використовується Spring

Framework, який розширює можливості Servlet API. Його великою перевагою є керування залежностями (їх впровадженням) та інверсія управління, яка дозволяє писати незалежні один від одного компоненти. Spring надає велику кількість різних модулей, наприклад, Core, Security, Cloud, Data, MVC.

Модуль Spring Core є основою всього фреймворку Spring. Він надає базові функціональність та основні концепції, такі як інверсія управління (Inversion of Control, IoC) та впровадження залежностей (Dependency Injection, DI). Цей модуль дозволяє ефективно управляти компонентами додатку та їх залежностями.

Модуль Spring Security надає механізми для забезпечення безпеки в додатках, зокрема автентифікацію, авторизацію, захист від атак та керування сесіями. Він дозволяє розробникам захистити ресурси та контролювати доступ до них, забезпечуючи безпеку веб-застосунків.

Модуль Spring Cloud надає набір інструментів та бібліотек для розробки розподілених та мікросервісних додатків в середовищі хмари. Він надає рішення для керування конфігурацією, збереження та відстеження журналів, обнаруження служб, моніторингу та іншого, що допомагає побудувати масштабовані та надійні хмарні системи.

Модуль Spring Data спрощує доступ до баз даних і роботу з ними. Він надає абстракції для взаємодії з різними типами баз даних, такими як SQL- та NoSQL-бази даних.

Модуль Spring MVC надає структуру та організацію для розробки веб-застосунків. Він реалізує паттерн проектування Model-View-Controller, де модель представляє дані, представлення відображає інтерфейс користувача, а контролер обробляє запити та керує потоком даних між моделлю та представленням. Spring MVC забезпечує обробку маршрутів, валідацію даних, керування сесіями та інші функції для розробки веб-додатків [20].

Буде використано SpringBoot, який спростить налаштування проекту та дозволить використовувати стартери (спеціальні пакети, які містять набір сконфігурованих залежностей для різних задач, наприклад, для моніторингу програми, тестування тощо). SpringBoot дозволить швидко розпочати роботу з

проєктом без необхідності вручну налаштовувати багато деталей. Також Spring Boot надає підтримку вбудованих серверів, таких як, наприклад, Tomcat, що полегшує розгортання системи.

Оскільки для розроблюваної системи було обрано реляційні бази даних, то буде використано ORM-систему Hibernate. ORM-система – це така технологія, яка дозволяє зв’язувати об’єкти БД з програмними об’єктами, концепцією ООП, створюючи віртуальну об’єктну БД. Завдяки цьому вийде скоротити код, оскільки основні запити до БД будуть представлені, як методи ORM-системи, що автоматично будуть представлені у вигляді нативних запитів відповідно до встановленого драйвера БД. Подібного роду системи є корисними у ситуаціях, коли важливо одночасно підтримувати декілька різних систем керування базою даних, що може значно збільшити кількість коду. Hibernate є інтегрованим з JPA специфікацією. Завдяки інтеграції з JPA, Hibernate надає розробникам можливість використовувати стандартні анотації та інтерфейси JPA для визначення маппінгу об'єктів на таблиці, використання критеріїв запитів і управління транзакціями. Це дозволяє зберігати код незалежним від конкретної реалізації ORM і спрощує перехід між різними ORM-фреймворками, якщо виникає така потреба.

Як інструмент для підтримки версіонування і керуванням процесами міграцій БД буде використано Liquibase. Liquibase підтримує різні бази даних, включаючи MySQL, PostgreSQL, Oracle, Microsoft SQL Server та інші. Ви можете використовувати одну і ту ж набір міграційних файлів для різних баз даних, забезпечуючи переносимість між ними. Liquibase використовує простий та зрозумілий синтаксис для визначення міграцій та змін схеми. Ви можете використовувати XML, YAML або SQL для опису необхідних змін, забезпечуючи зручність розробки та читабельність коду.

Для управління проектом буде використано Maven. Цей інструмент візьме на себе функції завантаження додаткових бібліотек, компіляцію, створення .jar, та генерації документів. Maven має багатий набір плагінів, які дозволяють розширити функціональність та налаштувати різні аспекти розробки. Наприклад, плагіни

можуть виконувати статичний аналіз коду, генерувати документацію, запускати тестування,

Для вирішення питання визначення місця знаходження екземпляру сервісу можна використати шаблон мікросервісної архітектури Server-Side discovery

Для реалізації мікросервісної архітектури буде використано Netflix стек. Netflix стек являє собою набір технологій та інструментів для розробки та розгортання масштабних мікросервісних застосунків. Цей стек був створений для вирішення складних завдань, пов'язаних з масштабуванням, стійкістю та ефективністю великих обсягів даних.

В якості Discovery Server буде використано реалізацію Eureka. Його задача - допомогти знайти потрібний сервіс. Коли якомусь сервісу потрібно отримати доступ до іншого, Discovery Sever надає всі деталі кінцевої точки запитуваного сервіса для встановлення з’єднання. Discovery Sever виступає в якості реєстру сервісів (Registry). Всім сервісам служб необхідно зареєструватися в Discovery Sever, інакше Discovery Sever не знає про цю послугу.

Зовнішні користувачі системи отримують доступ до наших основних служб через API Gateway. Відповідно до Netflix стеку буде використано Netflix Zuul API Gateway. Zuul може використовуватися як проксі та як фільтр запитів. Причина в тому, що коли запит надходить до Zuul, він збирається отримати доступ до певної основної служби за допомогою service id. Також до можливостей Zuul входить load-balancing, який може оптимально розподіляти навантаження між кількома екземплярами сервісів. Одним з алгоритмів, на якому базується ця реалізаціє є RoundRobbin. До можливостей Zuul також можна віднести фільтрацію і моніторинг. Основні типи фільтрів включають в себе:

- передфільтри – виконуються перед тим, як запит буде направлено на мікросервіс, можуть відстежувати та журналювати запити, а також збирати статистику про навантаження та продуктивність системи;
- фільтри маршрутизації – визначення, до якого мікросервісу буде направлено запит, балансування навантаження;

- післяфільтри – можуть фільтрувати або змінити відповідь перед поверненням її клієнту, а також збирати статистику про відповіді.

В якості черги повідомлень буде використано ActiveMQ. ActiveMQ забезпечує механізми для асинхронної комунікації між розподіленими додатками. Вона дозволяє вирішувати проблеми, пов'язані з надійністю та швидкістю доставки повідомлень, забезпечуючи гарантії доставки, очереді повідомлень та підписки на події [21].

В якості середовища розробки буде використано IntellijIdea. Воно розроблене компанією JetBrains і є одним з найпопулярніших інструментів для розробки в середовищі Java. IntelliJIDEA надає багато функціональних можливостей та інструментів, які полегшують розробку програмного забезпечення і підвищують продуктивність розробників. Це середовище розробки дозволяє використовувати більшість сучасних інструментів для застосунків, може інтегруватися з іншими інструментами, надає можливість ефективної відладки програми.

В якості системи контролю версій, буде використано Git. Її буде використано для відслідковування змін в файловій системі під час розробки програмного забезпечення.

5.2 Тестування

Процес тестування веб-застосунків включає в себе кілька етапів, які допомагають перевірити функціональність, надійність, безпеку та продуктивність системи. Створення тестових сценанаріїв відбувається для різних сценаріїв та різних ситуацій поведінки системи. Тестові сценарії описують сценарії тестування, які включають вхідні дані, очікувані результати та критерії прийняття.

Існують наступні види тестування:

- функціональне тестування - перевірка на відповідність система початковим вимогам;

- нефункціональне тестування - перевірка на відповідність нефункціональним вимогам, тобто включає в себе перевірку надійності системи, безпеки, продуктивності, навантаження системи.

Також під час роботи над розробкою системи має виконуватися тестування змін - перевірка після внесення певних змін в програму. Відповідно для перевірки системи мають бути наявні функціональні і нефункціональні тести.

Функціональне тестування може бути використано для перевірки різних частин системи:

- unit testing (модульне тестування) - перевірка окремих компонентів (класів або методів) веб-застосунку, щоб переконатися, що вони працюють правильно і відповідають вимогам;
- integration Testing (інтеграційне тестування) - Цей тип тестування перевіряє взаємодію між різними компонентами системи, щоб переконатися, що вони правильно взаємодіють один з одним;
- system testing - це тестування програмного забезпечення, що виконується на повній зібраній системі, яка є зінтегрованою з усіма компонентами, з метою перевірки відповідності розроблюваної системи вихідним вимогам;
- acceptance testing - це останній етап тестування, що виконується перед релізом програми, зазвичай таку версію називають бета-версією.

Нефункціональне тестування включає в себе тестування навантаження (Performance і Load Testing).

Для модульного тестування на Java з використанням фреймворку Spring буде використано:

- JUnit, який надає широкий спектр аннотацій та методів для написання тестових сценаріїв, перевірки очікуваних результатів та роботи з залежностями;
- Mockito - тестовий фреймворк для створення несправжніх (mock) об'єктів та проведення імітації поведінки певних залежностей під час модульного тестування, він дозволяє легко створювати та налаштовувати мок-об'єкти для заміни реальних залежностей, наприклад, бази даних чи іншого сервіса;

- Spring Test - модуль для тестування Spring програм, основна його задача - спростити налаштування і виконання тестів до програми, створеної за допомогою Spring;
- AssertJ - бібліотека для виразних перевірок в тестах на відповідність параметрів очікуваним результатам, яка надає велику кількість методів перевірки, які спрощують процес створення перевірок результату виконання методу чи частини програми загалом.

Для інтеграційного тестування буде використано наступні інструменти:

- Spring Boot Test, що може використовуватися не лише для модульного, а і для інтеграційного тестування; ця бібліотека спрощує процес налаштування тестових контекстів та взаємодії з реальними залежностями;
- Testcontainers - бібліотека, яка дозволяє легко створювати та управляти контейнерами Docker, розгортати систему в контейнері під час тестування, а також дозволяє запускати контейнери з базами даних, іншими сервісами та залежностями, що використовуються системою, для реалістичного інтеграційного тестування;
- REST Assured - бібліотека, спеціально розроблена для тестування RESTful API, що надає зручний DSL для написання тестів, що перевіряють відповіді API, перевіряють відповіді сервісів: коди статусу, тіла відповідей та інші аспекти;
- Postman - інструмент для тестування розробниками API, який має можливості створення запитів та генерації подальших звітів.

Для системного тестування та acceptance testing було використано Cucumber Studio, інструмент, що дозволяє описувати функціональні вимоги у зрозумілій для бізнесу мові, а потім автоматично перевіряти, чи задовольняє ваша система ці вимоги. Cucumber є корисним для написання тестів, які перевіряють сценарії взаємодії з користувачем.

Для тестування навантаження було використано JMeter. Він має вбудовані можливості для створення сценаріїв тестування, налаштування навантаження на сервер та аналізу результатів. JMeter дозволяє симулювати навантаження з багатьох користувачів, одночасно взаємодіючи з веб-застосунком, щоб перевірити

його продуктивність та стабільність. Також було використано Grafana для візуалізації та моніторингу даних. Його часто використовують для створення графіків, діаграм, панелей та інших візуальних елементів, що відображають дані з різних джерел.

Програма має коректно надавати оцінку психологічного стану людини, яка проходить опитування. Перед проходженням опитування бот має запитати, чи готова людина пройти опитування, у разі позитивної відповіді - перейти до питань. В якості відповіді на питання має бути прийнята цифра з номером відповіді і перехід до наступного питання. Якщо введено некоректні дані, то користувач має отримати відповідне повідомлення про помилку.

5.3Інструкція користувача

Оскільки дана система працює на базі телеграм-боту, отже для користування програмою необхідно виконати кроки, описані нижче.

Крок 1. Для користування системою необхідно завантажити та встановити мобільний застосунок Telegram на смартфон або відкрити веб-версію програми на комп'ютері чи іншому пристрої.

Крок 2. Зареєструйте або увійдіть в свій Telegram-акаунт.

Крок 3. У полі пошуку Telegram введіть назву чат-бота "@psycho_moninor _bot". Буде відображено список результатів пошуку.

Крок 4. Оберіть потрібний чат-бот зі списку результатів. Натисніть на нього, щоб відкрити його профіль.

Крок 5. Напишіть “/start", щоб почати роботу з ботом.

Крок 6. Взаємодійте з чат-ботом, використовуючи текстові повідомлення, кнопки або інші доступні функції, які надає чат-бот.

Крок 7. При першому використанні чат боту вас буде зареєстровано у внутрішній системі.

Типовий приклад комунікації з системою наведено на рис. 5.1. Процес проходження опитування наведено на рис.5.2.

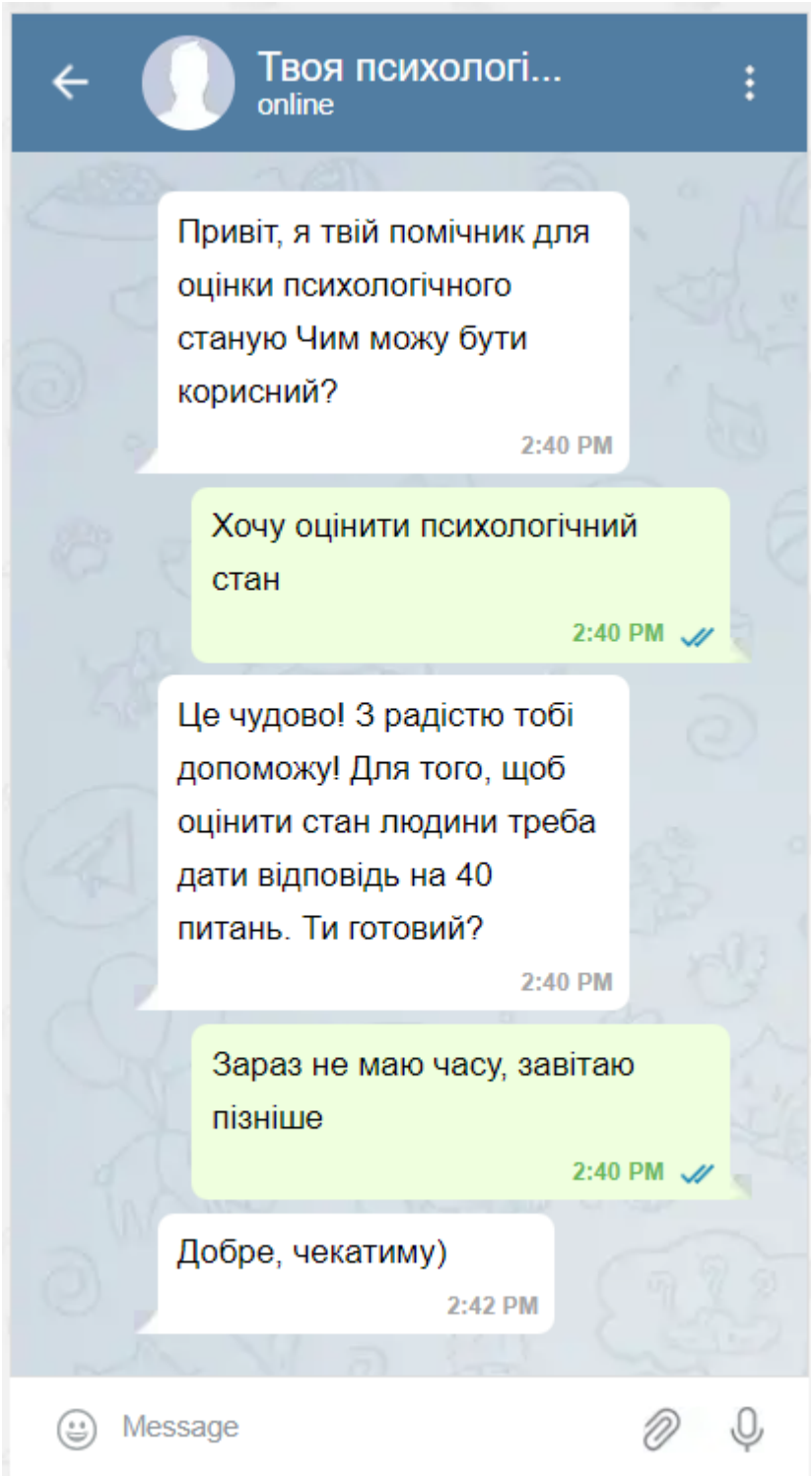

Рисунок 5.1 - Приклад комунікації користувача з системою

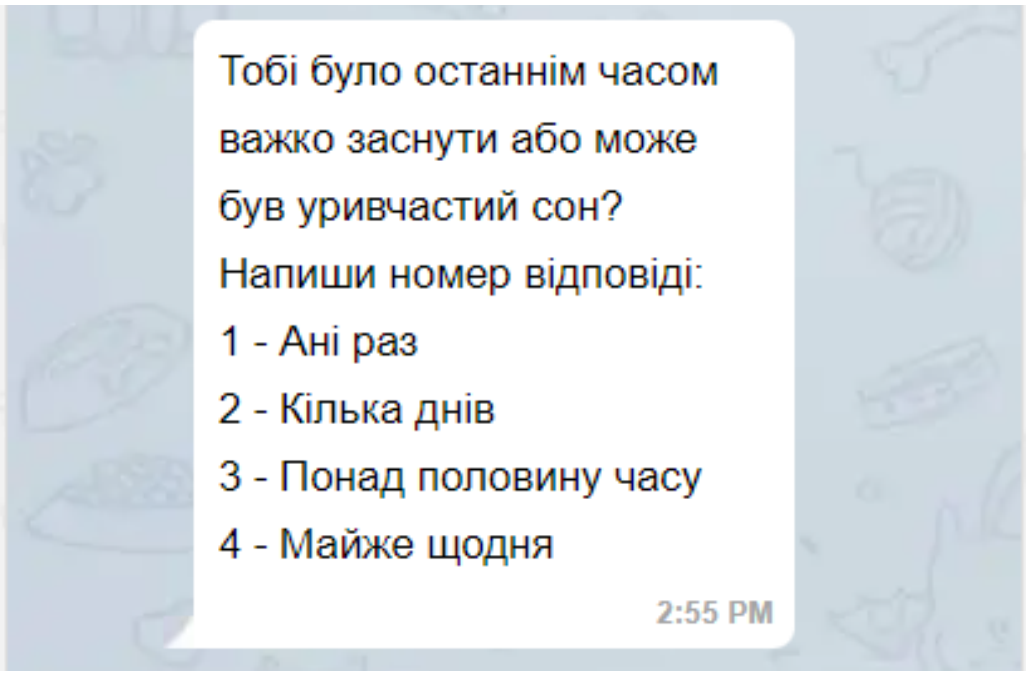

Рисунок 5.2 - Приклад питання з опитувальника

У разі введення некоректних даних в систему користувач отримає відповідне повідомлення про помилку і система буде чекати на ввід корректних даних (рис 5.3).

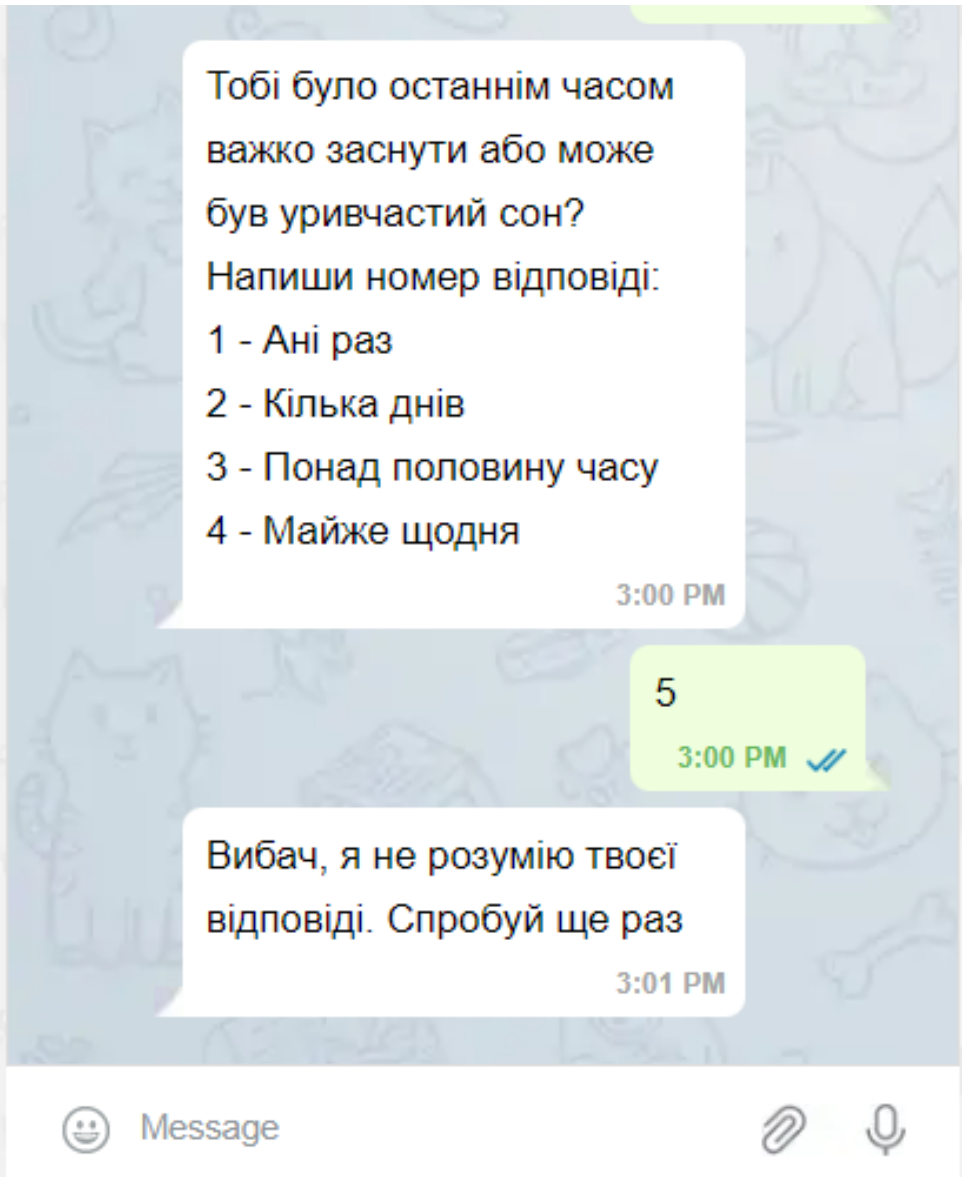

Рисунок 5.3 - Приклад введення некоректних даних

Приклад оцінки психологічного стану людини наведено на рис.5.4-5.5

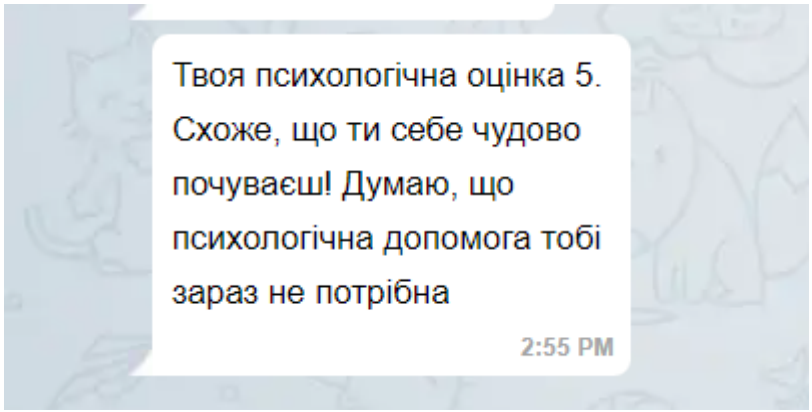

Рисунок 5.4 - Приклад отримання оцінки 5

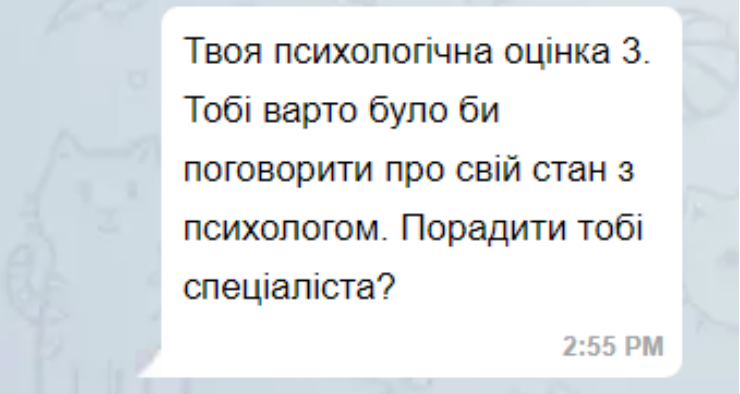

Рисунок 5.5 - Приклад отримання немаксимальної оцінки

Висновки до розділу

Розроблено систему для оцінки психологічного стану людини на мові програмування Java з використанням фреймворка Spring. Фронтенд-частина виконана у форматі телеграм-бота.

Було проведено функціональне (модульне та інтеграційне) тестування в рамках кожного сервіса за допомогою JUnit, AsertJ, Hamcrest, Mockito та SpringTest. За допомогою JMeter та Perfana було перевірено стійкість системи до навантаження та продуктивність відпрацювання операцій. Також було виконано тестування мануально з введенням різних коректних та некоректних значень.

Було створено інструкцію користувача з кроками щодо користування системою, а також наведено приклади реакції програми на виключні ситуації.

ВИСНОВКИ

В ході роботи над магістерською дисертацією було проаналізовано сферу автоматизації надання психологічної підтримки та виконано наступні завдання:

- огляд існуючих систем та зазначено пункти щодо удосконалення;
- аналіз інструментів для оцінки психологічного стану людини, а саме для задачі багатокласової класифікації;
- аналіз типів нейронів мереж для задач класифікації, в результаті було обрано мережу прямого поширення: MLP;
- описано процес побудови нейронної мережі для оцінки психологічного стану людини у вигляді оцінок за 5-бальною шкалою;
- побудова MLP моделей: базовою, з ReLu активацією та Unet-like , було обрано функцію втрат CrossEntropyLoss та алгоритм оптимізації Adam;
- проведено оцінки точності роботи моделей;
- наведено показники точності для моделей при спробі зміни кількості початкових питань;
- встановлено, що для вирішення поставленої задачі найкращим чином проявила себе Unet-like модель і показала точність оцінки близько 90%, 91% і більше на класах, представники яких доволі часто зустрічаються у вибірках, і близько 50% на зонах малої густини, а після початкової нормалізації вхідних даних шляхом ділення даних на максимальне значення вдалося піднят точність до 97% на зонах високої густини;
- побудовано загальну архітектуру системи, було обрано клієнт-серверну, мікросервісну архітектуру, визначено основні сервіси: користувачів, психологів, опитувальників, визначено методи міжсервісної комунцікації (REST та Messaging Queue);
- побудовано схеми БД до кожного з виділених сервісів, сервіс нотифікацій н має бази даних, оскільки не зберігає інформацію, а лише надсилаю нотифікацію користувачу;

- описано Layer-architecture підхід до розробки кожного з сервісів;
- систему було розроблено на мові програмування Java з використанням Spring фреймворка, а також дотичних технологій;
- виконано тестування системи;
- створено інструкцію користувача.

Загалом, розроблена система може бути корисна спеціалістам у сфері психології для підтримки в оцінюванні психологічного стану людини. Дана система показує доволі високий рівень ефективності при оцінюванні дорослих людей.

ПЕРЕЛІК ПОСИЛАНЬ

1. Tariq Rashid. Make Your Own Neural Network. First edition — CreateSpace Independent Publishing Platform, 2016. 98p.

2. Теслюк В. М. Вибір оптимального типу штучної нейронної мережі для автоматизованих систем "розумного" будинку / В. М. Теслюк, А. Г. Казарян. // Науковий вісник НЛТУ України. – 2020. – №30. – С. 90–93.

3. Aggarwal C. C. Neural Networks and Deep Learning: A Textbook first edition / Charu Aggarwal., 2018. – 284 с.

4. Waqar Ahmed Khan, S.H. Chung, Muhammad Usman Awan, Xin Wen. Machine learning facilitated business intelligence (Part I): Neural networks learning algorithms and applications. — 2020. — №1. 164-195.

5. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 3rd Edition / Aurélien Géron., 2022. 56с.

6. Matthew Botvinick, Sam Ritter, Jane X. Wang, Zeb Kurth-Nelson, Charles Blundell, Demis Hassabis. Reinforcement Learning, Fast and Slow. // CellPress. – 2019. – №5. – С. 408–422.

7. В. П. Марценюк, докт. техн. наук, проф., Н. В. Мілян.ОГЛЯД МЕТОДІВ ОПТИМІЗАЦІЇ В МАШИННОМУ НАВЧАННІ: ГРАДІЄНТНИЙ СПУСК ТА СТОХАСТИЧНИЙ ГРАДІЄНТНИЙ СПУСК. // Тернопільський національний технічний університет імені Івана Пулюя, Україна. – 2020.

8. Ruth Vang-mata. Multilayer Perceptrons: Theory and Applications. 30p.

9. Packt Publishing. Practical Convolutional Neural Networks: Implement advanced deep learning models using Python / Packt Publishing., 2018. 113p.

10. Fathi M. Salem. Recurrent Neural Networks: From Simple to Gated Architectures / Fathi M. Salem.. – 2022 с. – (1). 33p.

11. Chun Chet Tan. AUTOENCODER NEURAL NETWORKS: A Performance Study Based on Image Reconstruction, Recognition and Compression / Chun Chet Tan., 2009. – 88 с.

12. Корнага Я.І., Крилов Є.В., Сергієнко П.А. ОЦІНКА ПСИХОЛОГІЧНОГО СТАНУ ЛЮДИНИ З ВИКОРИСТАННЯМ MLP НЕЙРОННИХ МЕРЕЖ. // Телекомунікаційні та інформаційні технології. – 2023. – С. 20–26. / Я. І. Корнага, Є. В. Крилов, П. А. Сергієнко. // Телекомунікаційні та інформаційні технології. – 2023. – №2. – С. 20–26.

13. Guo, H., Guo, C., Xu, B. et al. MLP neural network-based regional logistics demand prediction. Neural Comput & Applic 33, 3939–3952 (2021).

14. Arun Kumar Dubey. Comparative Study of Convolution Neural Network's Relu and Leaky-Relu Activation Functions / Arun Kumar Dubey, Vanita Jain. // Applications of Computing, Automation and Wireless Systems in Electrical Engineering. Lecture Notes in Electrical Engineering. – 2019. – №553. – С. 96.

15. Bala Priya C. Cross-Entropy Loss: Make Predictions with Confidence [Електронний ресурс] / Bala Priya C – Режим доступу до ресурсу: https://www.pinecone.io/learn/cross-entropy-loss/.

16. Ricardo Llugsi, Samira El Yacoubi, Allyx Fontaine, Pablo Lupera. Comparison between Adam, AdaMax and Adam W optimizers to implement a Weather Forecast based on Neural Networks for the Andean city of Quito / Ricardo Llugsi, Samira El Yacoubi, Allyx Fontaine, Pablo Lupera. // IEEE Fifth Ecuador Technical Chapters Meeting (ETCM). – 2021.

17. Mark Lutz. Learning Python / Mark Lutz., 2013. – 67 с.

18. G. J. Kamani , R. S. Parmar, Y. R. Ghodasara. DATA NORMALIZATION IN DATA MINING USING GRAPHICAL USER INTERFACE: A PRE-PROCESSING STAGE / G. J. Kamani , R. S. Parmar, Y. R. Ghodasara. // Gujarat Journal of Extension Education. – 2019. – №2. – С. 106–109.

19. John Carnell. Spring Microservices in Action / John Carnell, Illary Huaylupo Sánchez., 2021. – 44 с.

20. Craig Walls. Spring in Action, Sixth Edition / Craig Walls., 2022. – 113 с.

21. Bruce Snyder. ActiveMQ in Action / Bruce Snyder, Dejan Bosanac, Rob Davies., 2011. – 59 с.