МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Секція інформаційно-комунікаційних технологій

ВИПУСКНА РОБОТА

на тему:

«Телеграм бот ассистент психолог»

Завідувач

випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Коробов А.Г.**

Студентка групи ІН-71 **Сухоставець М.В.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ

до випускної роботи

Студентки четвертого курсу, групи ІН-71 спеціальності "Комп'ютерні науки" денної форми навчання Сухоставець Марини Віталіївни.

Тема: "**Телеграм бот асистент психолог**"

Затверджена наказом по СумДУ

№_____ от _____2021 р.

Зміст пояснювальної записки: 1) аналіз проблеми та постановка задачі, 2)технології проектування інформаційних систем, 3) реалізація інформаційної системи – бота асистента психолога, 4) перевірка правильності функціонування системи

Дата видачі завдання "_____" _____2021р.

Керівник випускної роботи _____ Коробов А.Г.

Завдання прийняв до виконання _____ Сухоставець М.В.

РЕФЕРАТ

Записка: 71 стор., 16 рис., 1 додаток, 12 джерел.

Об’єкт дослідження — застосування Telegram API та моделі BERT для аналізу природньої мови.

Мета роботи — розробка бота асистента з використанням Telegram API та імплементація в нього двунаправленого кодувального представлення з трансформерів - BERT.

Методи дослідження — розробка алгоритма, опрацювання даних, розробка функціонала бота.

Результати — розроблено інформаційну систему, функціонал якої допомає вести контроль за психічним здоров’ям користувачам. Та як складова цієї системи була розроблена інтелектуальна модель оцінки емоційного стану людини.

Ключові слова: ТЕЛЕГРАМ БОТ, ПСИХІЧНЕ ЗДОРОВ’Я, ОБРОБКА ПРИРОДНОЇ МОВИ, РЕЛЯЦІЙНІ БАЗИ ДАНИХ, НЕЙРОНІ МЕРЕЖІ

ЗМІСТ

ВСТУП

Технології впливають на спосіб спілкування, навчання та мислення людей. Вони допомагають суспільству та визначає, як люди щодня взаємодіють між собою. Технології сьогодні відіграють важливу роль у суспільстві. Вони маюють позитивні та негативні наслідки для світу та впливають на повсякденне життя. Ми живемо в епоху, коли технологічний прогрес є загальним. Інтернет та мобільні телефони - ось кілька прикладів. Однак, завдяки технологічним досягненням, все це має і переваги і мінуси, для рішення яких, щодня розробляються нові технології.

Дана робота розгляне сферу ментального здоров’я сучасних людей, визначить проблематику,я ка пов’язани з нею сьогодні. Та запропонує інформаційне рішеня викладених проблем для поліпшення добробуту користувачів всі сучасних здобутків технологічного прогресу.

Буде розроблений телеграм бот асистент помічник для виявлення проблем з психологічним здоров’ям та з рекомендаціями щодо усунення цих негараздів. Будуть використанні сучасні технології розробки такі, як мова програмування Python та йдосі актуальна SQL для створення бази користувачів та управління їх даними. Буде інтегрований спосіб заробітку за рахунок рекомендаційного промошену спеціалістів психологів (реальних людей) в якості іще одного варіанта рішення дуже важких проблем користувачів. Буде виконано зв’язка з попередньорозробленою системою розпізнавань емоцій та демо версією бота.

1.АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сучасні технології підтримки психічного здоров’я

Психічне здоров’я має важливе значення для загального стану здоров’я та добробуту кожного, а психічні захворювання є загальними і піддаються лікуванню. Проте люди відчувають симптоми психічних захворювань по-різному - а деякі вживають потенційно небезпечну або ризиковану поведінку, щоб уникнути або приховати симптоми потенційної проблеми психічного здоров'я.

Так як наше життя майже як два десятиліття просоченно комп’ютерними технологіями і вони надалі так же експоненційно будуть розвиватися, то звичним буде запропонувати рішення проблем з ментальним здоров’ям у вигляді інформаційної системи, веб-додатку чи андроїд додатку.

Іноді люди, особливо молоді люди, які борються з проблемами психічного здоров'я, виробляють звички та поведінку, що збільшує ризик розвитку психічних захворювань. Або вони можуть посилити наявні психічні захворювання. Іноді ці звички та поведінка можуть бути ознаками проблем із психічним здоров’ям. Такі дії, як компульсивний секс, рекреаційне вживання наркотиків, нав’язливе вживання Інтернету, надмірні витрати або невпорядковані схеми фізичних вправ - все це може бути поведінкою, яка може порушити психічне здоров’я когось та потенційно вести його до шляху до кризи.[1]

Хоч сфера здорової психіки є дуже важливою, не всі люди звертаються до спеціалістів Існує також багато причин, за якими люди не ходять на психотерапію чи консультування, незважаючи на труднощі, яким це може допомогти. Наприклад думають, що це надто дорого, чи важжають, що не зможуть виділить достатньо часу на терапію чи навіть консультацію, не можуть знайти гарного спеціаліста, що є дійсно важкою проблемою, стереотипно думають, що лише люди з нездоровою психікою тобто лише

«божевільні» ходять до психотерапевтів та психіатрів; або елементранно люди не можуть знайти в собі сили це зробити самостійно, тобто вони настільки подавлені, або не розумібть що з ними, що і не можуть попросити про допомогу.

Також ті ж молоді люди є найбільшою та найактивнішою групою, яка вживає інтернет контент та користується всіми можливими засобами сучасних користувацьких додатків. Але ринок додатків стає все більше і більше: безліч соціальниж мереж (Instagram, Facebook, Twitter), ряд популярних месенджерів (Telegram, Viber, WhatsApp) та розважальних аплікейшинів (TickTock, YouTube) та ще багато іншого – сервіси Гугла, Яндекса чи розробника платформи, девайса; ігри, записники та додатки для менеджменту будь чого чи трекінгу. Більшість людей утопає в такій кількості додатків і не використовують або втрачають свій час із-за, або банально їм не вистачає місця (пам’яті) на мобільному пристрої чи комп’ютері. Тому варіант розробки андроїд додатку чи десктоп додатку є не самим актуальним. Веб-додаток є більш актуальним в цьому плані, адже користувач зможе мати доступ до ресурсу з будь-якого пристрою, в будь який час, за умови наявності інтернету. Але реалізація великої системи та адаптація її під мобільні, десктопні і навіть планшетні девайси є дуже дорогою, трудоємкою і затяжною.

Тому інеграція з якоюсь популярною інфрмаційною системою буде більш вигідним і більш актуальним рішенням. Київський міжнародний інститут соціології провів у лютому 2021 дослідження про використання месенджерів серед українців та прийшов висновку, що близько 73% громадян використовують додатки для обміну миттєвими повідомленнями. Тому вигідно буде інтегрувати нашу розробку в який небуть месенджер.

За версією сайта statista.com найбільш популярними месенджерами 2021 року є WhatsApp активність користувачів зіставляє 2 000 мільйонів користувачів за місяць, наступним іде Facebook Messenger з активністю 1 300

мыльйоныв, Weixin / WeChat (дуже популярний в Китаї) з 1 213 мільйоною кількістю користувачів в місяць, QQ – щомісячно 617 мільйонів користувачів (в основному серед тогож Китаю), Telegram щомісячно активними користувачами стають приблизно 500 мільйонів користувачів. А до найпопулярніших месенджер додатків серед українців за версією видання glavcom.ua можно выднести два перших світових (Facebook Messenger, WhatsApp) та той що подалі серед світових – Telegram. Розробка системи допомоги з психічним здоров'ям для Фейсбуку не є актуальним, адже він має веливу базу таких ресурсів, починаючи з відплвідних пабліків та користувачів, які є професіоналами в сфері ментального благополуччя (психотерапевтами та психологами), не кажучи вже про ігри-додатки та боти які вирішу.ть ту ж задачу. Месенджери WhatsApp та Telegram не мають такої великої внутрішньої конкуренції ресурсів про ментальне здоров'я. Але обраним для розробки платформи під був Telegram, адже має більшу базу знань, більше функціоналу та велике ком'юніті розробників.

Було обрано виконати розробку Телеграм бота асистента психолога, що буде допомагати користувачу підтримувати його психічне благополуччя та допомагати з контролем його стресового стану.

Чат боти - це своєрідні акаунти Telegram, якими керує не реальна людина, анаписаний програмний код - програмне забезпечення, зачасту таких ботів наділяють функціями штучного інтелекту.

Аналіз аналогів. В даній главі виконаємо аналіз аналогів додатків для спостереження за психічним здоров'ям та інших схожик аплікейшинів.

Facebook Bots. Woebot. X2AI (розробник боту) описує своїх ботів як терапевтичних помічників, тобто вони пропонують допомогу та підтримку, а не лікування. Здебільшого ці боти виконують допоміжну роль - більше інструменту, ніж терапії. Таким чином, Woebot відрізняється. Це оплачується як лікування самостійно, доступний варіант для тих, хто не має жодної

допомоги у боротьбі з психічним здоров’ям. Дарсі сприймає це як "терапію воротами", щоб дати людям хороший перший досвід і навіть допомогти зрозуміти, коли їм потрібна більш інтенсивна форма втручання.

Будучи єдиним терапевтичним чат-ботом з рецензованими клінічними даними для його резервного копіювання, Woebot відокремлюється від набору. Але використання цих результатів, щоб стверджувати, що це може значно зменшити депресію, може піддати Woebot юридичним зобов’язанням, яких ботам у допоміжних ролях вдалося уникнути. Без моральної свободи незалежний кодекс не може бути визнаний винним у будь-яких злочинних діях. Але якщо це заподіює шкоду, на нього можуть поширюватися цивільні закони, що регулюють відповідальність за товар. Більшість виробників справляються з цими ризиками, накладаючи на свою продукцію етикетки, що попереджають про можливу небезпеку; Woebot має дещо синонімічну відмову від відповідальності, яка говорить, що люди не повинні використовувати його як заміну для отримання допомоги.[3]

Переваги Woebot:

- В розробкі приймали участь спеціалісти
- Доступний на фейсбуці

Недоліки:

- Не безкоштовний
- Доступний лише на фейсбуці

Додатки Google PlayStore. Wysa: чат-бот для терапії стресів, сну та уважності. Вайса - ваш милий приятель, який мене підбадьорює, і трекер доброго самопочуття. Wysa наповнена щоденною духовною медитацією, яка покращує психічне здоров'я, а також є прекрасним способом поєднати сімейну медитацію. Він поміщається прямо в кишеню і допомагає вам залишатися емоційно здоровими за допомогою відстеження настрою, пошуку оптимізму,

переформулювання думок (ТГС) у дружніх чатах, щоб допомогти вам зменшити смуток.

Wysa - ваш друг ШІ, з яким ви можете спілкуватися безкоштовно. Ви можете поговорити з милим пінгвіном або скористатися його безкоштовними вправами на уважність для ефективного зняття тривоги, депресії та стресу. [2] Переваги перелычили, щодо мінусів:

- Не всі функції додатку є безкоштовні
- Хоч і обіцяє розуміючий штучний інтелект, насправді з огляду користувачів має дуже слабку систему штучного інтелекту.

MindDoc: Спутник психічного здоров’я. Розроблений клінічними психологами у тісній співпраці з провідними дослідниками для тих, хто хоче дізнатись про емоційне благополуччя або страждає на психічні захворювання від легкої до середньої тяжкості, включаючи депресію, тривогу, безсоння та розлади харчування. MinDoc дозволяє:

- Реєструвати своє психічне здоров'я та настрій в режимі реального часу.
- Отримувати розуміння та короткий опис своїх симптомів, поведінки та загального емоційного благополуччя, щоб допомогти вам розпізнати закономірності та знайти найкращі для вас ресурси.
- Відкривати їх бібліотеку курсів та вправ, які допоможуть користувачам у їх подорожі до емоційного благополуччя. [2]

Серед мінусів:

- Доступний лише андроїд юзерам
- Платний

Терапія самообслуговування **InnerHour**: тривога та депресія. Навчіть навичкам для хорошого психічного здоров’я за допомогою їх простих занять,

кожна з яких спирається на реальну терапію та дослідження психічного здоров’я в галузі КПТ, позитивної психології чи уважності. Продовжуйте вчитися, продовжуйте зростати завдяки таким ресурсам, як статті, поради та надихаючі цитати наших терапевтів. Наприклад, ви:

• Зрозумійте причини депресії

• Дізнайтеся про симптоми тривоги

• Знати, як визначити, чи пора їхати на терапію

Їх план самообслуговування відображає те, що відбувається в реальній терапії. Протягом 4 тижнів вони сподіваються допомогти вам відпрацювати навички боротьби з депресією, кращого сну, зняття тривоги, боротьби зі стресом та врешті-решт досягнення міцного психічного здоров’я.

Переваги:

- Фокусація на розповсюдженішу хворобу – депресію
- Сформований готовий курс

Недоліки:

- Специфічна направленість курсу на конкретну ціль, тобто зменшення страху та тривожності, а не контролю стану користувача взагалі.
- Частично платне
- Із звітів користувачів часто терпить крах системи та не втрачається результат.

Додатки Apple AppStore. На просторах AppStore досить мало додатків сконцентрованих саме на психічному здоров’ї, а зазвичай є міксом двох аспектів життєдіяльності людини.

HealthCare. Додаток Health спрощує візуалізацію та безпечне зберігання своїх медичних записів. Тепер пацієнти можуть зводити свої

медичні записи з багатьох установ разом зі своїми генерованими пацієнтами даними, створюючи більш цілісне уявлення про своє здоров'я. Недоліки – платне та сконцентроване просто на медичних показниках.

Додаток **Health**. Додаток Health було створено, щоб допомогти упорядкувати важливу інформацію про здоров’я та полегшити доступ до нього в центральному та безпечному місці.. Існують нові способи обміну даними зі своїми близькими та медичною командою, показник для оцінки стійкості ходьби та ризику падіння, а також аналіз тенденцій, який допоможе зрозуміти зміни у здоров’ї. Серед недоліків: орієнтованість лише на фізичне здоров’я, платне.

В середовищі Telegram для підтримки здорового психічно здоров’я було знайдено лише канали з літературою та статтями про те як справлятися зі стресом та ментальними проблемами. Серед ботів з релевантними назвами не було знайдено робочих екземплярів, які б відповідали хоча б на команду /start

Результати аналізу аналогів. Було визначино слабкы ы сильны сторони функцій додаткуз пов’язаних з психологічним здоров’ям. Був обраний вектор в сторону якого буде розроблюватися функціонал майбутньої інформаційної системи, а саме користувач повинен мати:

- Щоденик оцінки емоцій, та всього емоційного стана в цілоу.
- Розпізнавач емоцій
- Підтримку і поради
- Можливість проходити тести

1.2 Аналіз методів обробки природньої мови

Обробка природної мови, або коротше NLP, загалом визначається як автоматичне маніпулювання природною мовою, як-от мовою та текстом, за допомогою програмного забезпечення. Вивчення обробки природної мови існує вже понад 50 років і виросло з області мовознавства з розвитком

комп'ютерів. Тут ви дізнаєтесь, що таке обробка природними мовами і чому це так важливо.[13]

Підходи на основі RNN / LSTM

Більшість старих методів моделювання мови засновані на RNN (рекуррентна нейронна мережа). Прості RNN страждають від проблеми, відомої як проблема зникаючого градієнта, і тому не можуть моделювати довші контекстуальні залежності. Їх здебільшого замінили так звані довгострокові короткострокові нейронні мережі (LSTM), які також є формою RNN, але можуть охопити довший контекст у документах. Однак LSTM може обробляти послідовності лише односпрямованими, тому сучасні підходи, засновані на LSTM, еволюціонували у так звані двонаправлені LSTM, де ми можемо читати контекст зліва направо, а також справа наліво. Є дуже успішні моделі, засновані на LSTM, такі як ELMO або ULMFiT, і такі моделі досі діють для сучасних сучасних НЛП.

Підходи на основі трансформаторної архітектури

Одним з основних обмежень двонаправлених LSTM є їх послідовний характер, що ускладнює паралельне навчання. Архітектура трансформатора вирішує це шляхом повної заміни LSTM так званим механізмом уваги (Vashvani et al. 2017). З увагою ми бачимо цілу послідовність як єдине ціле, тому набагато легше паралельно тренуватися. Ми можемо змоделювати весь контекст документа, а також використовувати величезні масиви даних для попередньої підготовки в неконтрольованому режимі та для точного налаштування подальших завдань.

Сучасні моделі трансформаторів

Існує багато мовних моделей на основі трансформаторів. Найуспішнішими є (станом на квітень 2020 р.)

- Transformer (Google Brain/Research)

- BERT (Google Research)
- GPT-2 (OpenAI)
- XLNet (Google Brain)
- CTRL (SalesForce)
- Megatron (NVidia)
- Turing-NLG (Microsoft)

Між моделями є незначні відмінності. BERT вважається найсучаснішим результатом багатьох завдань NLP, але зараз схоже, що він перевершений XLNet також від Google. XLNet використовує моделювання мови перестановок, що тренує авторегресивну модель на всіх можливих перестановках слів у реченні. З метою ілюстрації в цій статті ми використаємо модель на основі BERT.

1.3 Аналіз методів конструювання інформаційних систем

В даному розділі буде проведено бізнес аналітику інформаційної системи, розроблено її архітектуру та дизайн її основних функцій.

UML (UML розшифровується як Unified Modeling Language.) - це стандартна мова для вказівки, візуалізації, побудови та документування артефактів програмних систем. UML була створена Групою управління об'єктами (OMG), а проект специфікації UML 1.0 був запропонований OMG у січні 1997 року.

Об'єктно-орієнтовані концепції були введені набагато раніше, ніж UML. На той момент не існувало стандартних методологій для організації та консолідації об'єктно-орієнтованого розвитку. Саме тоді UML з’явився в картині. Існує низка цілей для розробки UML, але найголовніше - це визначити якусь мову моделювання загального призначення, якою можуть користуватися всі моделісти, а також її потрібно спростити для розуміння та використання.

Діаграми UML створені не лише для розробників, а й для ділових користувачів, простих людей та всіх, хто бажає зрозуміти систему. Система може бути програмною або непрограмною системою. Таким чином, має бути зрозуміло, що UML не є методом розробки, а супроводжує процеси, щоб зробити його успішною системою.

На закінчення, ціль UML можна визначити як простий механізм моделювання для моделювання всіх можливих практичних систем у сучасних складних умовах.[3]

Серед діаграм та інших патернів мови UML ми використаємо та розробимо use case diagram та dataflow diagram, ER.

Для розроблення системи головним важливим аспектом є відображення динамічної поведінки додатку. Динамічна поведінка для інформаційної системи відображає стан коли додаток запущений, тобто працює.

Ці внутрішні та зовнішні агенти відомі як актори. Діаграми випадків використання складаються з акторів, випадків використання та їх взаємозв’язків. Діаграма використовується для моделювання системи / підсистеми програми. Діаграма одноразового використання фіксує певну функціональність системи. [3]

Завдяки онлайн ресурсу draw.io. Це безкоштовне програмне забезпечення для онлайн-діаграм. Ми можемо використовувати його як розробник блок-схем, програмне забезпечення мережевих діаграм, для створення UML в Інтернеті, як інструмент діаграми ER, для проектування схеми бази даних, для побудови BPMN в Інтернеті, як виробника схемних схем тощо.

Актори (зображення схкматичних чоловічків) – є користувачами системи, які не є конкретними її участниками, а лише використовують чи

доповнюють її функціонал. Або акторами можуть бути інші системи, що якимось чином взаємодіють з нашим додатком, наприклад Telegram чи система банку.

Варіанти використання (позначаються лежачими овалами) – є загальні випадки варіантів використання, що служать орієнтиром для розробників і способом перевірки якості для тестувальників, тобто це ті функції, той сервіс, що має надавати наша інформаційна система.

Великий квадрат на діаграмі позначає систему, яка розробляється, тобто кордони її функціоналу.

Стрілки чи лінії з’єднують відповідних користувачів (акторів системи) з їх варіантами використання, тобто тих які їм доступні. Є два особливих вида стрілочок <extend> та <included> .

<extend> використовується, коли варіант використання додає кроки до іншого першокласного випадку використання. <included> використовується для вилучення фрагментів випадків використання, які дублюються у багатьох випадках використання. Включений варіант використання не може стояти окремо, а оригінальний варіант використання не є повним без включеного.

Діаграма потоку даних (DFD) - це метод аналізу та проектування інформаційних систем. Це візуальний інструмент для зображення логічних моделей і виражає перетворення даних в системі. DFD включає механізм для моделювання потоку даних. Він підтримує декомпозиція для ілюстрації деталей потоків даних та функцій. DFD не може представити інформацію про послідовність операцій.

Потоки даних позначаються стрілками на діаграмі та підписуються, підпис повідомляє які конкретно дані йдуть в який процес. Прямокутниками відмічаються користувачі системи, до речі вони можуть дублюватися для зручності побудови. В скруглених прямокутниках відмічаються процес та їх

номери, процеси це основні функції системи, зазвичаю підписуються з використанням дієслів, щоб було зрозуміло, що конкретно робить процес. Останім значущим елементом діаграми потоків даних є бази даних, відмічаються як прямокутних зі зміщеним до іншої сторони стороною, підписуються відповідно до той інформації, яку база даних буде зберігати. Інколи означають не цілу базу даних, а лише деяку важливу таблицю, це потрібно для того щоб краще розуміти де які дані будуть зберігатися і в які процеси буде певна база даних надавати дані.

Є різні рівні діаграми потоків – DFD0, DFD1, DFD1.2 і так далі за необхідністю. Серед особливостей DFD0 є те що, вона має лише один процес, що і є основною інформаційною системою, функціонал якої розкривається на наступних рівнях діаграм – DFD1, DFD2 і так далі.

На стандартних діаграмах ER (Entity-Relationship Diagrams) прямокутники або квадрати представляють сутності, які є таблицями, що містять конкретну інформацію в базі даних. Діаманти представляють відносини, які є взаємодіями між сутностями. Овали представляють атрибути або дані, що описують сутність.

Хоча діаграми взаємозв'язку сутності можуть виглядати складними, ці діаграми допомагають обізнаним користувачам зрозуміти структури баз даних на високому рівні без супровідних деталей. Дизайнери баз даних використовують діаграми ER для моделювання взаємозв’язків між сутностями бази даних у чіткому форматі. Багато програмних пакетів мають автоматизовані методи створення діаграм ER з існуючих баз даних.[8]

Блоки діаграми є сутностями інформаційної системи, які в дальнійшому трансформуються в таблиці бази даних. Ці сутності мають ім’я воно стає назвою таблиці. У сутностей блоків є атрибути – це опис властивостей які майбутні об’єкти будуть мати. Атрибути в дальнійшому трансформуються в колонки бази даних.

Значення діаграми взаємозв'язку сутності полягає в її здатності відображати інформацію про взаємозв'язки між сутностями. На прикладі можна відстежити інформацію про місто, в якому проживає кожна людина. Також можна відстежувати інформацію про саме місто в структурі “Місто”, яка пов’язана між собою “Особа” та “Місто”.

Існує три типи взаємозв’язків між сутностями:

One-to-One: Іноді одна сутність асоціюється з іншою суттю. Наприклад, кожен працівник бази даних має лише один номер соціального страхування, і цей номер унікальний.

One-to-Many: окрема сутність також може бути пов’язана з кількома іншими сутностями. Наприклад, філія компанії та всі працівники, які працюють у цій філії, мають стосунки «один до багатьох».

Many-to-Many: кілька сутностей можуть бути пов’язані з кількома іншими сутностями. Наприклад, компанія може виробляти три товари та мати торговий персонал, який продає ці товари. Частина торгового персоналу може розподілити свій час між продуктами. [8]

1.3 Постановка задачі

В результаті проведеного огляду інформації, що стосується методів обробки природньої мови, методів рішення ментальних проблем та Телеграм API було принято рішення розробити телеграм бота з елементами штучного інтелекта в його функціоналі, з базою даних користувачів та їх щоденників та іншим функціоналом пов’язаний з наданням можливостей користувачам збільшити своє психічне благополуччя.

- Виконання поставленої означає виконання наступних задач:
- Розробити модель розпізнавання опису емоцій (підібрати дані, натренувати модель, провести тестування моделі)

- Розробити бота та його функціонал (отримати токен бота, додати список команд, описати обробники кожної команди)
- Розробити базу даних (розробити архітектуру бази даних, розробити таблиці)
- Імплементувати базу даних та інтелектуальну систему (модель) до боту, перевірить правильність роботи.

2. ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1 Діаграма використання

Представемо діаграму даного проекту. Розшифруєм значення і те, що мають під собою пункти діаграми (use cases):

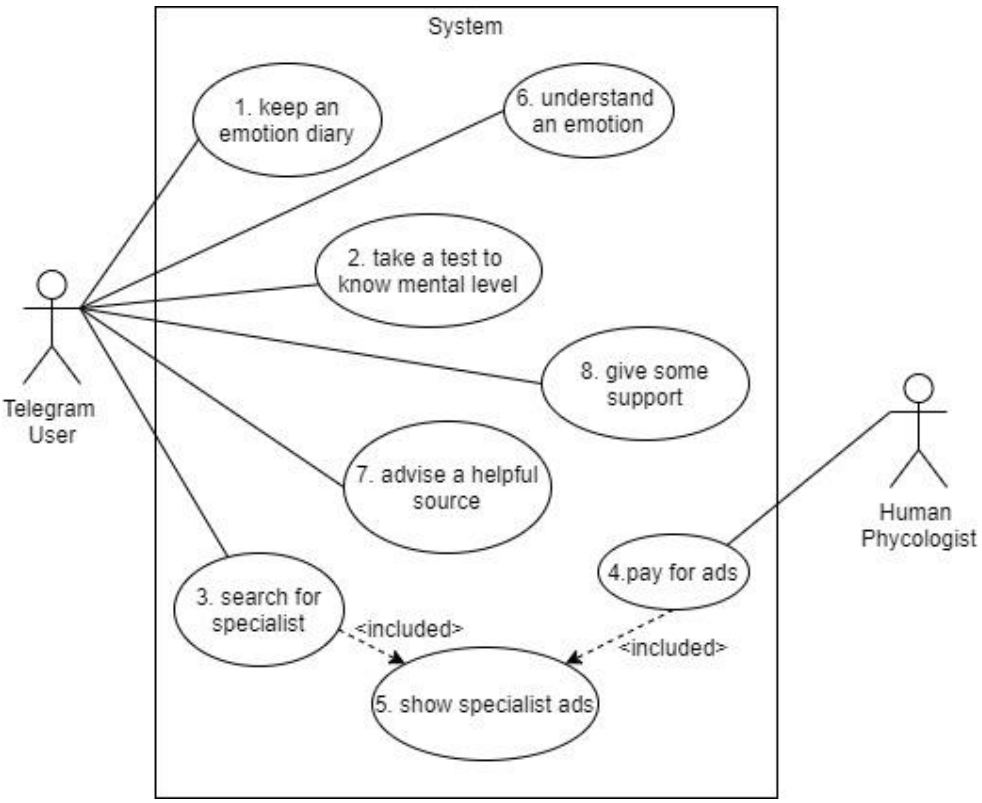

Рисунок 1. – Діаграма використання

1. **Keep an emotion diary.** Даний use case дозволяє користувачу розроблюваного телеграм боту вести щоденик емоцій, та відслідковувати свій емоційний рівень.

Робити записи про те, яку емоцію користувач відчуває, оцінювати шкалу гарно дня. Такі записи є одним із способів боротьби з будь-якими непереборними емоціями це пошук здорового способу самовираження. Це робить журнал корисним інструментом для догляду за психічним здоров'ям. Такий щоденик може допомогти користувачу: управляти тривогою, зменшити

стрес, справитися з депресією (в плані депресії це один із 4 основних способів її лікування).

При емоційно нестабільних розлад особистості щоденник допомагає контролювати симптоми та покращувати настрій, оскільки: допомогає у визначенні пріоритетів щодо проблем, страхів та турбот; щоденик допоможе розпізнавати тригери, та вивчити кращі способи контролю над ними; щоденик дозволить виявить негативні думки та дасть користувачу зробити крок до позитивної самарозмови.

Працюватиме наступний use cese таким чином, користувач натискає на команду типу «емоційно оцінити день» та на шкалі від -4 до 4 виставляє оцінку. (-4) – дуже поганий день – виникали жахливі думки про безглуздість життя, про ненависть до себе і схожу, (- 2) – день був невдалий, не встиг на автобус, із-за цього запізнився на роботу, а там пропустив важливу зустріч, ще і забув вдома важливий документ, зіпсувався настрій. (0) – звичайний день, нічого погано не сталося, але і нічого хорошого, можно назвати рутиним днем чи просто буденим; оцінкою 2 – позначаються вдалі дні, наприклад отримав шоколадку у подарунок до кави чи зробив велопрогулку в ппарку, цифрою (4) відмічаються дні коли емоції були через край і в основному радісні – було день наробження, де було багато друзів, відкривалися подарунки та були танці, тобто коли рівень радості просто нереально високий.

Команда додавання емоції користувача. Більшість емоційно нестабільних розлад особистості зумовлені тим, що люди не розусіють свої основних емоцій, або не можуть їх осмислити і правильно висловитти. Команда додавання призначена для тих користувачів, що можуть зрозуміть, що вони приблизно відчувають і додати цю емоцію до щоденика. Для тих хто не розуміє свої природні емоції є use case 6. Understand an emotion, який по фразам користувача підкаже йому, що той відчуває і зможе одати до щоденика запис.

2. **Taking a test to know a mental level.** Даний use case дозволяє користувачу пройти спеціалізовані психологічні тести, а саме тест шкала депресії Бека та тест Спілбергера-Ханіна

Шкала депресії Бека (BDI - Beck Depression Inventory) - це шкала з 21 елементом, що оцінює основні симптоми депресії, включаючи настрій, песимізм, почуття невдачі, невдоволення собою, провину, покарання, нелюбов до себе, самозавинення, ідеї самогубства , плач, дратівливість, соціальна абстиненція, нерішучість, зміна іміджу тіла, труднощі в роботі, безсоння, стомлюваність, втрата апетиту, втрата ваги, соматична зайнятість та втрата лібідо (Beck & Steer, 1993; Beck, Steer & Garbing, 1988) .

Хоч і планується розробити бота для підтримки психічного здоров'я вцілому, але на перших моментах його запуску хочеться сфокусуватися на більш загальних та роповсюдженіших випадках та захворюваннях. А одним із найбіль розповсюдженішим ментальним захворюваням є депресія. За даними всесвітньої організації охорони здоров'я (World Health Organisation). Ключові факти:

- Депресія є загальним психічним розладом. У всьому світі понад 264 мільйони людей різного віку страждають від депресії.
- Депресія є основною причиною інвалідності у всьому світі і є головним фактором загального тягаря захворювань у світі.
- Більше жінок страждає від депресії, ніж чоловіків.
- Депресія може призвести до самогубства.
- Існують ефективні психологічні та фармакологічні методи лікування середньої та важкої депресії.

Існують ефективні методи лікування середньої та важкої депресії. Постачальники медичних послуг можуть пропонувати психологічні методи лікування або антидепресанти. Розроблювану систему можно віднести до когнітивно-поведінкової терапії. Адже користувач може осмислити свої дії та

емоції завдяки нашого боту. Але система не може бути єдиним методом лікування. Наш додаток та методи які він використовує для лікування ефективні при легкій депресії.

Тест Спілбергера-Ханіна є одним із методів, що застосовуються для дослідження психологічний феномен тривожності. Цей дескриптор складається з 20 тверджень, які розглядати тривогу як стан (стан тривоги, реактивний чи ситуативний тривожність) та 20 тверджень для визначення тривожності як диспозиції, особливостей особистість (властивість тривоги).[4]

Випадки тривоги - це звичайна частина життя. Однак люди з тривожними розладами часто відчувають сильне, надмірне та постійне занепокоєння та страх щодо повсякденних ситуацій. Часто тривожні розлади включають повторювані епізоди раптових почуттів сильної тривоги та страху або жаху, які досягають піку протягом декількох хвилин (напади паніки).

Приклади тривожних розладів включають генералізований тривожний розлад, соціальний тривожний розлад (соціальна фобія), специфічні фобії та розлучний тривожний розлад. Ви можете мати більше одного тривожного розладу. Іноді тривога виникає внаслідок захворювання, яке потребує лікування.[6]

Після вибору команди пройти тест, бот дасть користувачу вибрати тест який він хоче чи тест Спілбергера-Ханіна чи шкалу депресії Бека і далі використовуючи інтерфейс фейсбуку буде користувач зможе виконати тестування та дізнатися свій психічний стан.

3. **Search for a specialist**. Даний use case допомагає знайти людину, що пропонує свої послуги як спеціаліста психолога або психіатра. Користувач відає відповідну команду боту, а той пропонує йому кандидатури та надає їх контакти.

У якийсь момент у нас виникне класична плутанина щодо різниці між психологом та психотерапевтом. Однак порівняно легко визначити різницю між психологом та психотерапевтом. Психотерапевт - це людина, яка, будучи терапевтом або психологом, також пройшла спеціальне навчання. Психотерапія - це психологічне втручання - механізм, який спрямований на боротьбу зі здоров'ям та захворюваннями..

4. **Pay for ads**. Цей use case доступний лише акторам користувачам спеціалістам психологам та терепевтом, тобто тим юзерам, які бажають знайти більше клієнтів за допомогою нашої системи та заробити кошти. Але перед цти вони повинні заплатити за використання нашого ресурсу як місця для реклами.
5. **Show spechialis ads.** Даний use case надає користувачам можливість переглянути перелік спеціалістів, які співпрацюють з ботом та рекламують завдяки нього свої послуги. Користувач переглядає інформацію про кваліфікацію та досвід спеціаліста психолога чи психотерапевта та вибирає чи він бажає співпрацювати з цьою людиною.

Сама терапія віч на віч користувача та обраного ним спеціаліста не являється частиною розроблюваної системи. Бот є лише своєрідним ринком для пошуку клієнтів для працівників сфери контролю ментального здоров'я і людей як потребують психологічної допомоги і підтримки.

6. **Understand an emotion** . Цей use case дає актору користувачу зрозуміти те що він відчуває, в тому випадку, коли людина не може правильно оцінити власні почуття та емоції. Користувач натискає відповідну команду та вводить фрази якими він описує свій стан.

Цей use case має під собою модель машиного навчання, яка по фразам користувача визнача емоцію яку він відчуває. Розроблена модель, які способи

навчання та інші дані були описані в попередній роботі, а також згадаються в цій, але нижче.

7. **Advise a helpful source**. Даний use case просто надає користувачу посилання на корисні ресурси з інформацією про психічне здоров’я, про додаткові тести, поради як уникати стресів, що таке стрес, що таке тривога, що таке граничні розлади психіки.

Також даний use case дає посилання на корисні статті, курси та тести. Відправляє список корисних книг та підборку відео від спеціалістів з YouTube. Можно сказати, що даний спосіб використання просуває ресурси конкурентів, так і є, але психічне здоров’я це дуже важлива річ і не можна нехтувати життями людей лише із-за прибутку.

8. **Give some support.** Даний use case відповідає за підтримку користувача, і надсилає йому мотиваційні фрази та цитати, інколи надсилає картинки, що полегшують рівень напруги. Ці картинки в повсякденості називають просто меми.

Люди, які страждають на депресію, відносяться до мемів в Інтернеті, які зображують темний та гнітючий вміст більш позитивно, ніж ті, хто не страждає симптомами депресії, згідно з новим дослідженням Університету Шеффілда Халлама.

Це дослідження, проведене науковцями кафедри психології, соціології та політики університету, вивчало різні трактування людей, які страждають на клінічно значущі симптоми депресії, порівняно з контрольною групою, яка не страждала на депресію.

У дослідженні взяли участь 43 людини з депресією, а також 56 людей, які не страждають на депресію. Кожна людина переглядала низку депресивних і нейтральних інтернет-мемів. Сприйняття гумору, взаємозв'язку, можливості

спільного використання та покращення настрою депресивних мемів були все більшими серед осіб із симптомами депресії.

Отримані дані свідчать про те, що, незважаючи на свою негативну орієнтацію, інтернет-меми, пов’язані з депресією, можуть бути корисними для людей, які мають постійні симптоми.

2.2 Діаграма потоків даних

Розроблена data flow diagram дозволить краще зрозуміти, які дані в яких блоках коду будуть передаватись.

Рисунок 2 – DFD0

На наступному малюнку зобразимо DFD1, діаграму потоків даних першого рівня, яка опише основні потоки даних, те що вводить користувач та ті дані, що видаються, розглянемо конкретно по кожному процесі, що являються підпроцесами головного процеса DFD0. Малюнок нижче DFD1 розроблюваної інформаційної системи.

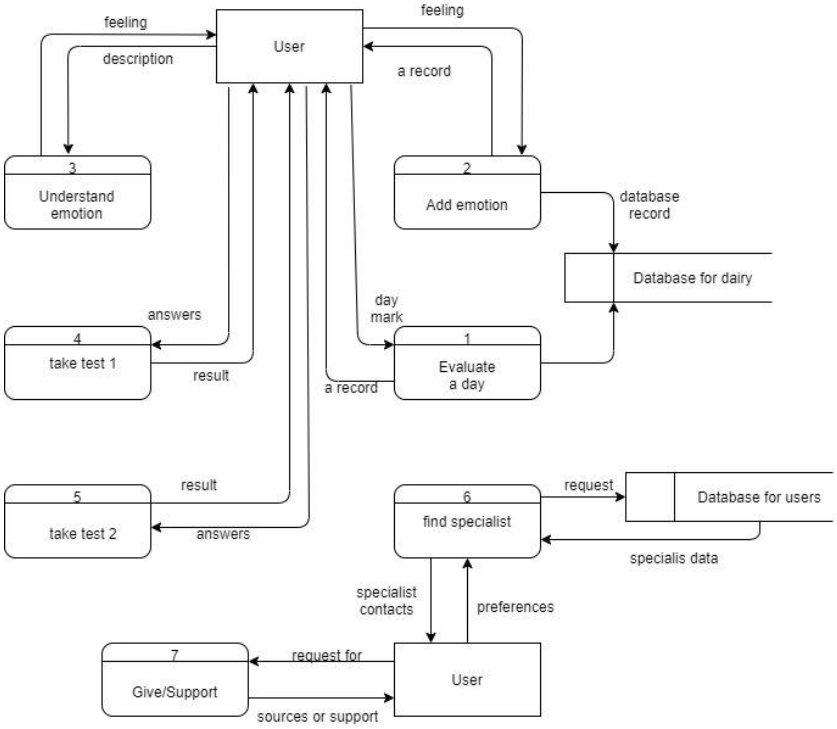

Рисунок 3 – DFD1

Опишимо деякі елементи даної діаграми. Прямокутник з підписом **User** є користувачем телеграму, та конкретно нашого бота. На діаграмі користувача двоє – це зроблено лише для зручності, щоб лінії стрілочок не виглядали надто запутаними.

На діаграмі зображено дві бази даних: **Database for diary** та **Database for users**, це не є окремі бази даних, а лише окремі таблиці, але онаковао важливі і мають в собі кардинально різну інформацію. Перша база даних для щоденників, в неї надходять та зберігаються дані про емоційний стан користувача та його оцінка дня. В другій базі даних – базі користувачів зберігаються дані про користувачів бота та психологів дані котрих будуть надаватися користувачам, яким це необхідно.

Перешлянемо перелік процесів. Під першим номером знаходиться процес **Evaluate a day.** Вхідними даними даного потоку є оцінка дня в шкалі від -4 до 4, вихідними даними є запис в базі даних та запис для користувача.

Add emotion є другим процесом діаграми потоків даних другого рівня вхідними даними є вибір емоції користувачем, вихідними даними є запис у щоденик.

Understand emotion є третім процесом діаграми потоків даних другого рівня, вхідними даними користувача є опис того що він відчуває, вихідними даними є відповідь моделі, те як вона розпізнала емоцію.

Take test 1 є четвертим процесом діаграми потоків даних другого рівня, користувач дає відповіді на запитання тест шкали депресії Бека, а вихідними даними є результат тестування, порахований рівень деприсивного значення та його опис.

Take test 2 є п’ятим процесом діаграми потоків даних другого рівня вхідними даними користувача дає відповіді на запитання тесту Спілбергера-Ханіна та отримує відповідь від бота – вихідні дані: оцінку його рівня тривожності, тобто результат тестування.

Find specialist є шостим процесом діаграми потоків даних другого рівня, вхідними дааними користувача є лише запрос того, що він бажає знайти спеціаліста, а вихідними даними є перелік спеціалістів, які пропонують свої послуги та предоставлені їх контакти.

Give/support є сьомим процесом діаграми потоків даних другого рівня, хоч це процес представлений двома способами використання, але являється схожим у своїй реалізації, вхідними даними користува є лише запрос на отримання даних, вихідними даними є перелік корисної літератури, відео та статей, в випадку підтримки вихіні дані це зображення чи цитата з підтримуючим тоном.

2.3 ER діаграма

Розробимо ER діаграму для нашого проекту.

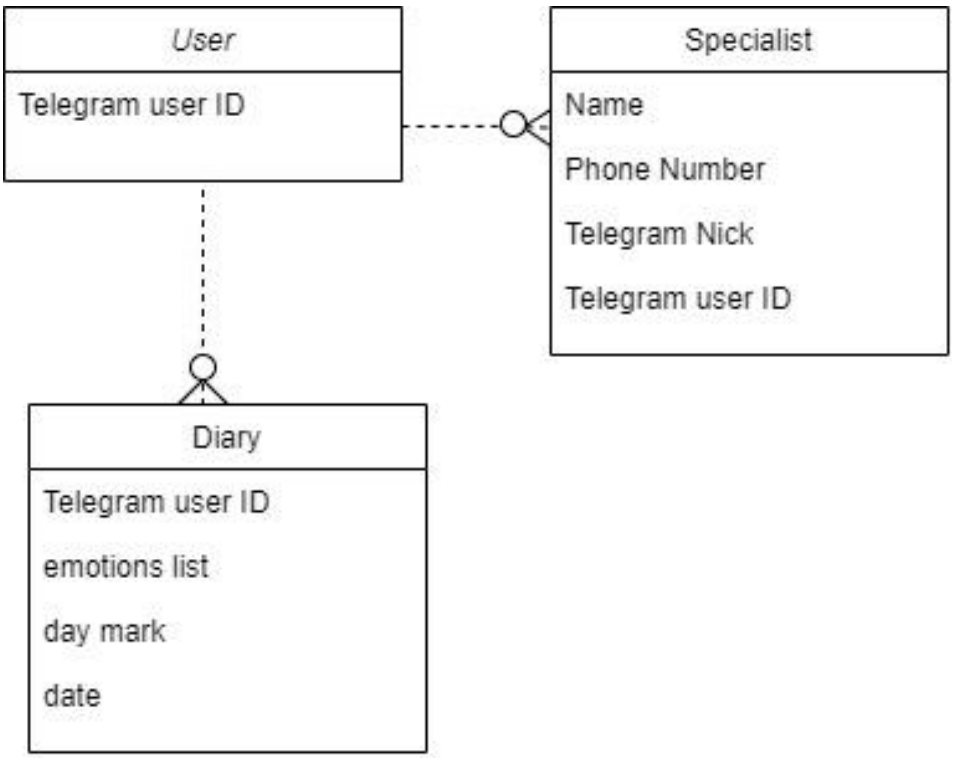

Рисунок 4 – ER діаграма

З даної діаграми ми бачимо, що в системі буде три таблиці бази даних: User, Specialist, Diary. В таблицы User лише один атрибут це telegram_user_id для збереження телеграм айді. Таблиця Specialist містить дані про психологів та психіатрів, має чотири атибути: name (ім’я спеціаліста), phone_number (номер телефона спеціаліста), telegram_user_id (айді користувача телеграм), telegram_nick (нік нейм користувача). Наступна таблиця Diary буде мати наступні колонки: telegram_user_id (айді користувача телеграму), emotion list (список емоцій відповідного дня), day mark (оцінка дня), date (дата відповілного дня).

В таблиці User telegram_user_id є primery key, він же мігрує в таблиці Specialist та Diary і стає там foregign key. А також в таблиці Diary атрибут telegram_user_id разом з date стає зіставним ключем і разом формують первиний ключ.

3. РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ – БОТА АСИСТЕНТА ПСИХОЛОГА

3.1 Реалізація та навчання інтелектуальної системи

DataSet або набір даних – це колекція, структурований певним чином набір даних, інколи таблиці, або просто файли з сеператорами (роздільними знаками). Як правило, набір даних містить більше однієї змінної і стосується однієї теми; ймовірно, це стосується однієї вибірки. Певних записів, які є одиницею інформації у відповідному датасеті повино бути дуже велика кількість.

В розроблюваній системі датасетом є три файли розширення txt – train, test і val (validation) та мають наступний вигляд:

i didnt feel humiliated	sadness
i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake	sadness
im grabbing a minute to post i feel greedy wrong	anger
i am ever feeling nostalgic about the fireplace i will know that it is still on the property	love
i am feeling grouchy	anger
….	…..

Таблиця 1. Вигляд датасету

Датасет має записи. Один запис характеризується наявністю опису і лейблу цього запису. Даний датасет був знайдений на безкоштовному сайті kaggle.com. та не є нашою розробкою.

Проведемо аналіз даних, підрахуємо кількість рядків, кількість леблів, їх назви та інше.

```
train_df = pd.read_csv('data_set/train.txt', sep=';')
test_df = pd.read_csv('data_set/test.txt', sep=';')
val_df = pd.read_csv('data_set/val.txt', sep=';')
print(train_df.shape)
print(test_df.shape)
print(val_df.shape)

train_df.columns = ['sentence', 'emotion']
test_df.columns = ['sentence', 'emotion']
val_df.columns = ['sentence', 'emotion']
print(train_df.head())

print(train_df['emotion'].value_counts())
print(test_df['emotion'].value_counts())
print(val_df['emotion'].value_counts())

def max_len(data):
    return data['sentence'].apply(lambda x: len(x.split())).max()
max_lens = [max_len(train_df), max_len(test_df), max_len(val_df)]
max(max_lens)
print(max_lens)
```

Рисунок 5 – код для аналізу даних

І вивід можемо спостерігати на консолі:

```
(15999, 2)
(1999, 2)
(1999, 2)
                                                sentence  emotion
0  i can go from feeling so hopeless to so damned...  sadness
1   im grabbing a minute to post i feel greedy wrong    anger
2  i am ever feeling nostalgic about the fireplac...     love
3                              i am feeling grouchy    anger
4  ive been feeling a little burdened lately wasn...  sadness
joy         5362
sadness     4665
anger       2159
fear        1937
love        1304
surprise     572
Name: emotion, dtype: int64
joy         695
sadness     580
anger       275
fear        224
love        159
surprise     66
Name: emotion, dtype: int64
joy         704
sadness     549
anger       275
fear        212
love        178
surprise     81
Name: emotion, dtype: int64
[66, 61, 61]
```

Рисунок 6 – Аналіз даних

Токенізація. Це процес перетворення вхідних даних моделі в зрозумілий вигляд для моделі. На початку першого речення вставляється токен [CLS], а в кінці кожного речення - токен [SEP]. Також видаляються лишні символи та все приводиться до нижнього регістру. Далі словам присвоюється векторні значення – замість слів набір чисел. Та накладаються маска уваги.

Напишемо модуль який буде преобразовувати вхідні повідомлення від користувача, а також під час навчання та тестування моделі. Наприклад подамо на модель фразу «I feel very worried I nearly breathe», токенайер видасть дальнійший результат.

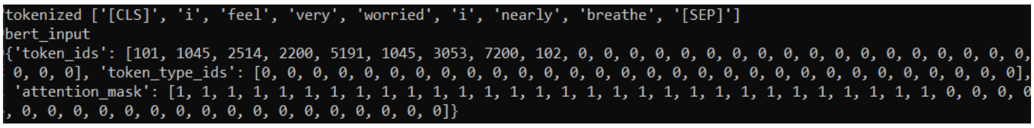

Рисунок 7 – Результат токенізації

Перед тренуванням моделі чи правильніше сказати перед fine turning. Це процес нарощування нових зв’язків поверх вже натренованої моделі. Вихідними даними моделі (Li) – це лейбл емоції.

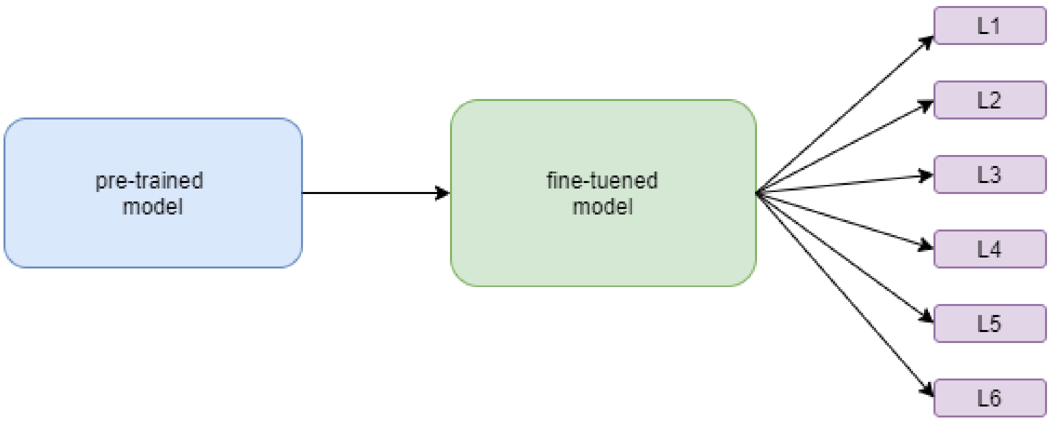

Рисунок 8 – Архітектура BERT

Ми створюємо зміні які відповідають за конфігурацію в файлі моделі: довжина вхідних рядків, розташування датасетів, ім’я моделі, слоіник лейблів, тобто кожній емоції присвоюємо числове значення.

LABEL_DICT = {'joy':0, 'sadness':1, 'anger':2, 'fear':3, 'love':4, 'surprise':5}

Далі створюємо клас для датасету - class Emotions_Dataset(Dataset), та для моделі розпізнавання емоцій - class BertEmotionClassifier (BertPreTrainedModel)

Розробляємо функцію оцінки, яка буде оцінювати навчання моделі, якв буде порівнбвати припущену оцінку з фактичною оцінкою, під оцінкою мається на увазі номери лейблів. Запускаємо процес навчання, котрий триватиме приблизно 4-5 годин на одному CPU.

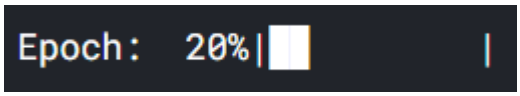

Рисунок 9 – Процес розпочався

Чекаємо час та виконуємо перевірку з тестовим датасетом та валідацію моделі.

3.2 Реалізаці Telegram бота

В цьому розділі буде розглянутий процес реалізації функціоналу бота відповідно до описаних вище схем.

Для початку додамо список команд для бота. Давайте для початку розберемося з таким поняттям, що таке взагалі команда в системі спілкування з телеграм ботом. Залежно від налаштувань бота, він може або бачити будь-які повідомлення будь-якого формату або тільки спеціально оформлені команди. Є два види команд, перший звичайний текст - звичайні повідомлення. У такому вигляді бот отримує повідомлення, коли йому пишуть в особистий чат. Інший тип спеціально оформлені команди[9]

Такі команди завжди починаються з косою риси: « / » Після слідує сама команда. Текст команди повинен бути без пробілів. приклад:

/ start

З цієї команди вище будь-який користувач завжди починає спілкування з вашим ботом. Тому за правилами хорошого тону реакцію на цю команду потрібно прописувати обов'язково, що ми і зробимо далі. На команду /start бот буде вітати користувача та розказувати коротко про те що він може робити:

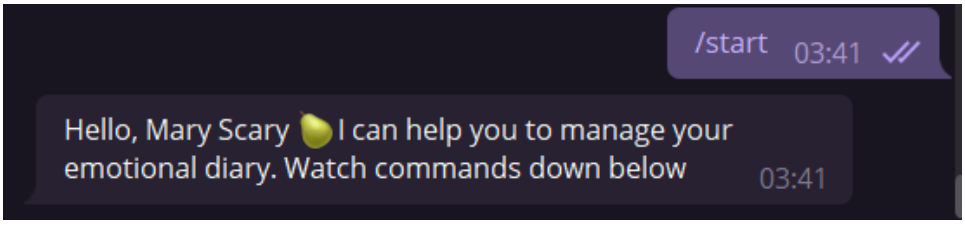

Рисунок 5 – Відповідь на команду запуску

А зараз створемо список команд та надішлемо його @BotFather, щоб редагувати бота.

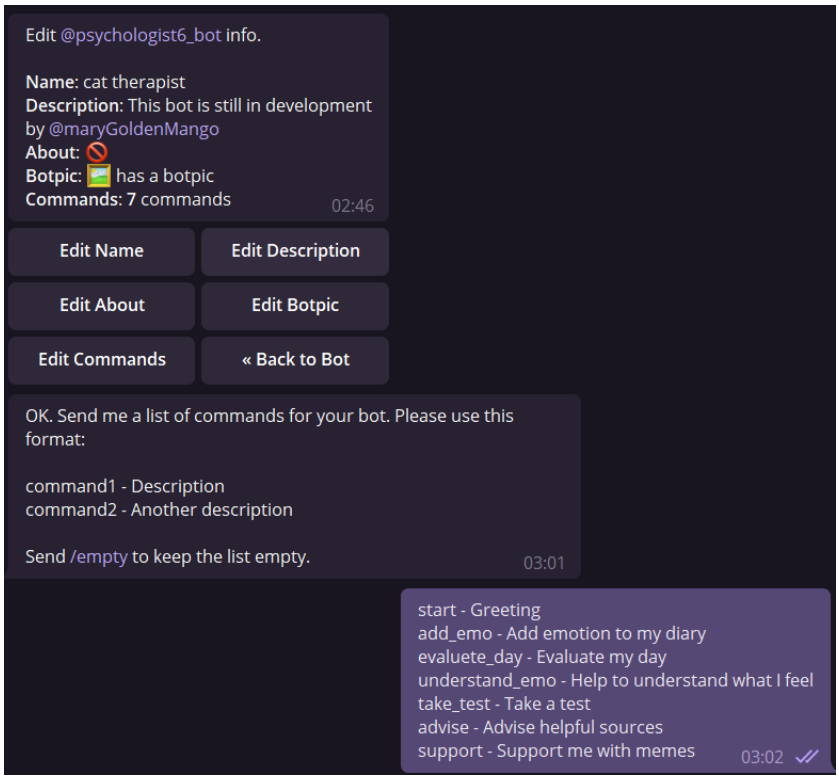

Рисунок 6 – Додавання команд бота

Тепер команди будуть візуально відображатися користувачу. І він зможе обрати будь-яку, вона підставиться в повідомлення та після відправлення буде надіслана боту на обробку.

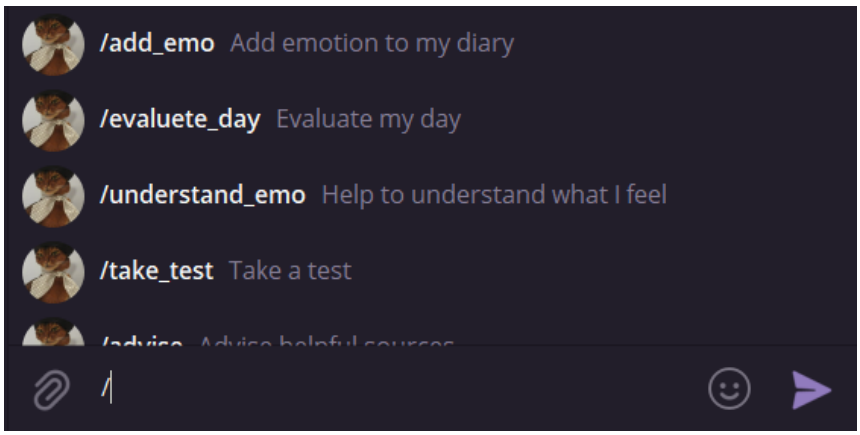

Рисунок 7 – Команди бота

Розробимо обробник команди для команди /**support**, за діаграмами розробленими перед цим ця команда буде надсилати підбадьорюючі цитати або зображення. Для його ми розробляємо функцію обробки подій - message_handler. Обробники повідомлень визначають фільтри, які повинно проходити повідомлення. Якщо повідомлення передає фільтр, викликається функція оформлення та вхідне повідомлення передається як аргумент.

@bot.message_handler(commands=['support'])

def send_helpful_sources(message):

Функція, яку прикрашає обробник повідомлень (message_handler), може мати довільне ім'я, однак воно повинно мати лише один параметр (файл повідомлення).

Далі за допомогою рандомайзера чисел вибираємо чи зображення чи вислів відправити користувачу та надсилаємо. Перелік цитат невеликий він же знаходиться в файлі bot.py. А перелык зображень значно быльше тому для нього був створений окремий модуль Python.

Модуль дозволяє вам логічно впорядкувати ваш код Python. Групування відповідного коду в модуль полегшує його розуміння та використання. Модуль - це об’єкт Python із довільно названими атрибутами, на які ви можете пов’язати посилання.

Просто модуль - це файл, що складається з коду Python. Модуль може визначати функції, класи та змінні. Модуль також може включати виконуваний код.

В модулі img_meme.py створеним для команди support створюємо масив з посиланнями на зображення та додаємо функцію get_img_meme(), яка повертає випадково вибране зображення.

За таким же принципом розроблений модуль helpful_source.py для команди advice, з масивом корисних послань та функцією їх предачі get_advice(). Ця функція як і get_img_meme() використовує рандомайзер чисел у описаном діапазоні, щоб визначити число, яке і буде індексом елемента, який буде надісланий користувачу.

Ми використовуємо бібліотеку random для цього. Random – це вбудований модуль, за допомогою якого можна робити випадкові числа. Цей Random модуль має набір методів, але ми використовуємо конкретно функцію randint, адже нам необхідні випадкові інтенжери в заданому діапазоні.

Перейдемо до розробки команди /**understand_emotion**, ця команда відправляється користувачем боту для того щоб по опису бот відповів, що це за емоція. Бот відповідає: Describe what you feel. Користувач повинен ввести те, що він відчуває і бот відправить повідомлення про приположну емоцію.

Для реалізації цієї функції ми вішаємо месендж хендлер, майже такий же як описаний вище, але з іншою командою та оголошуєм функцію:

@bot.message_handler(commands=['understand_emotion'])

def understanding_emotion(message):

В цій функції ми повині організувати register_next_step_handler, регістратор наступного кроку. Це необхідно для того, щоб бот “зрозумів”, що користувач буде вводити вхідні дані, а не команду знову.

msg = bot.send_message(message.chat.id, "Describe what you feel")

bot.register_next_step_handler(msg,understand_emotion_with_bert_mode)

Ми зберігаємо в msg те повідомлення на якому мпускаємося в дерево хендлерів. Далі додаємо цей об’єкт та функцію, яка буде виконувати обробку далі в якосту параметрів реєстратора хендлерів.

Для того, щоб цей реэстратор працював необхідно додати enable_save_next_step_handlers та load_next_step_handlers перед початком запуску бота.

bot.enable_save_next_step_handlers() – цей рядок дозволяє зберігати стан діалога

bot.load_next_step_handlers() – а цей рядок дозволяє завантажувати наступний хендлер, тобто переходити на наступний крок алгоритму.

bot.polling() – команда, рядок який сигналізує про запуск бота.

Повернемось до реалізації функції understanding_emotion, ця функція приймає параметром повідомлення користувача передає його в модель машиного навчання та остання видає припустиму емоцію. Функція яка це виконує emo.tokenize(message.text) виконує розкладення на токени та передає далі до моделі на розпізнавання. Ця функція лежить в модулі emo де знаходиться вся логіка моделі bert розробленої на попередніх етапах проектування.

BERT (Bidirectional Encoder Reprezentations from Transformers) - нещодавня стаття, опублікована дослідниками Google AI Language. Це викликало ажіотаж у спільноті машинного навчання, представивши найсучасніші результати у різноманітті завдань НЛП, включаючи відповіді на запитання (SQuAD v1.1), висновок про природну мову (MNLI) та інші.

Ключовою технічною інновацією BERT є застосування двонаправленого навчання Transformer, популярної моделі уваги, до моделювання мови. Це на відміну від попередніх зусиль, які розглядали

текстову послідовність або зліва направо, або комбіновану підготовку зліва направо та справа наліво. Результати статті показують, що мовна модель, яка є двонаправленою, може глибше відчувати мовний контекст і течію, ніж однонаправлені мовні моделі. У статті дослідники детально описують нову методику під назвою Masked LM (MLM), яка дозволяє двоспрямоване навчання в моделях, в яких це було раніше неможливо.

Розробка дальнійших команд вимагає наявності бази даних у проекта. Тому було обрано базу даних SqlLite.

3.3 Реалізація бази даних

SQLite - це бібліотека (набір заголовків, класів та функцій) написана на мові програмування C, яка реалізує і є реалізацією бази даних, що забезпечує її функціонування.

SQLite - це найбільш широко використовуваний в світі механізм баз даних. SQLite вмонтований в усі мобільні телефони і більшість комп'ютерів і включений в багато інших програми, які люди використовують щодня. Більш детальна інформація нижче. Вихідний код SQLite знаходиться у відкритому доступі і може використовуватися будь-ким в будь-яких цілях.

SQLite - це технологічна бібліотека, яка реалізує автономний, бессерверной, транзакційний механізм бази даних SQL. Код SQLite є загальнодоступним і, отже, безкоштовним для будь-яких цілей, комерційних чи приватних. SQLite - сама використовувана база даних в світі, вона містить більше програм, ніж ми можемо обчислити, включаючи кілька великих проектів. SQLite - це легка бібліотека. Існує компроміс між використанням пам'яті і швидкістю. Зазвичай SQLite буде працювати швидше, чим більше пам'яті ви йому надасте. Однак продуктивність в цілому непогана навіть в середовищах з низьким обсягом пам'яті. Залежно від того, як він

використовується, SQLite може бути швидше, ніж пряме введення-виведення файлової системи.[11]

Вибір SQLite був зроблений із-зп ряда переваг:

- Краща продуктивність
- Зниження вартості та складності програми
- Переносимість
- Надійність
- Доступність

Також SQLite важають найпрекраснішим вибором при розробкі невеликих додатків і проектів (як наприклад даний), вона є гарним вибором також для мобільних додатків.

Щоб використовувати модуль, спочатку потрібно його імпортувати (import sqlite3), далі створити об’єкт Connection, який представляє базу даних. В нас дані будуть зберігатися у файлі bot_phyco.db:

db = sqlite3.connect('bot_phyco.db')

Також необхідно створити об’єкт курсор (клас sqlite3. Cursor) який міститеме дані після виконаних запросів. Екземпляр курсора має такі атрибути та методи:

execute(sql[, parameters]). Виконує оператор SQL. Значення можуть бути прив’язані до виписки за допомогою заповнювачів. execute () виконає лише один оператор SQL. Якщо ви спробуєте виконати з ним більше одного виразу, це викличе Попередження. Використовуйте Executescript (), якщо ви хочете виконати кілька операторів SQL одним викликом.[12] Самец ей метод ми в основному будемо використовувати для звернень до бази даних. Він використовується при обробнику подій з командами start, add_emotion, evaluate_day, specialist.

fetchone() Отримує наступний рядок набору результатів запиту, повертаючи одну послідовність або None, коли більше даних немає.

Розпочнемо з команди specialists. Вона трохи схожа з командами /advise та /support. Адже використовується рандомайзер, так як база спеціалістів невелика. Користувач надсилає команду, а бот надсилає дані психолога чи терапевта – його ім'я, телеграм нік, номер телефону та короткий опис. І користувач сам обирає чи бажає він зв'язатися з цим спеціалістом. Функція showing_specialist виконує декілька подібних запитів до бази даних та складає відповідь поступово користувачу:

name = sql.execute ("ВИБЕРІТЬ ІМЯ ВІД спеціалістів ДЕ id_spe =?", (запустив,))

answ = "Meet " + str (name.fetchone () [0])

Далі бот відправляє answ методом send_message

В дальнійшому можливо будуть додані об'єкти зображень для спеціалістів, тобт їх фото. Також додане посилання на документи, що підтвердить кваліфікацію спеуваліста, або його резюме. Створимо необхідні таблиці для кристувача, їх щоденників та спеціалістів. Скрипт на SQL на малюнку виконує ці дії.

```
1  CREATE TABLE users (
2  id_user INTEGER PRIMARY KEY AUTOINCREMENT,
3  chat_id INTEGER);
4
5  CREATE TABLE diary (
6  date TEXT,
7  mark INTEGER,
8  emotion_list TEXT,
9  user INTEGER,
10 FOREIGN KEY(user) REFERENCES users(id_user));
11
12 CREATE TABLE specialists (
13 id_spe INTEGER PRIMARY KEY AUTOINCREMENT,
14 name TEXT,
15 nickname TEXT,
16 number TEXT,
17 describe TEXT);
```

Рисунок 8 – Створення таблиць sql

Далі розглянемо команду **evaluete_day**. Обробляємо її вже знайомим message_handler. Та оголошуємо функцію evalueting_day з параметром message. Для зручності користувача додаємо ReplyKeyboardMarkup. ReplyKeyboardMarkup – це обєкт представляє власну клавіатуру з опціями відповіді, тобто це кастомна клавіатура, для більш точної відповіді користувача. keyboard Array of Array of KeyboardButton Масив рядків кнопок, кожен представлений масивом об’єктів KeyboardButton. Оголошуємо цей об’єкт:

markup = types.ReplyKeyboardMarkup(resize_keyboard = True, row_width=3, one_time_keyboard=True), де resize_keyboard це логічне (булевеб Boolean) значення. Необов’язкове. Просить клієнтів змінити розмір клавіатури вертикально для оптимального прилягання (наприклад, зменшити клавіатуру, якщо є лише два ряди кнопок). За замовчуванням значення false - у цьому випадку спеціальна клавіатура завжди має однакову висоту зі стандартною клавіатурою програми. row_width – ширина рядка, але по факту кількість стовбців , тобто колонок в кастомній клавіатурі. selective - це логічне (булевеб Boolean) значення. Необов’язкове. Використовуйте цей параметр, якщо ви хочете показувати клавіатуру лише певним користувачам. Цілі: 1) користувачі, які @ згадуються в тексті об'єкта Повідомлення; 2) якщо повідомлення бота є відповіддю (має reply_to_message_id), відправник вихідного повідомлення. one_time_keyboard це логічне (булевеб Boolean) значення. Необов’язкове. Просить клієнтів приховати клавіатуру, як тільки вона використовується. Клавіатура все ще буде доступна, але клієнти автоматично відображатимуть звичайну буквену клавіатуру в чаті - користувач може натиснути спеціальну кнопку в полі введення, щоб знову побачити власну клавіатуру. За замовчуванням значення false.

Ми оголошуємо ряд із дев’яти об’єктів кнопок клавіатури, іменує міх від btn1 до btn9, а значення яке вони відображають від -4 до 4. Та додаємо їх створеного об’єкту marckup. Та прикріпляємо до відповіді бота:

msg = bot.send_message(message.chat.id, answ, parse_mode='html', reply_markup=markup)

Так як користувач має оцінити день, тобто вести додаткові дані ніж команда, ми реєструєм реєстратор хендлера та передаємо йому наступну функцію adding_mark_to_db.

bot.register_next_step_handler(msg, adding_mark_to_db)

В цій функції adding_mark_to_db ми повторюємо те, що вів користувач та додаємо ці дані до бази даних та конкретно в таблицю diary створену вище наступною функцією:

sql.execute("INSERT INTO diary(date, mark, user) VALUES (?, ?, (SELECT id_user FROM users WHERE chat_id = ?))", (tday, int(message.text), message.chat.id,))

Що варто тут відмітити, що такий запис з використанням знаків питання автоматично уберігає систему від sql ін’єкцій. Sql ін’єкція (SQL Injection) – це це техніка введення коду, яка може знищити вашу базу даних.. Це одна з найпоширеніших методів веб-злому. Це це розміщення шкідливого коду в операторах SQL за допомогою введення веб-сторінки.

SQL на веб-сторінках. SQL Injection зазвичай відбувається, коли ви запитуєте користувача про введення, наприклад, його ім’я користувача / ідентифікатор користувача, і замість імені / ідентифікатора користувач дає вам оператор SQL, який ви несвідомо запустите у своїй базі даних. Подивіться на наступний приклад, який створює оператор SELECT, додаючи змінну (txtUserId) до рядка select. Змінна отримується з вводу користувача (getRequestString):

txtUserId = getRequestString("UserId");

txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;

Така вбудована особливість даної бази даних SQL Lite допомагає захистити систему.

Обов’язково додається наступний рядок db.commit(), що означає, що зміни закомічені в базу даних. Використовуйте оператор COMMIT, щоб закінчити поточну транзакцію та внести постійно всі зміни, здійснені в транзакції. Транзакція - це послідовність операторів SQL, які Oracle Database розглядає як єдину одиницю. Це твердження також стирає всі точки збереження в транзакції та звільняє блокування транзакцій.

За схожим принципом реалізована реалізація команди **add_emotion**. Також використовується message_handler, оголошується функція adding_emotion з параметром message. Створюється об’єкт ReplyKeyboardMarkup з вісьмома кнопками btnx з написаними на них базоаими емоціями: anger, sadness, shame or guilty, anxiety or fear, disgust, joy, interest та love. Кнопки додаються до масива об’єкта ReplyKeyboardMarkup markup. Вводиться текст відповіді бота, зберігається повідомлення для передачі його в register_next_step_handler. Та відбувається перехід в функцію adding_emotion_to_db.

В цій функції бот повідомляє, що він додав до бази даних та саме додавання даних в базу відбувається, а саме в таблицю diary. Запис комітиться: db.commit()

4. ПЕРЕВІРКА ПРАВИЛЬНОСТІ ФУНКЦІОНУВАННЯ СИСТЕМИ

Розробники постійно помиляються. Ми можемо неправильно зрозуміти вимоги, пропустити крапку з комою, пропустити крайовий сценарій або просто забути про певні вимоги бізнесу. Трапляються помилки, і уникнути помилок практично неможливо.

Існують різні типи помилок та різні типи тестування. Тестування можуть проводити розробники, якщо воно має залучати вихідний код, спеціаліста з контролю якості для деяких спеціалізованих тестів, або може виконувати будь-хто з команди, якщо це просто виконання програмного забезпечення як користувача.

Так як розробник в нашый команды один, так само як тестер та бызнес аналітик, то роботу виконуватиме одна людина. Тому тестування буде white box. Тестування білого ящика - це коли ми перевіряємо вихідний код і переконуємось, що він працює відповідно до специфікації.

Також було проведено функцыональне тестування. Функціональне тестування використовує програмне забезпечення, щоб перевірити, чи відповідає його поведінка очікуванням. Очікування повинні бути задокументовані у вигляді технічної специфікації, яка може бути записана у вигляді документації або історії користувача. Функціональне тестування застосовується до всіх рівнів тестів від модуля до наскрізного. Якщо він відповідає на питання “що”, він перевіряє, чи повернула функція очікуване значення. [10]

Для економії ресурсу всі тестування були виконанні вручну, тобто мануальним тестуванням. Manual testing чи тестування вручну передбачає виконання ручних дій у мобільному додатку чи на веб-сайті, пошук помилок чи інших проблем із користувацьким досвідом.

Також впродовж всього тестування проводилось smoke testing. Smoke Testing - це процес тестування програмного забезпечення, що визначає стабільність розгорнутої збірки програмного забезпечення чи ні. Тестування диму є підтвердженням для команди з контролю якості для подальшого тестування програмного забезпечення. Він складається з мінімального набору тестів, що виконуються для кожної збірки для перевірки функціональних можливостей програмного забезпечення. Тестування на дим також відоме під назвою "Тестування перевірки конструкції" або "Тестування впевненості".

Говорячи простими словами, ми перевіряємо, чи працюють важливі функції, і немає демонстраторів у збірці, яка перевіряється. Це міні-і швидкий регресійний тест основних функціональних можливостей. Це простий тест, який показує, що продукт готовий до тестування. Це допомагає визначити, чи є збірка хибною, що робить подальше тестування марною тратою часу та ресурсів.[10]

Виконаємо ручне тестування команд доступних боту, перевірим правильність додавання в базу та проаналізуємо поведінку бота.

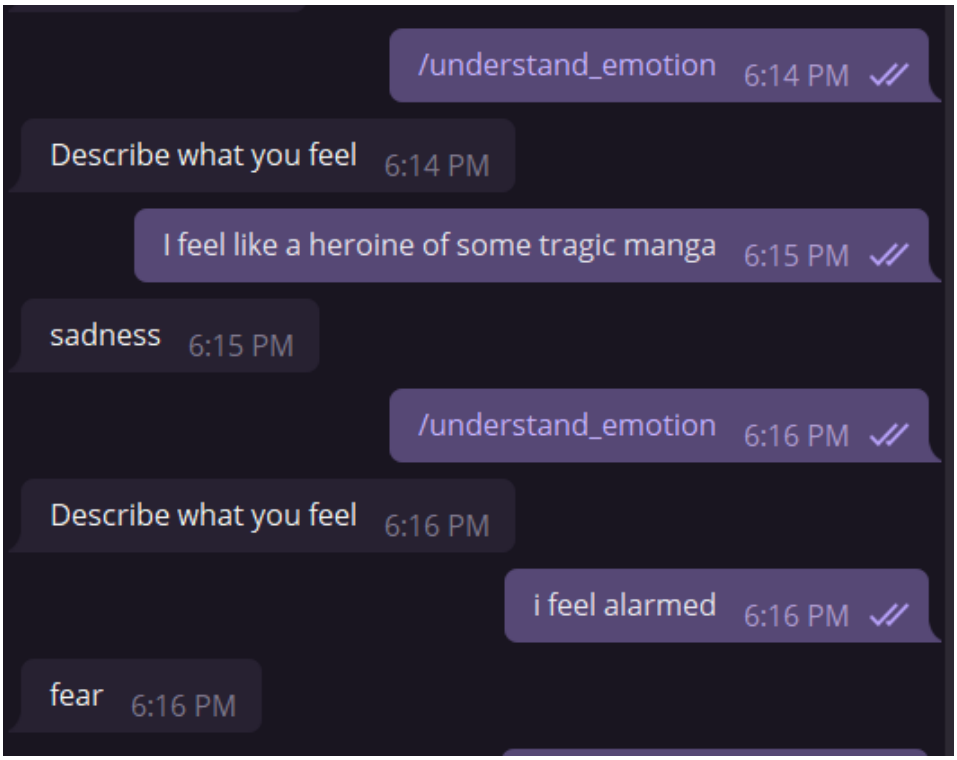

Рисунок 9 – Робота команди understand_emotion

Тестування команди understand_emotion. Для виконання цього тесту надсилаємо команду боту / understand_emotion. Бот пропонує описати відчуття користувача, користувач надсилає свої думки і роб виконує обробку даних даних та дає припустиму відповідь, того, що це за емоція.

Тестування команди add_emotion. Для тестування команди, надсилаємо команду боту в чат /add_emotion, бот запропонує вибрати емоцію зі списка реплай клавіатури. Вибираємо, натимкаємо. При цьому клавіатура зникає і бот повідомляє, що додав відповідний запис.

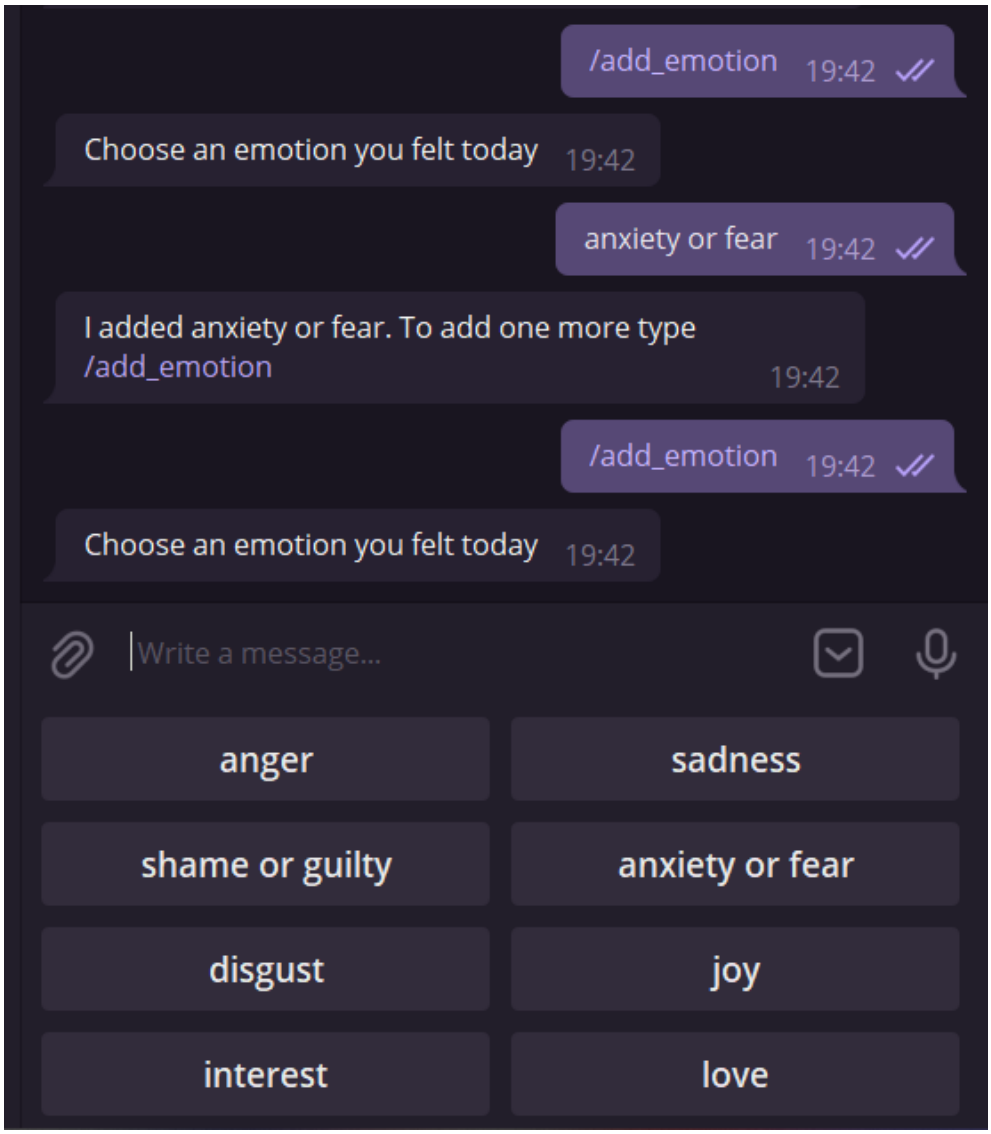

Рисунок 10 – Робота команди add_emotion

Тестування команди evaluete_day. Дана команда схожа на попередню add_emotion, користувач відправляє команду, бот пропонує оцінити день та виводить реплай клавіатуру, користувач вводить дані та відправляє. Бот повідомляє про те, що дані додані. Команда працює коректно.

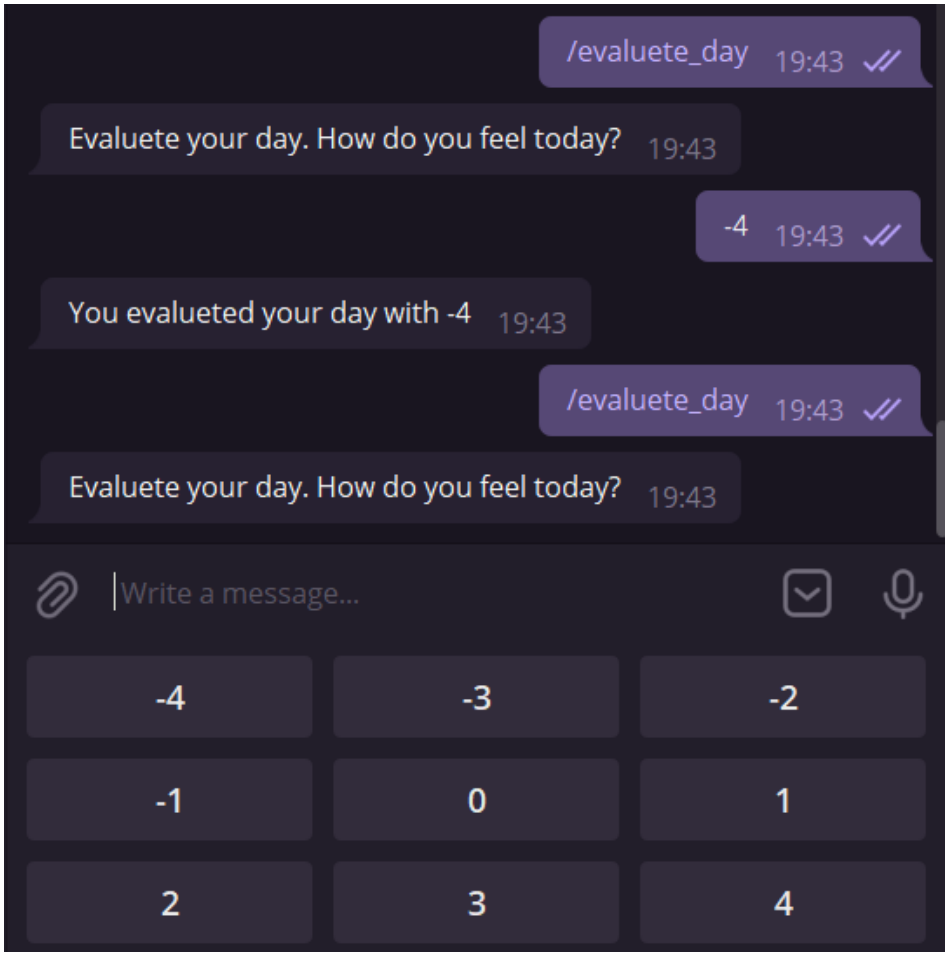

Рисунок 11 – Робота команди evaluete_day

Тестування команди advise. Відправляємо відповідну команду боту та після короткого проміжку часу бот відправляє користувачу ресурс з корисною інформацією, тобто отримуємо корисне джерело.

Так само проводимо тестуваня для команди support, яка працюэ за схожим принципом, выдправляжио команду боту / support та отримуэмо у відповідь зображення, або одну з перелічених цитат: «I couldn't heal because I kept pretending I wasn't hurt», «Sleep just isn't sleep anymore, it's an escape», «No

one knows how I cried that day». Виконання команд приведено на рисунках нижче.

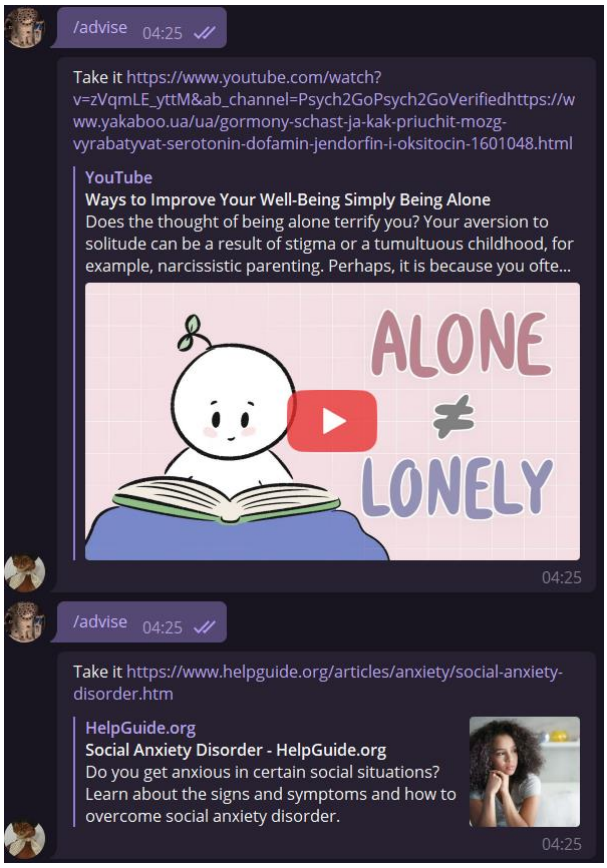

Рисунок 12 – Робота команди advise

Зображення чи цитата у відповідь приходять з рівною вірогідністю.

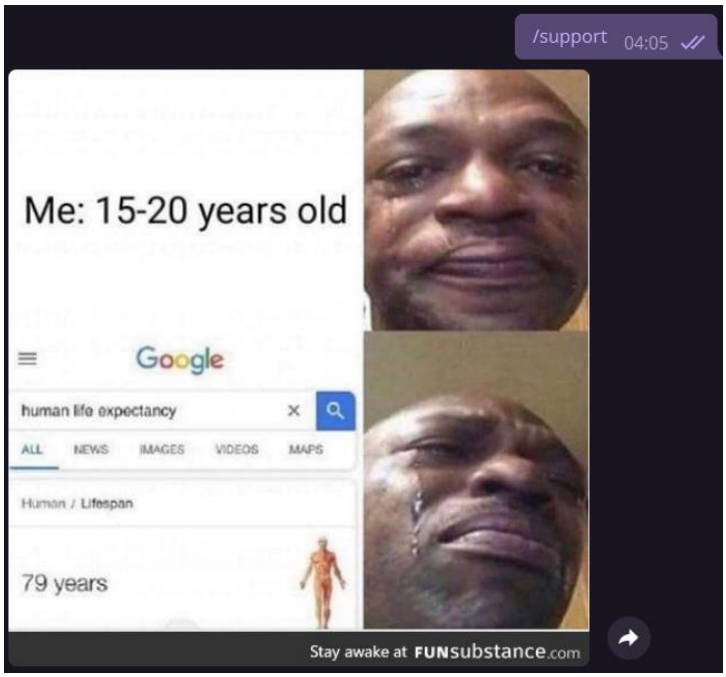

Рисунок 13 – Робота команди support

Приклад виконання тесту коли користувач отримує цитату.

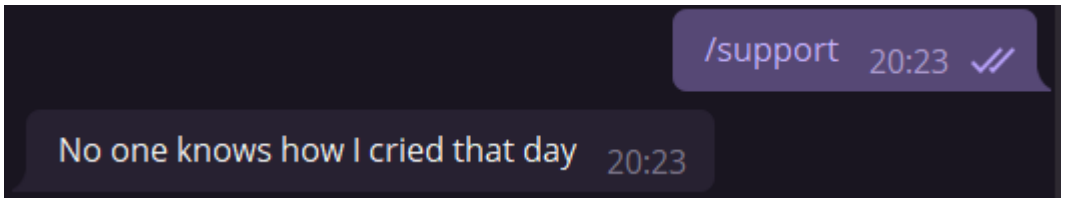

Рисунок 14 - Робота команди support

Тестування команди specialists. На даному етапі розробки бота, при невеликій тестові базі спеціалістів (психологів і терапевтів) неможливо надати релевантні дані реальних людей, тому використовуємо вигадані.

Команда працює за схрдим принципом з попередніми, а саме вибирає рандомного спеціаліста, але на відміну від команд support та advise вибирає дані не з масиву (list python), а з бази даних SQL Lite, назва бази даних bot_phyco.db, з таблиці specialists.

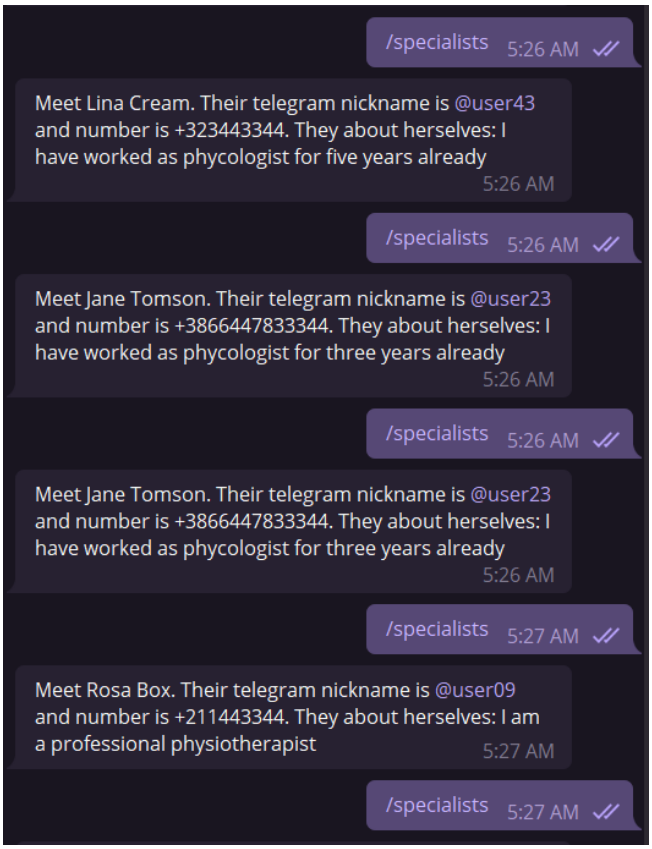

Рисунок 15 – Перевірка команди specialists

Також необхідно перевірити чи зберігаються дані в базі даних bot_phyco.db вірно. Для цього необхідно виконати селект за базм даних. Для цього ми створили Python скрипт для тестування бази даних, стороній від коду основного проекту. Це дозволить перевіряти більш простіше та ефективніше дані та роботу бази даних. Створюємо скрипт db.py, оголошуэмо об’єкт конектор до бази даних db = sqlite3.connect('bot_phyco.db') та об’єкт курсор, який дані буде отримувати - sql = db.cursor()

Використаэмо команду SQL SELECT. Оператор SELECT використовується для вибору даних із бази даних. Повернуті дані зберігаються в таблиці результатів, яка називається набором результатів. Нас цікавлять таблиці diary та users.

```
for value in sql.execute("select * from diary"):
    print(value)
for value in sql.execute("select * from users"):
    print(value)
```

Рисунок 16 – Витяг з таблиці

Дані таблиці виглядають наступним чином для таблиці diary:

('2021-06-12', -4, 'anxiety or fear', 7)

('2021-06-12', 0, 'joy', 9)

Для таблиці users:

(7, 386044539)

(9, 438403060)

З даних бачимо, що ботом користувалися 2 користувача. З яких обидва, оцынили свій день та додали емоцію.

ВИСНОВОК

Ментальне здоров’я вже стоїть на рівні з фізичним в більшості високорозвинутих країнах. І цим проектом був зроблений крок в збільшення рівня благополуччя людської психіки.

В результаті виконання даної роботи було створено Телеграм бот асистент психолог. Який володіє функціоналом, що може допомогти під час виконання наступних психічно терапевтичними методів:

- Відстеження кореляції загального денного стану до тригерів, що на нього впливають
- Відстеження емоційного фону

Також було додано можливість пошуку спеціаліста для еористувачів.

Дана інформаційна система використовує в собі сучасні технології Pythong, Telegram API та SQL Lite для свого функціонування.

Була розроблена модель розпізнавання емоцій з використанням методів машиного навчання та штучного інтелекту. А саме розроблена система на трансформера BERT, з використанням модуля TensorFlow та transformers.

Система має потенціальний спосіб заробітку: показуючи релевантну рекламу та за допомогою бази спеціалістів.

В дальнійшому інформаційна система буде покращуватися та нарощувати свій функціонал.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Mental Health Is Essential [Електронний ресурс] – 2021. - Доступно: https://choicespsychotherapy.net/mental-health-is-essential/ Дата звернення: 31 травня 2021
2. Google Play Store [Електронний ресурс] – 2021. - Доступно:: https://play.google.com/store Дата звернення: 31 травня 2021
3. Unified Modeling Language [Електронний ресурс] – 2021. - Доступно:: https://www.tutorialspoint.com/index.htm Дата звернення: 31 травня 2021
4. Оцінка рівня ситуативної (реактивної) тривожності (Тест Спілбергера-Ханіна) [Електронний ресурс] – 2019. - Доступно:: https://www.lnu.edu.ua/life-safety/wp-content/uploads/2019/09/OZDSH_PR-4-2019.pdf Дата звернення: 1 червня 2021
5. Depression [Електронний ресурс] – 2019. - https://www.who.int/news-room/fact-sheets/detail/depression Дата звернення: 1 червня 2021
6. Anxiety disorders [Електронний ресурс] – 2020. - Доступно: https://www.mayoclinic.org/diseases-conditions/anxiety/symptoms-causes/syc-20350961 Дата звернення: 1 червня 2021
7. Internet memes could help people with depression new study finds [Електронний ресурс] – 2020. - Доступно: https://blogs.shu.ac.uk/hwbstaffnews/2020/01/29/internet-memes-could-help-people-with-depression-new-study-finds/?doing_wp_cron=1609774349.9593820571899414062500 Дата звернення: 1 червня 2021

8. Entity-Relationship Diagram Definition [Електронний ресурс] – 2020. – Доступно: https://www.lifewire.com/entity-relationship-diagram-1019253 Дата звернення: 1 червня 2021
9. TelegramBot. Базовий функціонал. Мухи окремо, котлети окремо. [Електронний ресурс] – 2020. – Доступно: https://habr.com/ru/post/481354/ Дата звернення: 1 червня 2021
10. Understanding software testing[Електронний ресурс] – 2018. - Доступно https://medium.com/@netxm/how-to-get-started-with-software-testing-9fa1ce4f2a64 Дата звернення: 1 червня 2021
11. About SQL Lite [Електронний ресурс] – 2019. - Доступно: https://www.sqlite.org/about.html Дата звернення: 1 червня 2021
12. sqlite3 — DB-API 2.0 interface for SQLite databases [Електронний ресурс] – 2021. – Доступно: https://docs.python.org/3/library/sqlite3.html# Дата звернення: 1 червня 2021
13. What Is Natural Language Processing? [Електронний ресурс] – 2017. – Режим доступу: https://machinelearningmastery.com/natural-language-processing/

Додаток А

```
#bertemomod/py

from transformers import BertTokenizer

import emotion_model


def bert_answer(bert_in):

    #give to model

    return emotion_model.guess(bert_in)



def tokenize(sentence):

    tokenizer    =    BertTokenizer.from_pretrained('bert-base-uncased',
do_lower_case=True)

    max_length_test = 50

    test_sentence = sentence

    # add special tokens

    test_sentence_with_special_tokens  =  '[CLS]'  +  test_sentence  +
'[SEP]'

    tokenized = tokenizer.tokenize(test_sentence_with_special_tokens)

    print('tokenized', tokenized)


    # convert tokens to ids in WordPiece

    input_ids = tokenizer.convert_tokens_to_ids(tokenized)


    # precalculation of pad length, so that we can reuse it later on

    padding_length = max_length_test - len(input_ids)


    # map tokens to WordPiece dictionary and add pad token for those
text shorter than our max length

    input_ids = input_ids + ([0] * padding_length)
```

```
            # attention should focus just on sequence with non padded tokens
            attention_mask = [1] * len(input_ids)

            # do not focus attention on padded tokens
            attention_mask = attention_mask + ([0] * padding_length)

            # token types, needed for example for question answering, for our purpose we will just set 0 as we have just one sequence
            token_type_ids = [0] * max_length_test
            bert_input = {
                "token_ids": input_ids,
                "token_type_ids": token_type_ids,
                "attention_mask": attention_mask
            }

            print("bert_input")
            print(bert_input)

            bert_input = tokenizer.encode_plus(
                                test_sentence,
                                add_special_tokens = True, # add [CLS], [SEP]
                                max_length = max_length_test, # max length of the text that can go to BERT
                                pad_to_max_length = True, # add [PAD] tokens
                                return_attention_mask = True, # add attention mask to not focus on pad tokens
                           )
            print('encoded', bert_input)

            return bert_answer(bert_input)
```

```
#bot.py
import telebot
from telebot import types
import configparser
import random
import datetime
import sqlite3
import bertemomod as emo
import img_meme
import helpful_source
import test_beck_depression
#import test_anxiety

config = configparser.ConfigParser()
config.read("config.ini")

#оголошення бота
bot = telebot.TeleBot(config["bot"]["token"])

#з'єднання з базою
db = sqlite3.connect('bot_phyco.db', check_same_thread=False)
sql = db.cursor()


#команда старт
@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
    sql.execute("INSERT INTO users(chat_id) VALUES(?)",
(message.chat.id,))
    db.commit()
```

```
            answ =  "Hello, " + message.from_user.first_name + "I can help you to manage your emotional diary. Watch commands down below"
            bot.send_message(message.chat.id, answ)


        #відправка корисних посилань і подібного
        @bot.message_handler(commands=['advise'])
        def send_helpful_sources(message):
            bot.send_message(message.chat.id,      "Take      it      "      + helpful_source.get_advice())


        #відправка картинок і цитат
        @bot.message_handler(commands=['support'])
        def send_helpful_sources(message):
            type = random.randint(1,3)
            if type < 3:
                  url = img_meme.get_img_meme()
                  bot.send_photo(message.chat.id, url)
            else:
                  num = random.randint(1,5)
                  if num == 1:
                        bot.send_message(message.chat.id,  "I  couldn't  heal because I kept pretending I wasn't hurt")
                  elif num == 2:
                        bot.send_message(message.chat.id,  "Sleep  just  isn't sleep anymore, it's an escape")
                  else:
                        bot.send_message(message.chat.id, "No one knows how I cried that day")


        #оцінка дня
        @bot.message_handler(commands=['evaluete_day'])
```

```
        def evalueting_day(message):
            markup = types.ReplyKeyboardMarkup(resize_keyboard = True, row_width=3, one_time_keyboard=True)
            btn1 = types.KeyboardButton("-4")
            btn2 = types.KeyboardButton("-3")
            btn3 = types.KeyboardButton("-2")
            btn4 = types.KeyboardButton("-1")
            btn5 = types.KeyboardButton("0")
            btn6 = types.KeyboardButton("1")
            btn7 = types.KeyboardButton("2")
            btn8 = types.KeyboardButton("3")
            btn9 = types.KeyboardButton("4")
            markup.add(btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9)
            answ = "Evaluete your day. How do you feel today?"
            msg = bot.send_message(message.chat.id, answ, parse_mode='html', reply_markup=markup)
            bot.register_next_step_handler(msg, adding_mark_to_db)


        def adding_mark_to_db(message):
            bot.send_message(message.chat.id, "You evalueted your day with " + message.text)
            # тут ложим оцінку в database
            tday = datetime.date.today()
            sql.execute("INSERT INTO diary(date, mark, user) VALUES (?, ?, (SELECT id_user FROM users WHERE chat_id = ?))", (tday, int(message.text), message.chat.id, ))
            db.commit()



        #пройти тест функціонал
```

```
@bot.message_handler(commands=['take_test'])
def taking_test(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard = True, row_width=2, one_time_keyboard=True)
    btn1 = types.KeyboardButton("Beck's Depression Inventory")
    btn2 = types.KeyboardButton("State Trait Anxiety Inventory")
    markup.add(btn1, btn2)
    answ = "Choose which test you want to take"
    msg = bot.send_message(message.chat.id, answ, parse_mode='html', reply_markup=markup)
    bot.register_next_step_handler(msg, which_test)


def which_test(message):
    if message.text == "Beck's Depression Inventory":
        markup = types.ReplyKeyboardMarkup(resize_keyboard = True, row_width=1)
        msg = bot.send_message(message.chat.id, "You chose Beck's Depression Inventory. Let's start", parse_mode='html', reply_markup = markup.add(types.KeyboardButton("OK")))
        bot.register_next_step_handler(msg, taking_test_beck_depression)

    elif message.text == "State Trait Anxiety Inventory":
        bot.send_message(message.chat.id, "You chose State Trait Anxiety Inventory. Let's start")
    else:
        bot.send_message(message.chat.id, "There is no test with that name")


def taking_test_beck_depression(message):
    score = 0
```

```
            amount = len(test_beck_depression.answ_list)
            i = 0
            while i<amount:
                  question = str(i+1) + ". Choose what describes you best"
                  bot.send_message(message.chat.id, question)
                  i=i+1


      #Додати емоцію до щоденика
      @bot.message_handler(commands=['add_emotion'])
      def adding_emotion(message):
            markup = types.ReplyKeyboardMarkup(resize_keyboard = True,
row_width=2, one_time_keyboard=True)
            btn1 = types.KeyboardButton("anger")
            btn2 = types.KeyboardButton("sadness")
            btn3 = types.KeyboardButton("shame or guilty")
            btn4 = types.KeyboardButton("anxiety or fear")
            btn5 = types.KeyboardButton("disgust")
            btn6 = types.KeyboardButton("joy")
            btn7 = types.KeyboardButton("interest")
            btn8 = types.KeyboardButton("love")
            markup.add(btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8)
            answ = "Choose an emotion you felt today"
            msg = bot.send_message(message.chat.id, answ, parse_mode='html',
reply_markup=markup)
            bot.register_next_step_handler(msg, adding_emotion_to_db)


      def adding_emotion_to_db(message):
            bot.send_message(message.chat.id, "I added " + message.text + ". To
add one more type /add_emotion")
            # тут ложим емоцію в бд
            tday = datetime.date.today()
```

```
#helpful_source.py

import random


url_list = ["https://www.youtube.com/channel/UCc7te9kr2fOVkkFay1ntCQQ",

            "https://www.youtube.com/watch?v=hgGI-
75FMCA&list=PLIjqXVLsr53Hss0aD-uwX4GFZGfadePvR",


"https://www.youtube.com/watch?v=opFb4ei918k&list=PLIjqXVLsr53EsD_QkIx0qk0dg8
ilXnyCe&ab_channel=%D0%95%D0%B2%D0%B3%D0%B5%D0%BD%D0%B8%D1%8F%D0%A1%D1%82%D1%
80%D0%B5%D0%BB%D0%B5%D1%86%D0%BA%D0%B0%D1%8F%D0%95%D0%B2%D0%B3%D0%B5%D0%BD%D0
%B8%D1%8F%D0%A1%D1%82%D1%80%D0%B5%D0%BB%D0%B5%D1%86%D0%BA%D0%B0%D1%8FVerified
",

            "https://www.youtube.com/watch?v=5Yz-
Bvm5tRg&list=PLIjqXVLsr53HmOvyAkEwNpvsVv-
oZ0k6D&ab_channel=%D0%95%D0%B2%D0%B3%D0%B5%D0%BD%D0%B8%D1%8F%D0%A1%D1%82%D1%8
0%D0%B5%D0%BB%D0%B5%D1%86%D0%BA%D0%B0%D1%8F%D0%95%D0%B2%D0%B3%D0%B5%D0%BD%D0%
B8%D1%8F%D0%A1%D1%82%D1%80%D0%B5%D0%BB%D0%B5%D1%86%D0%BA%D0%B0%D1%8FVerified"
,

"https://www.youtube.com/watch?v=o3rGwIOvUZQ&ab_channel=Psych2GoPsych2GoVerif
ied",

"https://www.youtube.com/watch?v=hXlFxceM4R8&ab_channel=Psych2GoPsych2GoVerif
ied",

"https://www.youtube.com/watch?v=zVqmLE_yttM&ab_channel=Psych2GoPsych2GoVerif
ied"
            "https://www.yakaboo.ua/ua/gormony-schast-ja-kak-priuchit-
mozg-vyrabatyvat-serotonin-dofamin-jendorfin-i-oksitocin-1601048.html",
            "https://www.helpguide.org/articles/anxiety/social-
anxiety-disorder.htm",
            "https://youtu.be/wERgEDqRTuY",
            "https://youtu.be/BqtweDhvG50",]



def get_advice():
```

```
            num = random.randint(0,len(url_list)-1)
            return url_list[num]
        #img_meme.py
        import random

        url_list =
["https://i.pinimg.com/564x/b6/25/c3/b625c31d2396305d6fbed05bb181cf6a.jpg",

"https://i.pinimg.com/564x/95/e7/e5/95e7e595ef8284edf99c7709c18ba09f.jpg",

"https://www.redbubble.com/people/binchcity/works/33688615-girls-just-wanna-
have-
serotonin?finish=semi_gloss&p=poster&size=small&gdffi=aa308446f0204ceb814e04f
9c214a536&gdfms=A603846FF62641D8A5580FD407BB6848",

"https://i.pinimg.com/564x/11/2c/68/112c682388a7484fd925cc59404985f9.jpghttps
://i.pinimg.com/564x/11/2c/68/112c682388a7484fd925cc59404985f9.jpg",

"https://i.pinimg.com/564x/ce/ef/c2/ceefc270f00f681edb8e67b1d89061ab.jpg",

"https://i.pinimg.com/564x/b8/aa/a6/b8aaa6ab048fcad6bca0fdbfcc244d66.jpg",

"https://i.pinimg.com/564x/73/6c/95/736c9556fbfd62d5a324a5cfb91f3b53.jpg",

"https://i.pinimg.com/564x/22/bc/13/22bc139f92a8ab02313c23f4b2c12dda.jpg",

"https://i.pinimg.com/564x/e8/b8/ad/e8b8ad3ab670dea1200aa5a5330be215.jpg",

"https://i.pinimg.com/564x/0c/48/51/0c4851dcb32f766b450a94164eca4f70.jpg",

"https://i.pinimg.com/564x/be/2a/80/be2a801cc9eefe90360d336cf3c317dc.jpg",

"https://i.pinimg.com/564x/e2/5b/20/e25b20489889af6b333273b3c5145f7a.jpg",

"https://i.pinimg.com/236x/b3/68/65/b36865e361a76bd662a526cda907dcf5.jpg",
```

```
"https://i.pinimg.com/564x/59/07/b8/5907b83541fd2325b3909ef9e4ea03e5.jpg",

"https://i.pinimg.com/564x/b5/52/42/b55242e2ec3741f5fafa85f4f1e88872.jpg",

"https://i.pinimg.com/564x/09/9e/7c/099e7ca92f0231f704b0547fd58b6acf.jpg",

"https://i.pinimg.com/564x/c0/3e/c9/c03ec9e83c573e9ff38c7b6ea4182810.jpg",

"https://i.pinimg.com/564x/9a/24/3e/9a243e2d009d1fd32037dbb77046f4d9.jpg",

"https://i.pinimg.com/564x/aa/4a/49/aa4a49101fb93c87d6ab0ce28113a1d8.jpg",

"https://i.pinimg.com/564x/93/f2/dd/93f2dda20459d28f8f5d751034f035cc.jpg"]


        def get_img_meme():
            num = random.randint(0,len(url_list)-1)
            return url_list[num]
        # emotion_model.py
        # data analysis and manipulation package
        import pandas as pd
        import os
        import numpy as np
        from tqdm import tqdm, trange


        import torch
        import torch.nn as nn
        import torch.nn.functional as F
        import torch.optim as optim
        from torch.utils.data import Dataset, DataLoader
        from transformers import BertPreTrainedModel, BertModel
        from transformers import AutoConfig, AutoTokenizer
```

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report

# read a comma-separated values (csv) file into DataFrame.
train_df = pd.read_csv('../../data_set/train.txt', sep=';')
test_df = pd.read_csv('../../data_set/test.txt', sep=';')
val_df = pd.read_csv('../../data_set/val.txt', sep=';')

# return a tuple representing the dimensionality of the DataFrame.
print(train_df.shape, test_df.shape, val_df.shape)

# the column labels of the DataFrame.
train_df.columns = ['sentence', 'emotion']
test_df.columns = ['sentence', 'emotion']
val_df.columns = ['sentence', 'emotion']

# return the first n rows of data frame
print(train_df.head(n = 5))

# count emotion values
print(train_df['emotion'].value_counts())
print(test_df['emotion'].value_counts())
print(val_df['emotion'].value_counts())

# get max len of sentenses
def max_len(data):
    return data['sentence'].apply(lambda x: len(x.split())).max()
max_lens = [max_len(train_df), max_len(test_df), max_len(val_df)]
```

```
        print(max_lens)
        print(max(max_lens))

        # Configs
        MODEL_OUT_DIR = '../bert_emotion'
        TRAIN_FILE_PATH = '../../data_set/train.txt'
        VALID_FILE_PATH = '../../data_set/val.txt'
        TEST_FILE_PATH = '../../data_set/test.txt'
        ## Model Configurations
        MAX_LEN_TRAIN = 68
        MAX_LEN_VALID = 68
        MAX_LEN_TEST = 68
        BATCH_SIZE = 160
        # recommended learning rate for Adam 5e-5, 3e-5, 2e-5
        # learning_rate
        learning_rate = 1e-5
        # we do 10 epoch
        # though multiple epochs might be better as long
        # as we will not overfit the model
        number_of_epochs = 5
        NUM_THREADS = 1  ## Number of threads for collecting dataset
        MODEL_NAME = 'bert-base-uncased'
        # dictionary of labels
        LABEL_DICT  =  {'joy':0,  'sadness':1,  'anger':2,  'fear':3,  'love':4,
'surprise':5}


        # Create Dataset
        class Emotions_Dataset(Dataset):

            def __init__(self, filename, maxlen, tokenizer, label_dict):
```

```
                #Store the contents of the file in a pandas dataframe
                self.df = pd.read_csv(filename, delimiter = ';')
                # name columns
                self.df.columns = ['sentence', 'emotion']
                #Initialize the tokenizer for the desired transformer model
                self.df['emotion'] = self.df['emotion'].map(label_dict)
                self.tokenizer = tokenizer
                #Maximum length of the tokens list to keep all the sequences of
fixed size
                self.maxlen = maxlen


            def __len__(self):
                return len(self.df)


            def __getitem__(self, index):
                #Select the sentence and label at the specified index in the
data frame
                sentence = self.df.loc[index, 'sentence']
                label = self.df.loc[index, 'emotion']
                #Preprocess the text to be suitable for the transformer
                tokens = self.tokenizer.tokenize(sentence)
                tokens = ['[CLS]'] + tokens + ['[SEP]']
                if len(tokens) < self.maxlen:
                    tokens = tokens + ['[PAD]' for _ in range(self.maxlen -
len(tokens))]
                else:
                    tokens = tokens[:self.maxlen-1] + ['[SEP]']
                #Obtain the indices of the tokens in the BERT Vocabulary
                input_ids = self.tokenizer.convert_tokens_to_ids(tokens)
                input_ids = torch.tensor(input_ids)
                #Obtain the attention mask i.e a tensor containing 1s for no
padded tokens and 0s for padded ones
```

```
            attention_mask = (input_ids != 0).long()


            label = torch.tensor(label, dtype=torch.long)


            return input_ids, attention_mask, label


    class BertEmotionClassifier(BertPreTrainedModel):
        def __init__(self, config):
            super().__init__(config)
            self.bert = BertModel(config)
            #The classification layer that takes the [CLS] representation
and outputs the logit
            self.cls_layer = nn.Linear(config.hidden_size, 6)


        def forward(self, input_ids, attention_mask):
            #Feed the input to Bert model to obtain contextualized
representations
            reps, _ = self.bert(input_ids=input_ids,
attention_mask=attention_mask)
            #Obtain the representations of [CLS] heads
            cls_reps = reps[:, 0]
            logits = self.cls_layer(cls_reps)
            return logits



    # Training function
    def train(model, criterion, optimizer, train_loader, val_loader, epochs,
device):
        best_acc = 0
        for epoch in trange(epochs, desc="Epoch"):
            model.train()
            train_acc = 0
```

```
            for i, (input_ids, attention_mask, labels) in
enumerate(iterable=train_loader):
                optimizer.zero_grad()

                input_ids, attention_mask, labels = input_ids.to(device),
attention_mask.to(device), labels.to(device)

                logits = model(input_ids=input_ids,
attention_mask=attention_mask)

                loss = criterion(logits, labels)
                loss.backward()
                optimizer.step()

                train_acc += get_accuracy_from_logits(logits, labels)

            print(f"Training accuracy is {train_acc/len(train_loader)}")
            val_acc, val_loss = evaluate(model=model, criterion=criterion,
dataloader=val_loader, device=device)
            print("Epoch {} complete! Validation Accuracy : {}, Validation
Loss : {}".format(epoch, val_acc, val_loss))



    # Evaluation function
    def evaluate(model, criterion, dataloader, device):
        model.eval()
        mean_acc, mean_loss, count = 0, 0, 0
    #     predicted_labels = []
    #     actual_labels = []
        with torch.no_grad():
            for input_ids, attention_mask, labels in (dataloader):
```

```
                    input_ids, attention_mask, labels = input_ids.to(device),
attention_mask.to(device), labels.to(device)
                    logits = model(input_ids, attention_mask)

                    mean_loss += criterion(logits.squeeze(-1), labels).item()
                    mean_acc += get_accuracy_from_logits(logits, labels)
                    count += 1

        #             predicted_labels += output
        #             actual_labels += labels

            return mean_acc/count, mean_loss/count

        def get_accuracy_from_logits(logits, labels):
            probs = F.softmax(logits, dim=1)
            output = torch.argmax(probs, dim=1)
            acc = (output == labels).float().mean()
            return acc



        # Predict function
        def predict(model, dataloader, device):
            predicted_label = []
            actual_label = []
            with torch.no_grad():
                for input_ids, attention_mask, labels in (dataloader):

                    input_ids, attention_mask, labels = input_ids.to(device),
attention_mask.to(device), labels.to(device)
                    logits = model(input_ids, attention_mask)
```

```
                probs = F.softmax(logits, dim=1)
                output = torch.argmax(probs, dim=1)

                predicted_label += output
                actual_label += labels

        return predicted_label, actual_label



    ## Configuration loaded from AutoConfig
    config = AutoConfig.from_pretrained(MODEL_NAME)
    ## Tokenizer loaded from AutoTokenizer
    tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
    ## Creating the model from the desired transformer model
    model = BertEmotionClassifier.from_pretrained(MODEL_NAME,
config=config)
    ## GPU or CPU
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    ## Putting model to device
    model = model.to(device)
    ## Takes as the input the logits of the positive class and computes the
binary cross-entropy
    # criterion = nn.BCEWithLogitsLoss()
    criterion = nn.CrossEntropyLoss()
    ## Optimizer
    optimizer = optim.Adam(params=model.parameters(), lr=learning_rate)



    ## Training Dataset
```

```
        train_set          =          Emotions_Dataset(filename=TRAIN_FILE_PATH,
maxlen=MAX_LEN_TRAIN, tokenizer=tokenizer, label_dict=LABEL_DICT)

        valid_set          =          Emotions_Dataset(filename=VALID_FILE_PATH,
maxlen=MAX_LEN_VALID, tokenizer=tokenizer, label_dict=LABEL_DICT)

        test_set           =           Emotions_Dataset(filename=TEST_FILE_PATH,
maxlen=MAX_LEN_TEST, tokenizer=tokenizer, label_dict=LABEL_DICT)



        ## Data Loaders

        train_loader   =   DataLoader(dataset=train_set,   batch_size=BATCH_SIZE,
num_workers=NUM_THREADS)

        valid_loader   =   DataLoader(dataset=valid_set,   batch_size=BATCH_SIZE,
num_workers=NUM_THREADS)

        test_loader    =    DataLoader(dataset=test_set,    batch_size=BATCH_SIZE,
num_workers=NUM_THREADS)


        # print(len(train_loader))



        train(model=model,

              criterion=criterion,

              optimizer=optimizer,

              train_loader=train_loader,

              val_loader=valid_loader,

              epochs = 5,

             device = device)


        actual_label,    predicted_label    =    predict(model,    test_loader,
device=device)

        actual_label = np.array([item.to('cpu') for item in actual_label])

        predicted_label = np.array([item.to('cpu') for item in predicted_label])
```

```
        print("Accuracy                :",metrics.accuracy_score(actual_label,
predicted_label))

        print("f1      score      macro      :",metrics.f1_score(actual_label,
predicted_label, average = 'macro'))

        print("f1      scoore      micro      :",metrics.f1_score(actual_label,
predicted_label, average = 'micro'))

        print("Hamming        loss        :",metrics.hamming_loss(actual_label,
predicted_label))

        print("Classification Report: \n", classification_report(actual_label,
predicted_label))

        print("Confusion     Matrix:     \n",     confusion_matrix(actual_label,
predicted_label))
```