МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ТИМЧУК Роман Ігорович

Математичне та програмне забезпечення для формування задач з тексту електронного листа у JIRA / Mathematical Tools and Software for Tasks Generating from Text in Email under JIRA

спеціальність: 121 - Інженерія програмного забезпечення
освітньо-професійна програма - Інженерія програмного забезпечення

Кваліфікаційна робота

Виконав студент групи ІПЗм-21
Р. І. Тимчук

_____

Науковий керівник:
к.т.н., доцент І. С. Стасів

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2022

ВСТУП

Актуальність теми. В сучасних реаліях, є актуальною автоматизація будь-яких можливих процесів. Автоматизація означає підвищення продуктивності, вона дозволяє працівникам уникати рутинної роботи, і приділяти основний фокус більш важливим проблемам, що потребують вирішення.

Мета і задачі дослідження є вирішення проблеми автоматизації створення задач в Jira на основі електронного листа за допомогою штучного інтелекту та розробка відповідного програмного забезпечення.

Об'єктом дослідження є автоматизація процесу створення задач в Jira на основі аналізу вмісту електронних повідомлень.

Предмет дослідження: програмне забезпечення для аналізу вмісту електронних листів та взаємодії з API Jira для створення задач.

Методи дослідження: теоретичний аналіз, дослідження великих вибірок даних для тренування штучного інтелекту; алгоритмізації процесів з використанням сучасних пакетів прикладних програм.

Науковою новизною отриманих результатів є вирішення проблеми подібної автоматизації з використанням тренованої нейромережі. При виконанні вручну, така монотонна робота може використати велику кількість людських ресурсів. Отож, виникає необхідність в автоматизації такого процесу. Для цього є два очевидних способи:

- Аналіз листа з вказанням чіткої структури для тексту.
- Використання штучного інтелекту для аналізу.

Практичне застосування такого штучного інтелекту полягає в отриманні постановки задачі з тексту електронного листа, нейромережа реалізовуватиме “лейблінг” тексту: вона визначатиме ключові слова, фрази та відрізки тексту, які відносяться до конкретних полів в задачі Jira, наприклад

summary, reporter, due date, assignee та інші. Такий метод дозволяє чітко розбити вміст листа на необхідні модулі, і створити з них постановку задачі.

Після машинного навчання на основі великої вибірки даних, нейромережа зможе аналізувати листи, довжиною до 800 символів, визначати з них основні поля для задачі Jira, та повертати готовий об’єкт, який одразу ж можна використовувати при створенні задачі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Система Jira та мікросервіси

Jira — це інструмент менеджменту, який для продуктивного керування проектами, один з інструментів від австралійської компанії Atlassian. Створена в далекому 2002 році, Jira дуже швидко стала популярним інструментом в провідних компаніях.

В момент запуску, цільовими користувачами Jira були в основному розробники програмного забезпечення. Сьогодні, використання цього інструменту дає необмежені можливості для команд різних типів, як-от відділ кадрів, юрист, IT, освіта, охорона здоров'я, медіа та зв'язок. Переломний момент, який дозволив цьому статися, відбувся о 2012 році, з відкриттям Jira Marketplace.

Маркетплейс надав розробникам можливість створювати користувацькі плагіни для взаємодії з основною системою Jira. Сьогодні, на маркетплейсі можна знайти тисячі різноманітних плагінів(адонів), які покращують роботу користувачів з Jira, доповнюють а також розширюють її функціонал.

Jira надає розробникам плагінів великий набір інструментів для їх створення.Однією з найновіших платформа розробки плагінів є Forge. Forge створена командами для команд і дає можливість створювати надійні та масштабовані програмні системи за допомогою інфраструктури, розміщеної в Atlassian.

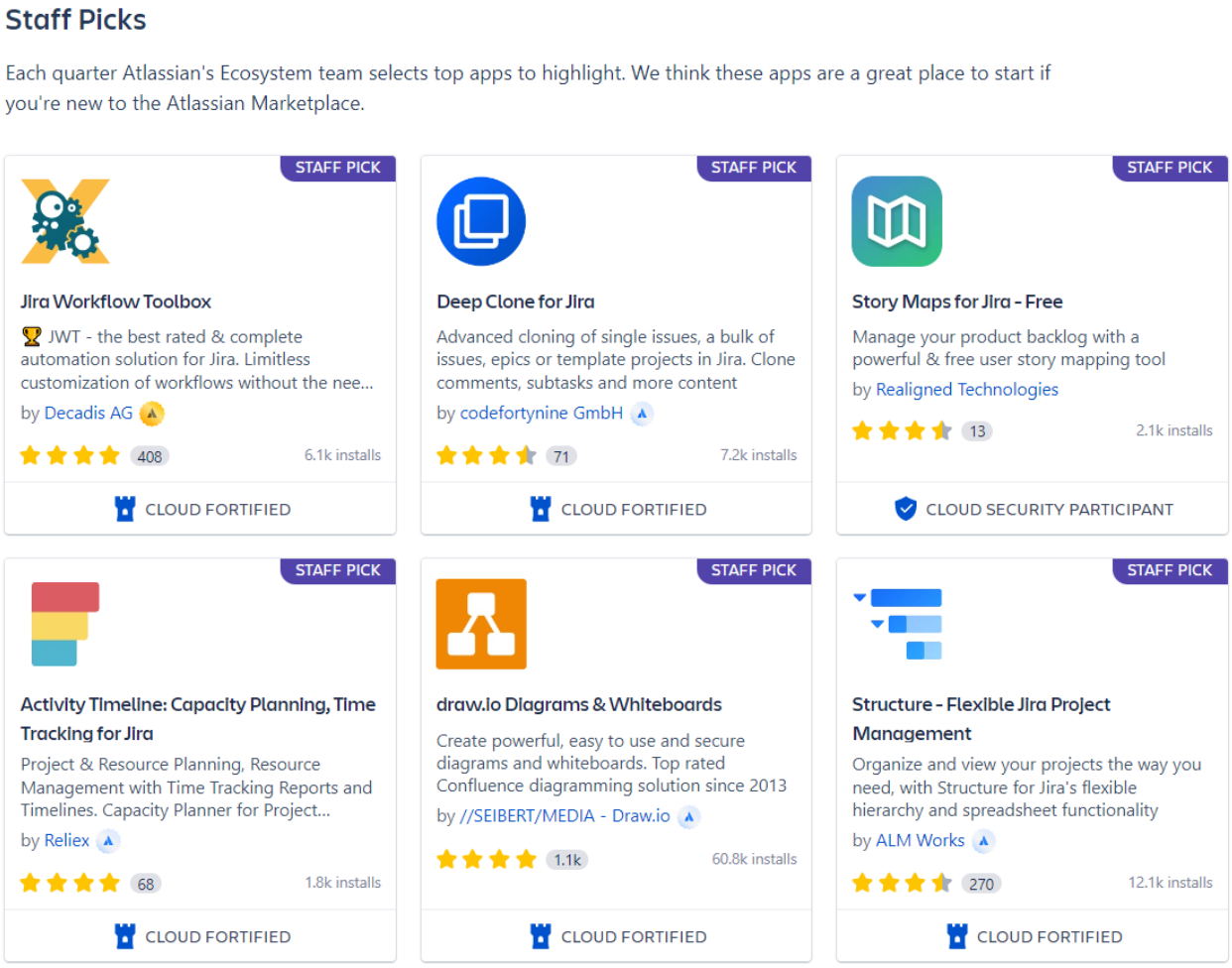

Рис.1.1. Рекомендовані плагіни на головній сторінці Jira Marketplace

1.2. Дослідження проблеми та предметної області

Основною проблемою яку вирішує дана дипломна робота є страх (вищий рівень стресу, дискомфорт) менеджера забути або втратити інформацію про поставлену задачу під час комунікації її (задачі) з колегами, і, як наслідок, втратити контроль за її прогресом та результатами. Сам процес переключення уваги на інструменти постановки/формулювання задачі відволікає і збільшує ризик втрати контексту/деталей думок під час. Поточне вирішення цієї проблеми полягає в високому рівні самоорганізованості менеджера або команди для збереження достатнього фокусу по трекінгу поставлених задач (вищий рівень стресу).

З цього і виникає необхідність у високорівневому інструменті для допомоги в організації задач в команді з мінімізацією фокусу на процес їх постановки. Додаток може стати корисним інструментом для всіх команд, що

використовують Джиру як інструмент для проектного менеджмента. Цільова аудиторія продукту в перетині:

- USA + West Europe
- B2B
- Managers of 20+ team members
- Users of Atlassian Jira
- Users of Google Workspace (Gmail)
- English as a communication language
- “Early adopters” of new tech-based multimodal accessibility tools like Fireflies etc.

В сучасних реаліях, є актуальною автоматизація будь-яких можливих процесів. Автоматизація означає підвищення продуктивності, вона дозволяє працівникам уникати рутинної роботи, і приділяти основний фокус більш важливим проблемам, що потребують вирішення. При виконанні вручну, така робота може використати велику кількість людських ресурсів. Отож, виникає необхідність в автоматизації такого процесу.

Основними проблемами, які можуть виникнути в розробці даної програмної системи є складність правильної інтерпретації полів задач Jira. Основний фокус буде спрямовано на базові поля, які є в більшості проектів і мають чіткі призначення.

Окрім того, що задачі Jira містять десятки можливих вбудованих полів, користувачі можуть створювати так звані кастомні поля, контекст яких не відомий нашій програмній системі, тож їх інтерпретація буде проблематичною. На Рис.1.2 показано список кастомних полів, до яких входять і вбудовані, і створенні користувачем вручну.

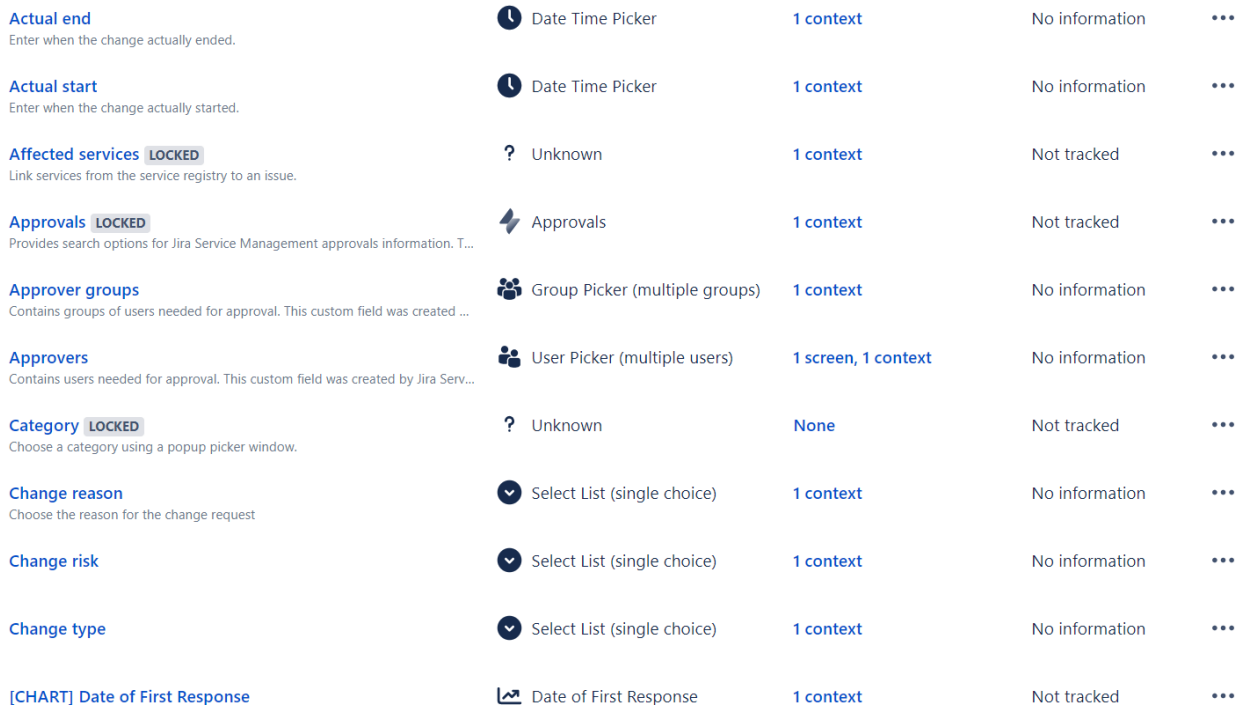

Рис.1.2. Список кастомних полів Jira

На відміну від кастомних, базові поля можна знайти одразу, відкривши будь яку issue. Ці поля описують основну концепцію поставленої задачі і надають основну інформацію про її постановку. Список таких полів:

- Summary - короткий опис задачі
- Description - детальний опис задачі з можливістю підключення мультимедіа
- Priority - пріоритет виконання задачі
- Assignee - призначений виконавець задачі
- Due date - кінцева дата для виконання завдання
- Status - статус задачі в воркфлові
- Labels - теги задачі, що допомагають орієнтуватись між ними та пришвидшувати пошук необхідних

На Рис.1.3 виділені вказані поля на панелі задачі.

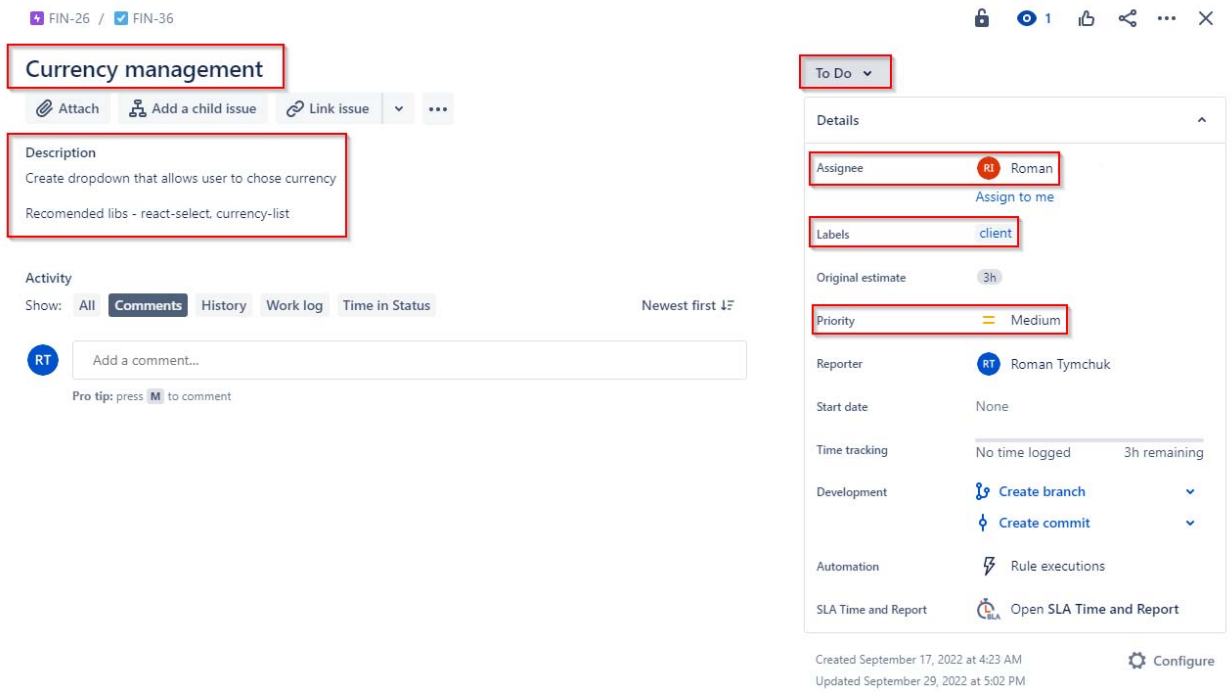

Рис.1.3. Базові поля Jira на панелі задачі

Окрім визначення основних полів - необхідно визначити проект, в якому потрібно створити задачу. Для цього можна враховувати такі фактори як:

- Назва проекту
- Тип проекту
- Опис проекту
- Задачі, які належать до проекту

Варто врахувати також те, що штучний інтелект не завжди є бездоганним в виконанні своїх задач. Висока точність його роботи є результатом тренування на великих вибірках даних. Якщо ML було проведено лише поверхнево, в недостатній кількості або не враховувало усіх аспектів, які можуть виникнути в реальних ситуаціях, результати роботи готової моделі можуть бути непередбачуваними. Для досягнення максимальної точності, необхідно проаналізувати велику кількість електронних листів різного типу та вмісту, а також провести дослідження задач Jira, їх структури, опису, формату даних та інших деталей, що стосуються створення.

Однією з проблем розробки такої програмної системи може бути використання в тексті листа різних мов. Оскільки більшість компаній світу використовують в комунікації англійську мову, яка також є найпоширенішою мовою в світі, доцільно розробити програмну систему для роботи з саме цією мовою, з можливістю розширення цього функціоналу в майбутньому. На Рис. 1.4 показано топ 10 найпоширеніших мов на світовому бізнес ринку.

Top 10 Business Languages in Global Market

Rank	Language	'Score'
1	French	22
2	German	21
3	Chinese	20
4	Arabic	19
5	Spanish	16
6	Dutch	13.5
7	Polish	10
8	Italian	8
9	Japanese	7
10	Hindi	6

Рис.1.4. Найпопулярніші мови на світовому бізнес ринку

Для зручності використання без переходу на інші сторінки, програмна система централізує та надасть користувачеві такі можливості як аналіз тесту листа штучним інтелектом, ручне створення чи редагування задачі, коментування та пошук задач за допомогою зручного вбудованого інтерфейсу. Нижче наведено перелік бізнес процесів, доступних користувачеві:

- Аналіз залиста нейромережею
- Створення задачі на основі даних аналізу
- Ручне створення/редагування задачі
- Пошук задач в інстансі Jira
- Коментування задач

Бізнес-процес “Аналіз листа” необхідний для трансформації тексту електронного листа в зрозумілу структуру з постановкою задачі. Після виконання цього процесу, користувач отримує результат аналізу, може його редагувати і використовувати в процесі “Створення задачі на основі даних аналізу”.

Бізнес-процес “Створення задачі на основі даних аналізу” відповідає за створення задачі з використанням чистих або редагованих користувачем даних аналізу листа. Цей процес може бути автоматизованим для автоматичного створення задач з усіх листів які містять постановку задачі.

Бізнес-процес “Ручне створення/редагування задачі” надає можливість створювати задачу за допомогою ручного введення її полів в надану форму, яка має можливість розширюватись коли користувач додає свої кастомні поля. Такий самий функціонал надається для редагування задачі - користувач знаходить необхідну йому за необхідними фільтрами та редагує поточні значення на необббхідні.

Бізнес-процес “Пошук задач в інстансі Jira” слугує для зручного пошуку задач Jira за допомогою JQL або фільтрів які надаються програмною системою, наприклад за:

- Summary
- Description
- Status
- Priority
- Assignee
- Інші

Бізнес-процес “Коментування задач” дозволяє користувачу знаходити задачу та додавати до неї коментарі. Користувач може вибрати задачу з запропонованих системою або скористатись можливістю пошуку задач для вибору необхідної йому.

В таблиці 1.1 наведена характеристика для наведених вище бізнес-процесів.

Таблиця 1.1

Характеристика бізнес-процесів

Бізнес процес ”Аналіз листа нейромережею"	
Процес	Проведення аналізу листа нейромережею
Учасники	Користувач, надсилач листа
Вхідна подія	Отримання електронного листа
Вхідний документ	Вміст електронного листа
Вихідна подія	Отримання результату аналізу користувачем
Вихідний документ	Дані аналізу листа
Клієнт бізнес	Користувач
Бізнес процес ”Створення задачі на основі даних аналізу"	
Процес	Створення задачі в Jira
Учасники	Користувач
Вхідна подія	Отримання результатів аналізу листа або підтвердження створення

	задачі користувачем
Вхідний документ	Дані аналізу листа
Вихідна подія	Задача створена на інстансі Jira
Вихідний документ	Задача Jira
Клієнт бізнес	Користувач
Бізнес процес ”Ручне створення/редагування задачі"	
Процес	Створення/редагування задачі методом ручного введення необхідних значень в поля
Учасники	Користувач, автор задачі
Вхідна подія	Вибір опції створення/редагування задачі
Вхідний документ	Форма з полями для створення задачі (заповненими полями в разі редагування задачі)
Вихідна подія	Створення задачі/Оновлення вмісту задачі
Вихідний документ	Задача Jira
Клієнт бізнес	Користувач
Бізнес процес ”Коментування задач"	
Процес	Додавання коментаря в задачу Jira
Учасники	Користувач, автор задачі, члени

	обговорення
Вхідна подія	Вибір опції коментування задачі
Вхідний документ	Задача Jira
Вихідна подія	Додання коментаря в задачу Jira
Вихідний документ	Коментар в задачі Jira
Клієнт бізнес	Користувач

Для більш детального уявлення про етапи цих процесів, на рис.1.5 - 1.8 відображено структуру описаних вище бізнес-процесів.

Рис.1.5. структура бізнес-процесу “Аналіз листа нейромережею”

Рис.1.6. Структура бізнес-процесу “Створення задачі”

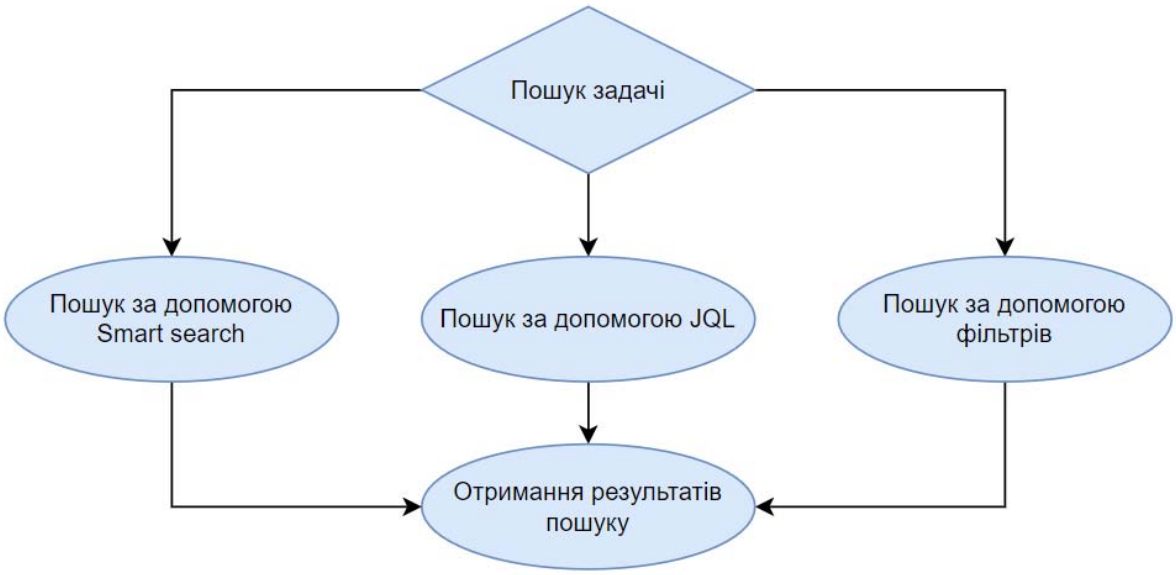

Рис.1.7. Структура бізнес-процесу “Пошук задачі”

Рис.1.8. Структура бізнес-процесу “Коментування задачі”

1.3. Аналіз додатків конкурентів

Дане програмне рішення проблеми є інноваційним, однак, існують конкурентні програмні системи, які опосередковано відносяться до поставленої проблеми.

acta.ai - програмний продукт, який використовує алгоритми нейронної мережі для аналізу мітингів. На Рис.1.9 зображено модель використання даної програмної системи.

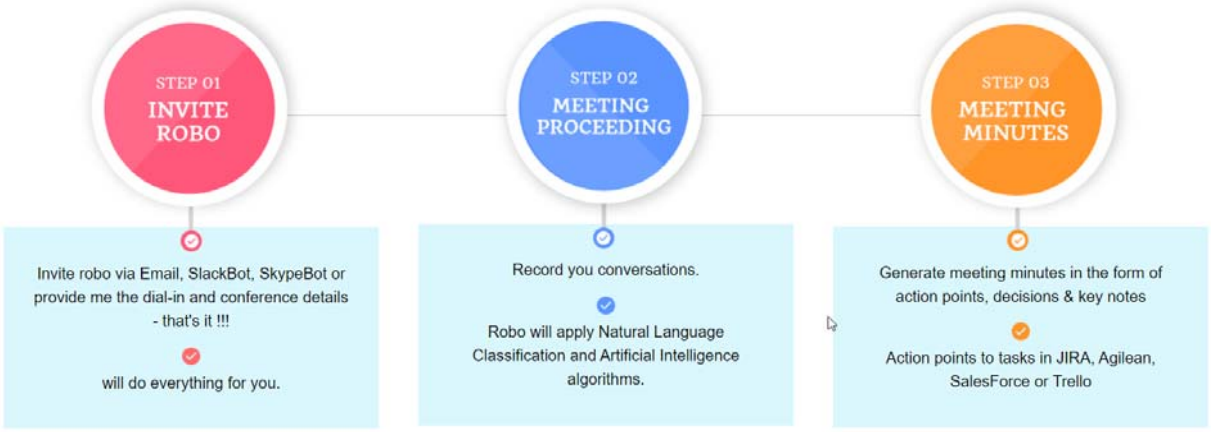

Рис.1.9. Використання acta.ai

Також, є можливість інтеграції з Jira, для автоматизованого створення задач, та інших сутностей. Дане рішення є досить схожим до розроблюваного, однак не надає користувачам необхідного функціоналу.

fireflies.ai - друге конкурентне рішення яке розглядається в аналізі. За функціоналом, воно дуже схоже з попереднім, однак має свої переваги та недоліки. Перевагами є бізнес-орієнтованість, високорівневі алгоритми нейронних мереж, надання розширених уявлень про концепцію мітингів. Однак, в нашому випадку, неможливість інтеграції з системою Jira є критичним недоліком. На Рис.1.10 наведено категоризацію функціоналу, який надається даною програмною системою.

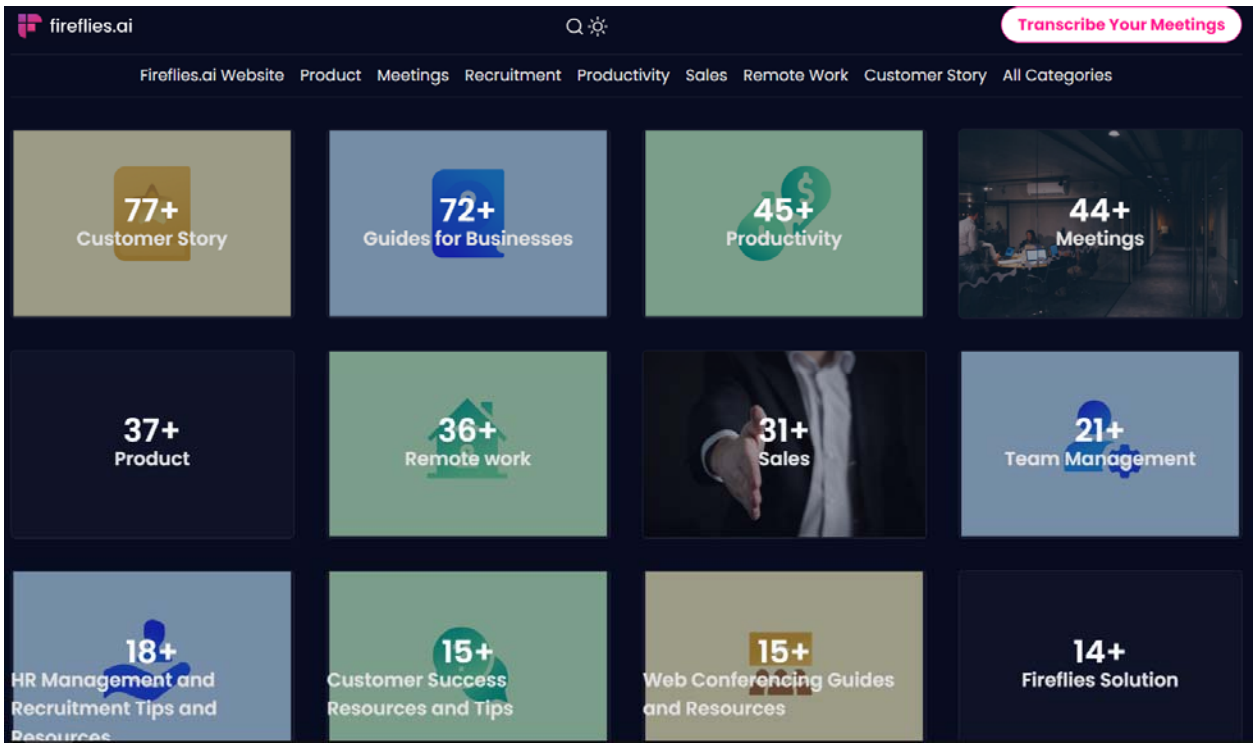

Рис.1.10. Функціонал, що надається системою fireflies.ai

Google To Jira - остання програмне рішення, що розглядається в якості конкурента. Ця програмна система є плагіном на Jira Marketplace, отже є прямим конкурентом для розробленого рішення. На Рис.1.11 показано плагін на сторінці Jira Marketplace.

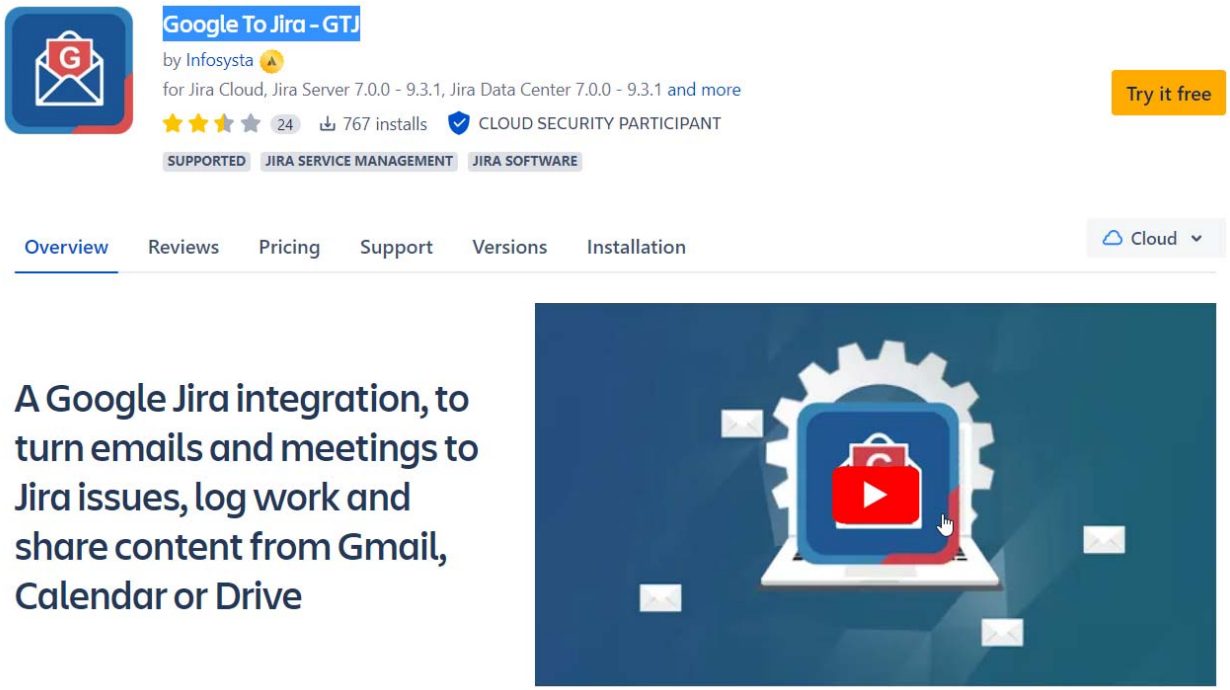

Рис.1.11. Google To Jira

Основним функціоналом даного плагіну є інтерпретація листів та мітингів в задачі, ворклоги та поширений контент з Gmail. Однак, це рішення не повторює функціоналу який надає програмний продукт, що є результатом даної роботи, створення задач з листів - не автоматизоване, користувач має мануально вказувати необхідні поля, обирати необхідні листи. Розширюється лише програмний інтерфейс для взаємодії.

1.4. Розробка специфікацій вимог

Для формування специфікації потрібно розробити діаграму використання для програмної системи. На рис. 1.12 зображена діаграма варіантів використання.

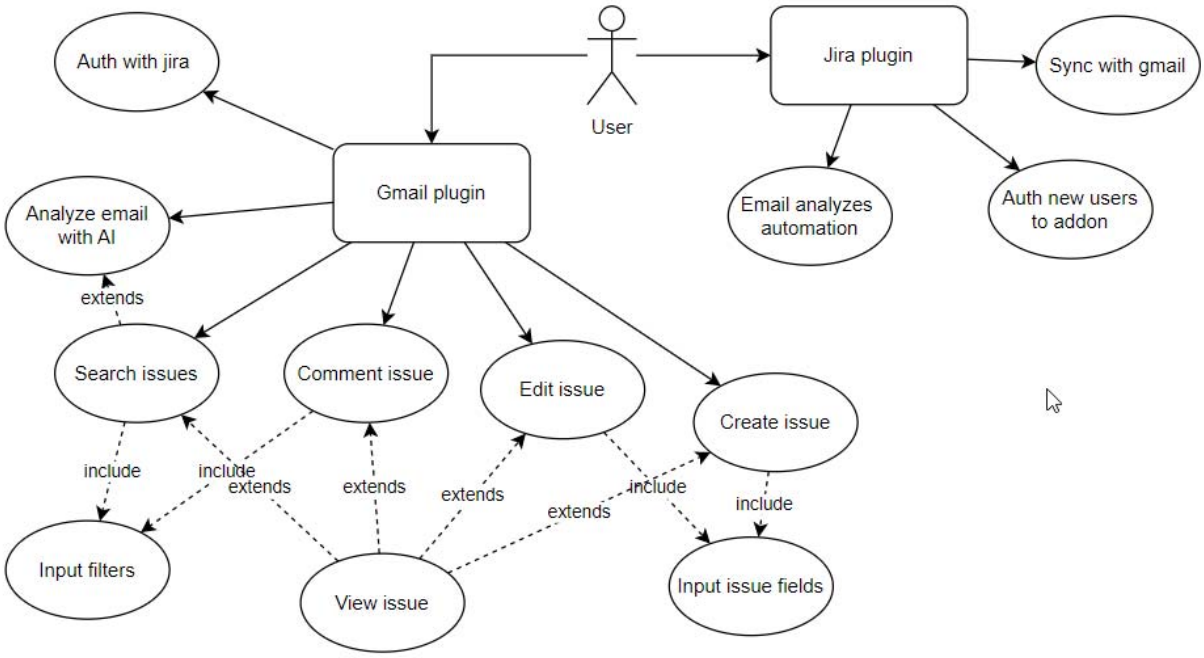

Рис.1.12. Діаграма варіантів використання

Далі, необхідно описати кожен можливий варіант використання за допомогою таблиці варіантів використання. Першим варіантом є аналіз задачі

за допомогою нейромережі. Таблиця варіантів використання для аналізу задачі наведена у таблиці 1.2.

Таблиця 1.2

Варіанти використання для “Аналіз задачі за допомогою нейромережі”

Використання	Отримання результатів аналізу обраного листа
Дійові особи	Користувач
Передумова	Користувач перейшов на сторінку листа або обрав опцію “AI Generated”
Тригер	Натиснення кнопки “Analyze”
Сценарій	Варіант 1. Обрати необхідний лист, натиснути кнопку “Analyze”. Варіант 2. Перейти на необхідний лист, вибрати опцію “Analyze”
Післяумова	Візуалізація результатів аналізу

Варіанти використання для “Створення задачі” зображені на Таблиці 1.3.

Таблиця 1.3

Варіанти використання для “Створення задачі”

Використання	Створення задачі в системі Jira
Дійові особи	Користувач
Передумова	Користувач отримав результати аналізу

	нейромережею або обрав опцію “Create issue”
Тригер	Натиснення кнопки “Create”
Сценарій	Варіант 1. Натиснути кнопку “Create” після отримання результатів аналізу штучним інтелектом Варіант 2. Обрати опцію “Create issue”, заповнити всі необхідні поля, натиснути кнопку “Create”
Післяумова	Виведення створеної задачі на екран

Варіанти використання для “Коментування задачі” зображені на Таблиці 1.4.

Таблиця 1.4

Варіанти використання для “Коментування задачі”

Використання	Коментування задачі в системі Jira
Дійові особи	Користувач
Передумова	Користувач вибрав опцію “Comment issue”
Тригер	Натиснення кнопки “Post comment”
Сценарій	Введення тексту коментаря, натиснення кнопки “Post comment”
Післяумова	Виведення прокоментованої задачі та обговорення на екран

Варіанти використання для “Пошук задачі” зображені на Таблиці 1.5.

Таблиця 1.5

Варіанти використання для “Пошук задачі”

Використання	Пошук задачі в системі Jira
Дійові особи	Користувач
Передумова	Користувач вибрав опцію “Search issue”
Тригер	Натиснення кнопки “Search”
Сценарій	Введення в відповідне поле JQL стрічки, тексту для Smart search або налаштування необхідних фільтрів для пошуку.
Післяумова	Виведення списку знайдених задач на екран.

Для розуміння необхідного функціоналу в процесі розробки, необхідно також розробити специфікацію функціональних вимог. Вона наведена у таблиці 1.6.

Таблиця 1.6

Специфікація функціональних вимог

ID	Title	Priority	Complexity
1	Auth Gmail plugin with Jira	High	High
2	Analyse issue with AI from gmail plugin	High	Medium
3	Create Jira issue from gmail plugin	High	Medium

4	Edit Jira issue from gmail plugin	High	Medium
5	Comment Jira issue from gmail plugin	Medium	Medium
6	Search Jira issues from gmail plugin	High	Medium
7	Automate emails analyse	Medium	High
8	Time logging from gmail plugin	Medium	High

Висновок до першого розділу.

Проведено дослідження по необхідності розробки програмної системи для Jira та Gmail, яка використовуватиме алгоритми нейронної мережі для аналізу тексту і формуватиме з них основні пункти. Переглянуто усі типи полів в задачах Jira, їх структуру, призначення та можливі асоціативні слова та фрази. Проведено порівняння різних способів аналізу тексту за допомогою нейромереж.

РОЗДІЛ 2
АРХІТЕКТУРА ПРОГРАМНОЇ СИСТЕМИ

2.1. Архітектура плагіну

Програмна система поділяється на три частини:

- плагін для Google workspace, завдання якого взаємодіяти з Gmail сервісом.
- плагін для Jira, задача якого взаємодіяти з системою Atlassian, створювати задачі.
- нейромережа для аналізу тексту листів, яка зв'язує ці два плагіни.

На Рис.2.1 показано архітектуру взаємодії між всіма компонентами системи.

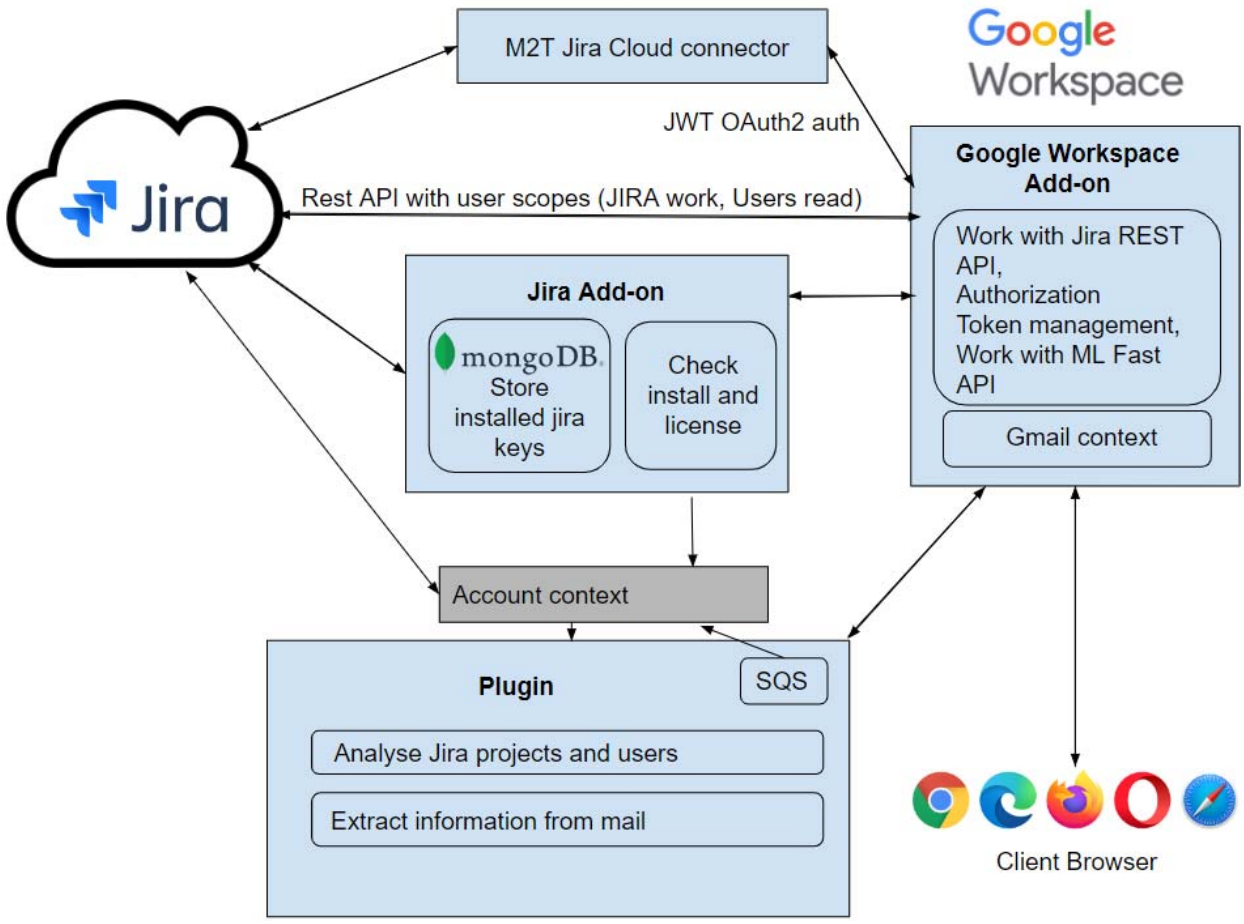

Рис.2.1. Архітектура роботи програмної системи

2.2. Функціональна частина

Функціонал, що надається програмною системою, передбачає надання користувачеві таких можливостей:

- Автоматично створювати задачі з ел. листів
- Редагувати поля задачі в Jira за допомогою ел. Листів
- Включати в такі листи додатки та вкладення
- Коментування задач будь-яким членом команди
- Логування витраченого часу
- Вставляти задачі Jira в ел. листи
- Шукати будь які задачі за допомогою швидкого пошуку

Для розгорнутого представлення про роботу програмної системи, необхідно розробити діаграми станів і послідовностей для основних процесів, а саме таких:

- Аналіз листа
- Створення задачі
- Коментування задачі
- Редагування полів задачі

Процес аналізу листа доступний користувачам які встановили плагін для системи Gmail, він дозволяє обрати лист, який необхідно проаналізувати, і, за допомогою нейронної мережі, генерує з його вмісту поля нової задачі. На Рис.2.2 та 2.3 зображено діаграми станів та послідовності для процесу аналізу листа.

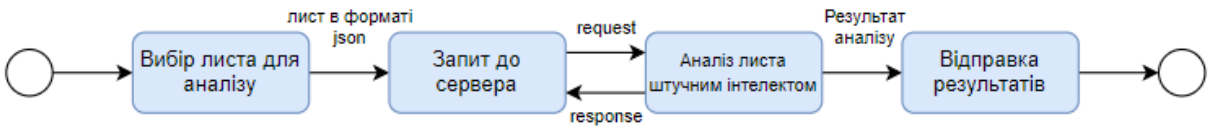

Рис.2.2. Діаграма станів процесу аналізу листа

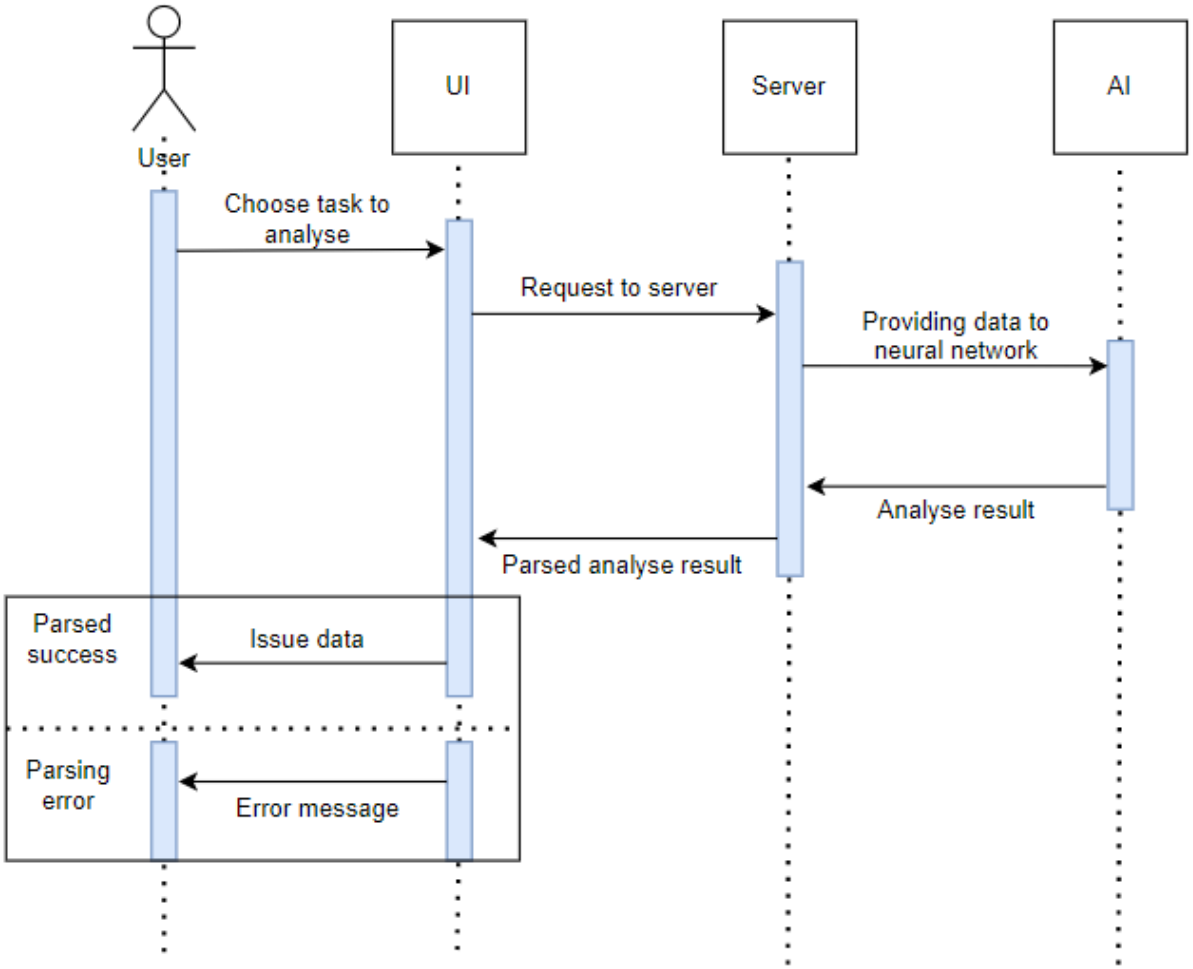

Рис.2.3. Діаграма послідовності для процесу аналізу листа

Процес створення задачі доступний користувачам які встановили плагін для системи Gmail. Здійснення процесу можливо двома способами - за допомогою мануального введення необхідних полів, або створення на основі аналізу листа.. На Рис. 2.4 та 2.5 зображено діаграми станів та послідовності для процесу створення задачі на основі даних повернених з аналізу листа.

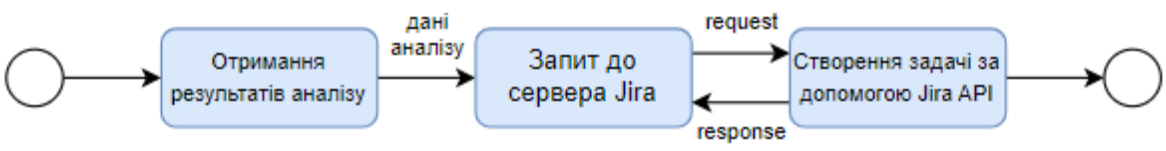

Рис.2.4. Діаграма станів процесу створення задачі

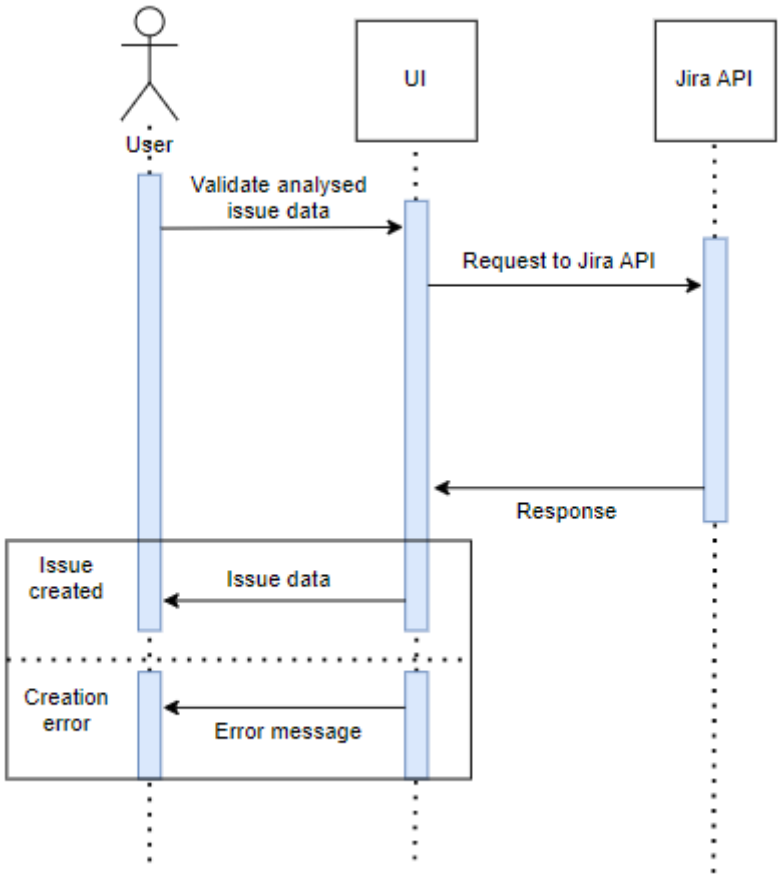

Рис.2.5. Діаграма послідовності для процесу створення задачі

Процес коментування задачі доступний користувачам які встановили плагін для системи Gmail. Здійснення процесу можливо двома способами - за допомогою мануального введення тексту коментаря, або на основі даних з аналізу листа. На рис. 2.6 та 2.7 зображено діаграми станів та послідовності для процесу коментування задачі на основі даних повернених з аналізу листа.

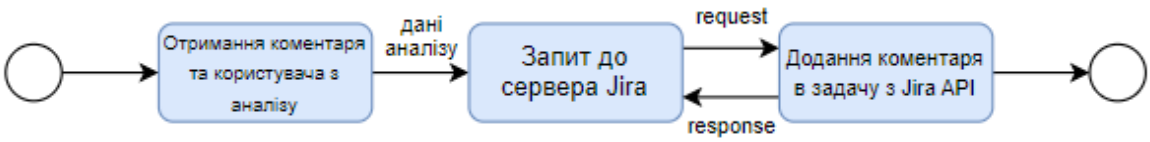

Рис.2.6. Діаграма станів процесу коментування задачі

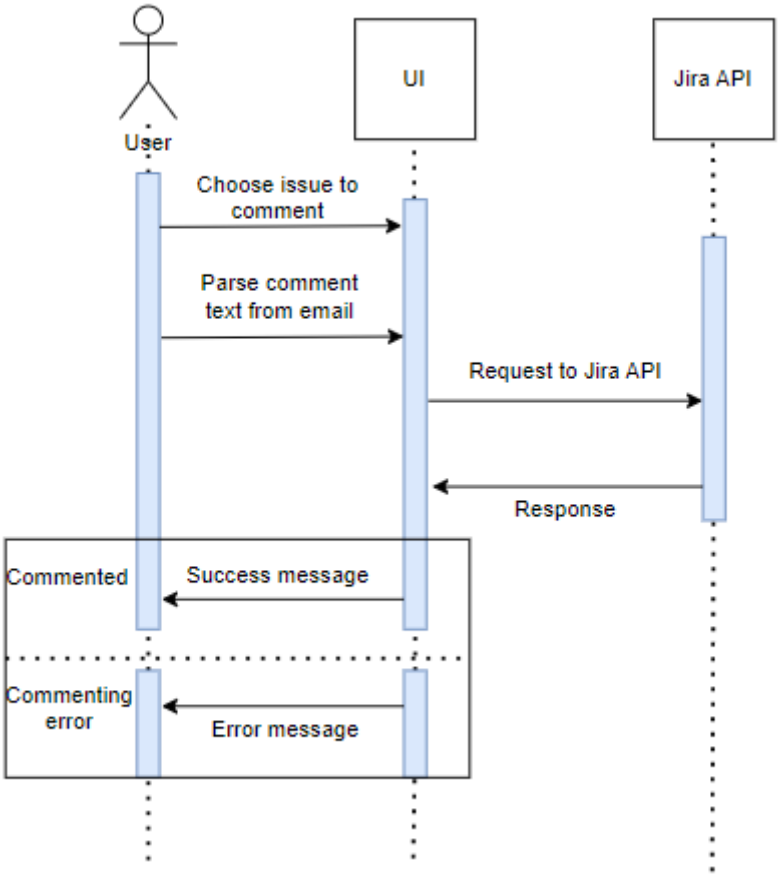

Рис.2.7. Діаграма послідовності для процесу коментування задачі

Процес редагування задачі доступний користувачам які встановили плагін для системи Gmail. Здійснення процесу можливо двома способами - за допомогою мануального введення нових полів задачі, або на основі даних з аналізу листа. На Рис.2.8 та 2.9 зображено діаграми станів та послідовності для процесу коментування задачі з ручним введенням даних.

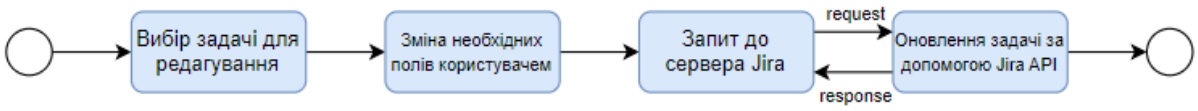

Рис.2.8. Діаграма станів процесу редагування задачі

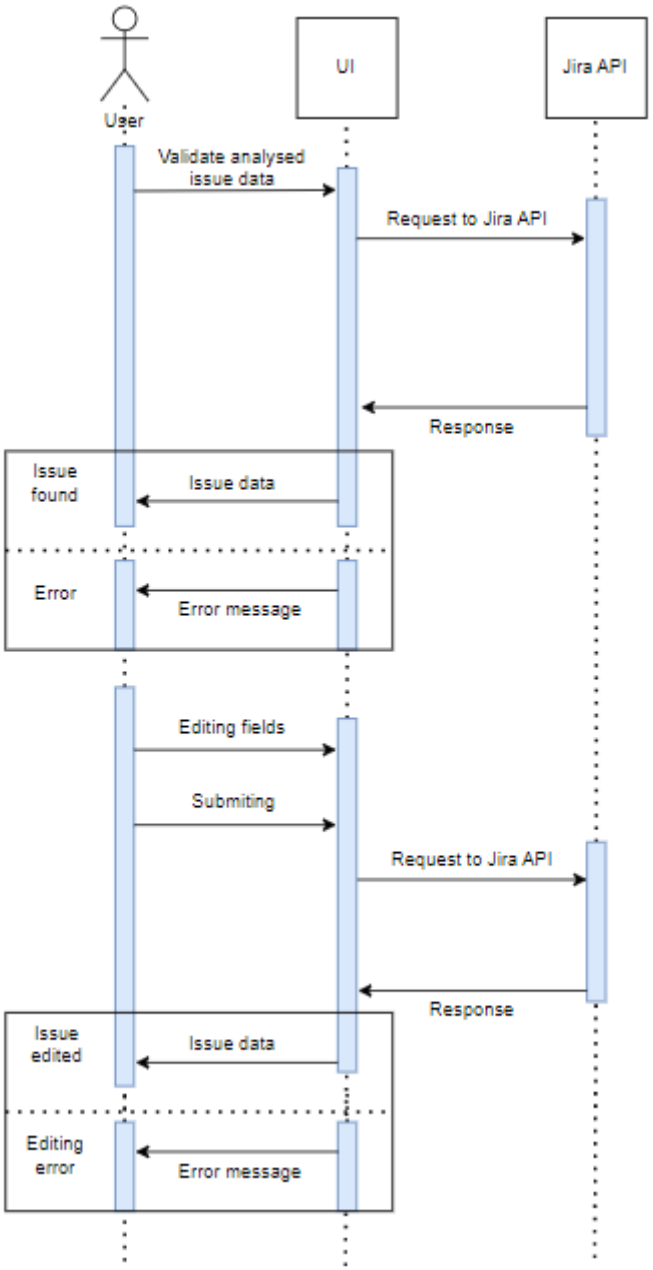

Рис.2.9. Діаграма послідовності для процесу редагування задачі

2.3. База даних програмної системи

В даній програмній системі використовуватиметься документоорієнтована база даних - MongoDB Atlas. Для стабільної роботи бази даних на протязі тривалого часу необхідне правильне її проектування. Воно передбачає виконання таких пунктів:

- Вибір системи управління базою даних.
- Визначення основних сутностей сутностей.
- Визначення зв’язків між ними.
- Створення діаграм потоків даних.

Для даної програмної системи в базі даних необхідно зберігати таку сутності:

- users
- jiras
- addonSettings
- runConfig

Для кожної з них буде створено окреме сховище(колекція), де зберігатимуться документи у вигляді json об’єктів.

Колекція users міститиме інформацію про окремих користувачів. Інформація про них братиметься безпосередньо з Jira, і міститиме основну інформацію про те, до якого інстансу належить користувач, його налаштування персоналізації та рівень доступу.

Колекція jiras зберігатиме документи з окремими інстансами Jira. Інформація про них також надається самою системою і включає великий обсяг даних про рівні та систему доступів,токени доступу, інформацію про стадію взаємодії з програмною системою та віджети.

В колекції addonSettings зберігатимуться актуальні ключі доступів для користувачів та окремо для різних інстансів. Дані є динамічними, тому постійно змінюватимуться.

runConfig - колекція для сінглтон документа, який створюється при запуску додатку. Він містить в собі основні налаштування запуску, токени доступу та інформацію про програмну систему.

В таблицях 2.1, 2.2, 2.3 та 2.4 зображено основні поля для сутностей users, jiras, addonSettings та runConfig, відповідно.

Таблиця 2.1

Структура даних в колекції users

Поле	Тип даних	Опис
info	object	Містить персональну інформацію користувача
info.self	string	Містить посилання на користувача в системі Jira
info.timeZone	string	Містить інформацію про часовий пояс користувача
info.displayName	string	Містить ім’я користувача яке відображається в системі Jira
info.locale	string	Містить інформацію про локалізацію користувача
info.active	boolean	Відображає статус активності користувача
info.expand	string	Містить в собі список груп до яких належить цей користувач у вигляді стрічки
info.accountId	string	Відображає id користувача в системі Jira
account	string	Містить id інстансу до якого належить користувач
bearerExp	number	Містить інформацію про час дії токену доступу

gadgets	object	Містить повну інформацію про додані користувачем на дашборд гаджети, їх конфігурацію та доступність.
productTour	boolean	Відображає чи пройшов користувач інструкцію по використанню програмної системи

Таблиця 2.2

Структура даних в колекції jiras

Поле	Тип даних	Опис
installed	object	Містить інформацію про інстанс Jira
installed.key	string	Містить ключ встановленого плагіну
installed.clientKey	string	Містить id інстансу в системі Jira
installed.baseUrl	string	Містить посилання на інстанс Jira
version	string	Містить інформацію про встановлену на інстанс версію плагіну
encodedToken	string	Містить інформацію про загальний токен доступу
calendars	array	Містить список календарів які використовуються в інстансі
accountInfo	object	Містить дані про дату першого встановлення плагіну на інстанс та дату останнього оновлення

gadgets	object	Містить повну інформацію про додані на дашборд гаджети в інстансі, їх конфігурацію та доступність.

Таблиця 2.3

Структура даних в колекції addonSettings

Поле	Тип даних	Опис
clientKey	object	Містить id інстансу в системі Jira
key	string	Містить ключ встановленого плагіну
val	string	Містить інформацію про доступ до плагіну, токен доступу
val.baseUrl	string	Містить посилання на інстанс Jira
val.publicKey	string	Містить загальний ключ доступу до плагіну з інстансу Jira
val.token	string	Містить інформацію про загальний токен доступу

Таблиця 2.4

Структура даних в колекції runConfig

Поле	Тип даних	Опис
logMode	object	Містить інформацію про внутрішнє логування помилок, що виникають під

		час роботи додатку
logMode.log	boolean	Містить інформацію про необхідність логування
logMode.path	string	Містить інформацію про шлях до сервісу для логування
production	object	Містить основні змінні для запуску програмної системи
production.baseUrl	string	Містить посилання по якому буде доступною програмна системи після запуску
production.store	string	Містить інформацію про локальне сховище програмної системи
expressErrorHandling	boolean	Містить інформацію про необхідність автоматичного виявлення помилок за допомогою окремого плагіну.
mongo	object	Містить інформацію про базу даних
mongo.url	string	Посилання доступу до бази даних
mongo.database	string	Назва бази даних що використовується
mongo.collections	array	Список колекцій, які є в базі даних програмної системи

Для проектування з’єднання серверної частини з базою даних необхідно визначити яким чином будуть відправлятись та записуватись дані. Діаграма потоків даних є інструментом для проектування з’єднання

серверної частини з базою даних додатку і відображає які процеси системи викликають запити до бази даних. DFD діаграма представлена на Рис.2.10.

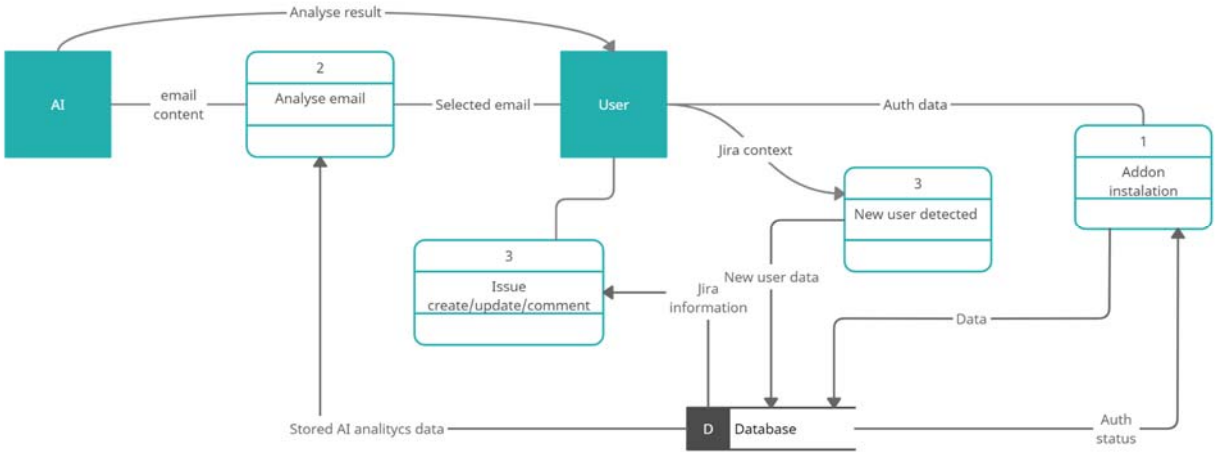

Рис.2.10. DFD діаграма додатку

Для детального уявлення про структуру даних в базі даних, необхідно створити діаграму структури сутностей. Така діаграма створена окремо для кожної з них і включає в себе відображення зв’язків полів між сутностями. На Рис.2.11 - 2.14 зображені діаграми структури сутностей для колекцій users, jiras, addonSettings, runConfig.

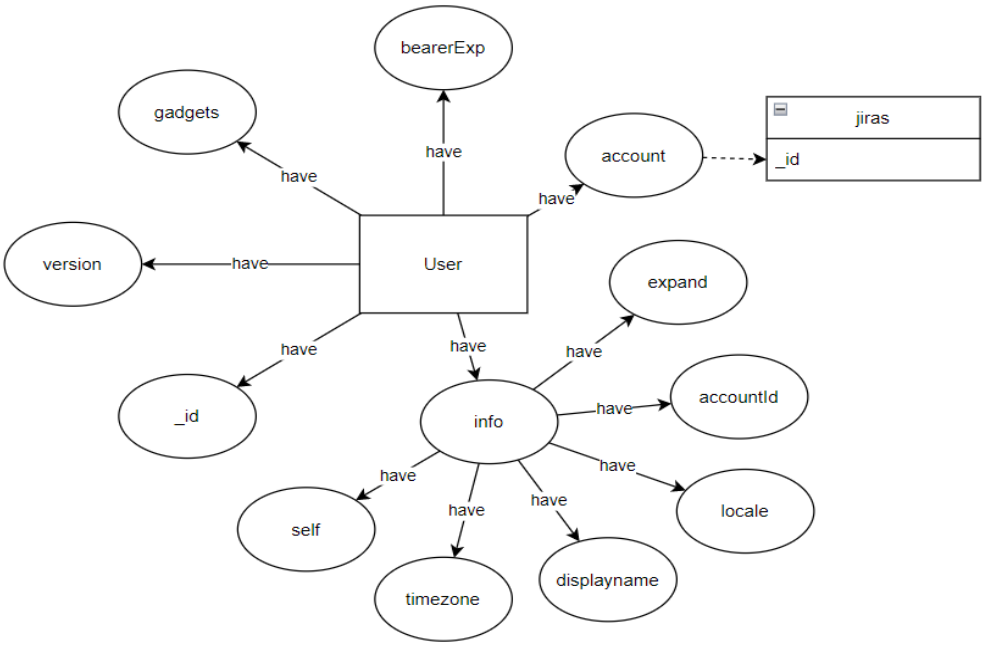

Рис.2.11. Діаграма структури сутності user

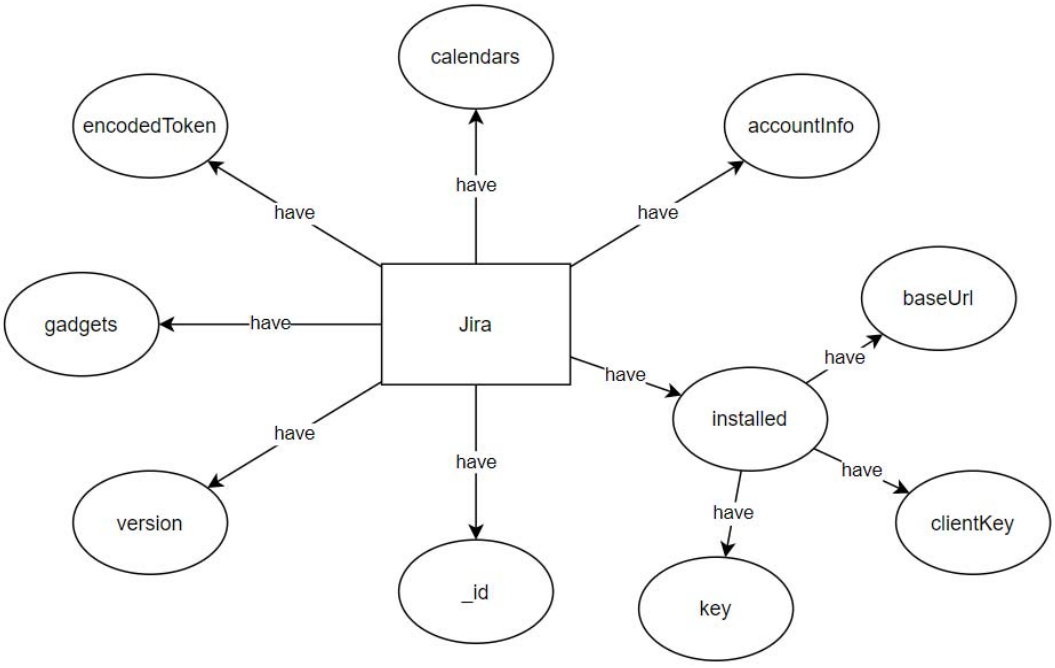

Рис.2.12. Діаграма структури сутності jira

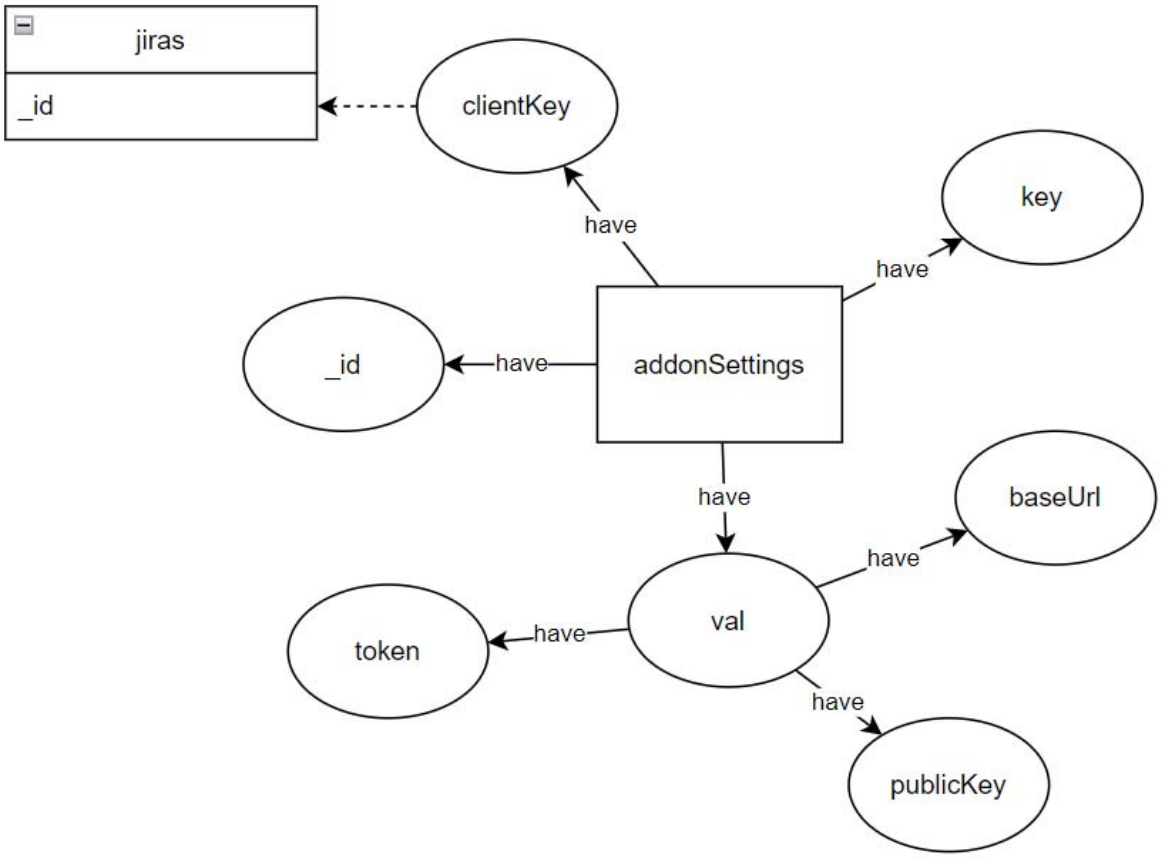

Рис.2.13. Діаграма структури сутності addonSetting

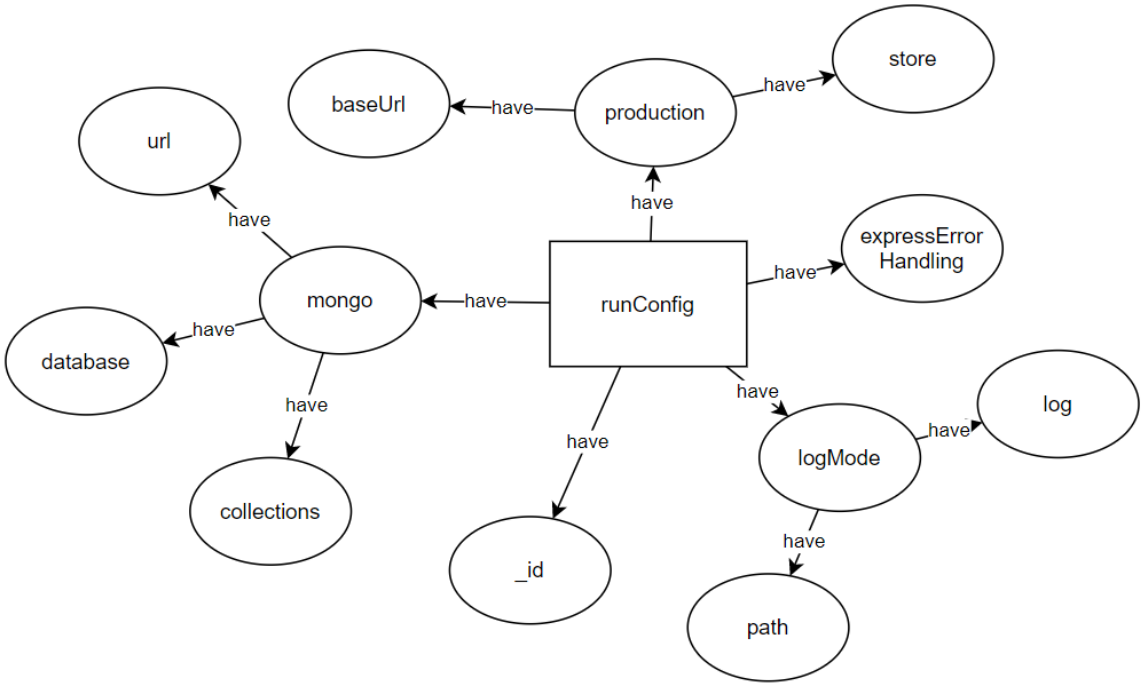

Рис.2.14. Діаграма структури сутності runConfig

2.4. Алгоритми нейромережі

ML технологія дозволяє створювати програмні системи, здатні виконувати поставлені їм задачі з досить високою точністю. Для розгляду найпростішого штучного інтелекту можна взяти перцептрон. Це математична або програмна модель сприйняття вхідної інформації обробником, здатна до навчання та вдосконалення. На Рис.2.15 зображена модель перцептрону.

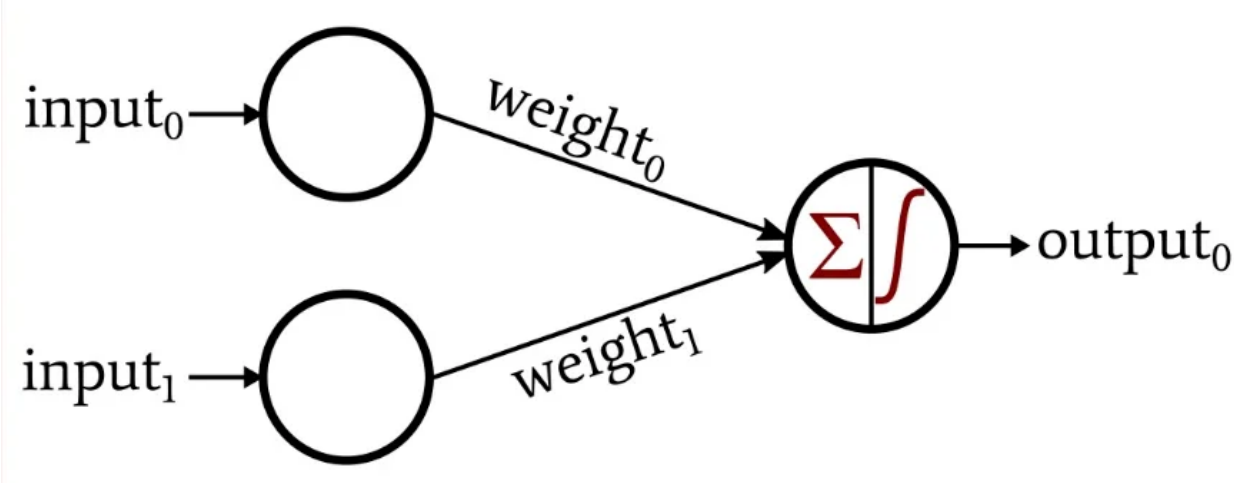

Рис.2.15. Перцептрон

Принцип його роботи заключається в наступному: модель приймає на вхід деяку кількість вхідних параметрів, кожен з яких має свою вагу, вплив на остаточний результат. ML призначене для калібрування цих ваг з метою покращення точності вихідного результату. Також, модель може мати мати приховані шари, які є проміжними між вхідними даними та результатом.

Для виконання поставленої задачі, необхідно розробити нейронну мережу, що зможе визначати, за який контекст відповідають відповідні частини тексту в електронному листі. Для цього використаємо технологію проставлення лейблів в тексті. На Рис.2.16 зображено принцип роботи лейблінгу.

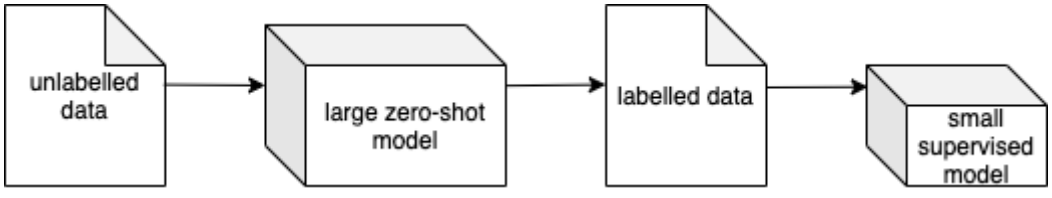

Рис.2.16. Процес лейблінгу тексту

Можна побачити, що така модель приймає звичайний текст без лейблів та колекцію даних з списком можливих лейблів, а в результаті повертає текст з уже проставленими лейблами та контрольовану модель. Точність лейблінгу зростає зі збільшенням кількості тестових ітерацій, які визначають більш оптимальні ваги.

ML такої моделі відбувається методом тестування на великій кількості текстів, що є заздалегідь пролейбленими. Після кожної ітерації, модель змінює свої ваги залежно від точності результатів, наприклад, при неправильному визначенні поля SUMMARY, значення ваг для цього поля можуть сильно змінитись, в той час як ваги для правильного поля або залишаються незмінними, або змінюються лише незначно. Різкість збільшення ваг визначається Learning rate показником.

Висновок до другого розділу

У другому розділі проведено розробку архітектури додатку та бази даних, опис функціональної частини програмної системи та проведено дослідження математичної моделі формування “ваг” для нейромережі, їх стабілізації та обрахунку, що впливають на швидкість та якість подальшого навчання штучного інтелекту на великих вибірках даних.

Мтворено модель роботи парсера тексту, який аналізуватиме тексти листів та проставлятиме в них необхідні лейбли.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Загальні принципи реалізації програмної системи

Дана програмна система розробляється як мікросервіс для системи Jira, однак містить і супутній плагін для gmail, тому набір інструментів визначатиметься батьківським середовищем. І Jira і Gmail пропонують розробникам можливість розробки плагінів на мові програмування javascript та всіх підтримуваних ним модулів.

Для реалізації клієнтської частини додатку використовується бібліотека React.js, що є однією з найбільш популярних на даний момент в веб-розробці. Також, використовуватимуться надані сервісами інструменти розробки, наприклад, atlaskit.

Плагін Jira та плагін Gmail будуть пов’язані між собою через серверну частину додатку. Вона прийматиме запити від обох плагінів, робитиме записи в базу даних і взаємодіятиме з сторонніми сервісами, такими як Jira API чи Google API. В таблиці 3.1 показано список npm модулів які використовуватимуться в розробці додатків.

Таблиця 3.1

Перелік npm модулів.

Назва	Версія	Опис
axios	1.1.2	Створення та відправка http запитів до сервера та сторонніх сервісів
react	16.14.0	Основний пакет для використання React.js
express	4.18.1	Комплексний інструмент для створення сервера та його інтерфейсу
typescript	4.7.4	Використання typescript

atlassian-jwt	2.0.2	Модуль для аутентифікації що надається Atlassian
mailgun-js	0.22.0	Відправка електронних листів
nodemon	2.0.16	Використовується під час розробки
webpack	5.74.0	Необхідний для збірки додатку в вигляд, який підтримується браузерами
babel-loader	8.2.5	Компілятор js необхідний для перетворення коду ES 2015+ в підтримувані різними середовищами версії коду.
@atlaskit	-	Набір інструментів, що надається Atlassian і містить в собі велику кількість різноманітних модулів для полегшення розробки та кращої інтеграції плагіну в систему. Список підмодулів: ● avatar ● button ● checkbox ● datetime-picker ● flag ● form ● icon ● inline-message ● modal-dialog ● select ● tabs ● textarea ● textfield

		● theme ● tokens ● tooltip

Для програмної реалізації плагінів для Jira та Gmail використовуються мови програмування javascript та typescript, кастомізована мова розмітки jsx/tsx та css/less. Реалізація нейронної мережі відбувається за допомогою сторонніх сервісів та мови програмування python.

3.2. Програмна реалізація плагіну для Gmail

Плагін для Gmail розробляється окремим проектом. В самій системі, він відображатиметься в боковій панелі. Структура файлів досить проста: коренева папка містить конфігурації, налаштування деплойменту, файл з токенами доступу та папку src, в якій знаходяться файли розмітки та коду.

В панелі плагін відображається у вигляді карток, для кожної з них - окремий файл. Список карток що були розроблені:

- Auth
- Comment
- DraftOpen
- Edit
- Feedback
- Home
- Issue
- IssueTypes
- Search
- Settings

Приклад фрагменту коду з карти Feedback:

```
function feedbackCard() {
```

```
const card = CardService.newCardBuilder()
const radioGroup = CardService.newSelectionInput()
                    .setType(CardService.SelectionInputType.RADIO_BUTTON)
                    .setTitle('I want to')
                    .setFieldName('radioGroup')
                    .addItem('Ask a question', 'Question', true)
                    .addItem('Leave a comment', 'Comment', false)
                    .addItem('Report a bug', 'Bug', false)
                    .addItem('Suggest an improvement', 'Improvement', false)
const input = CardService.newTextInput()
                .setFieldName('input')
                .setValue('')
                .setTitle('What do you want to say?');
…
```

CardService - об’єкт який надає розробнику усі необхідні інструменти, такі як текстові поля, мультимедіа, поля для вводу, відображення даних будь-якого формату, а також їх кастомізації. Після того, як картка створена, її розмітка згенерована - необхідно використати card.build(), щоб вона з’явилась на користувацькому інтерфейсі. На Рис.3.1 -3.3 зображений користувацький інтерфейс плагіну.

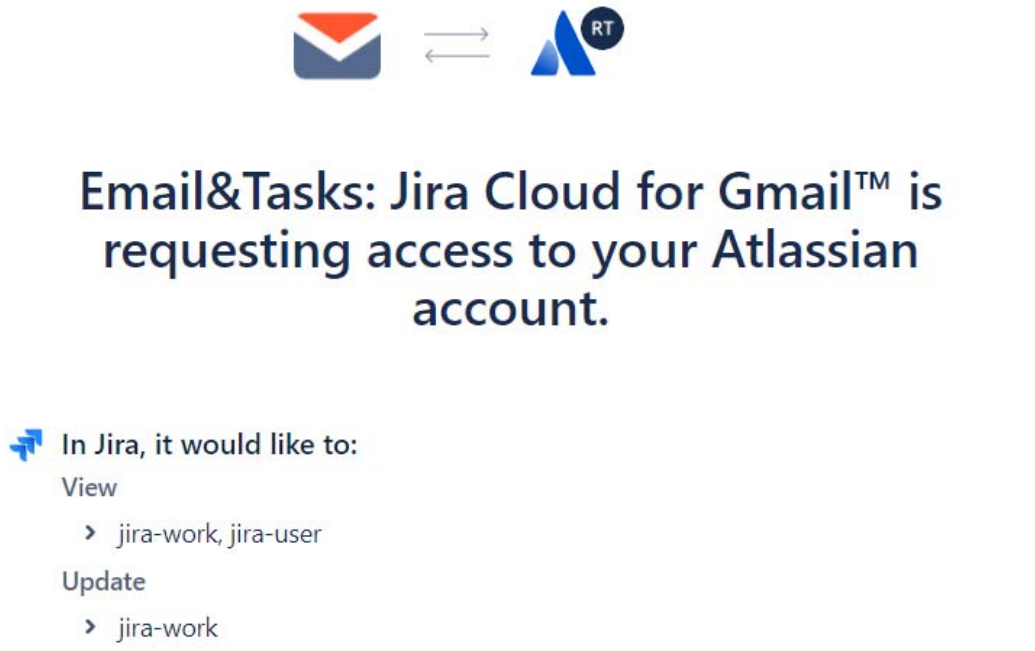

Рис.3.1. Авторизація плагіну з системою Jira

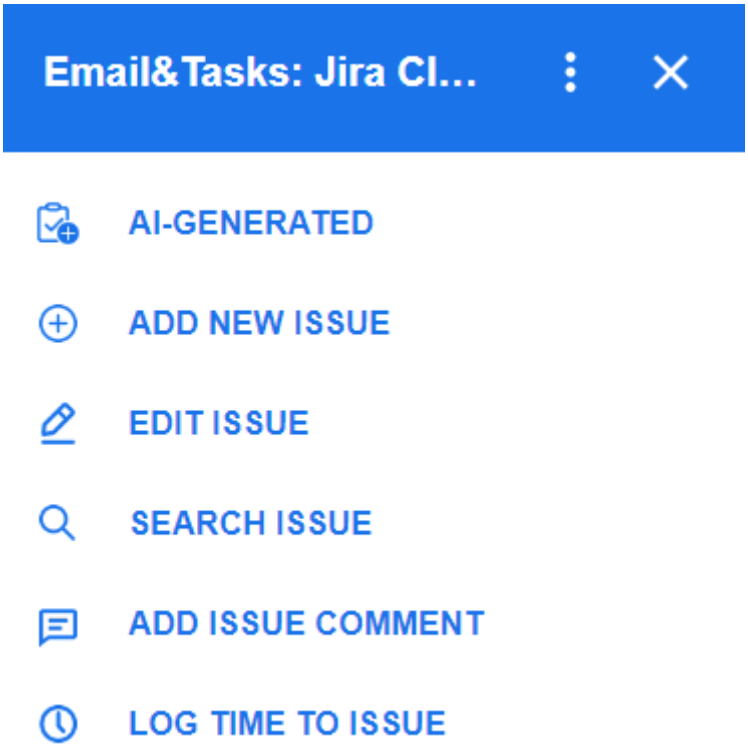

Рис.3.2. Початкова картка плагіну

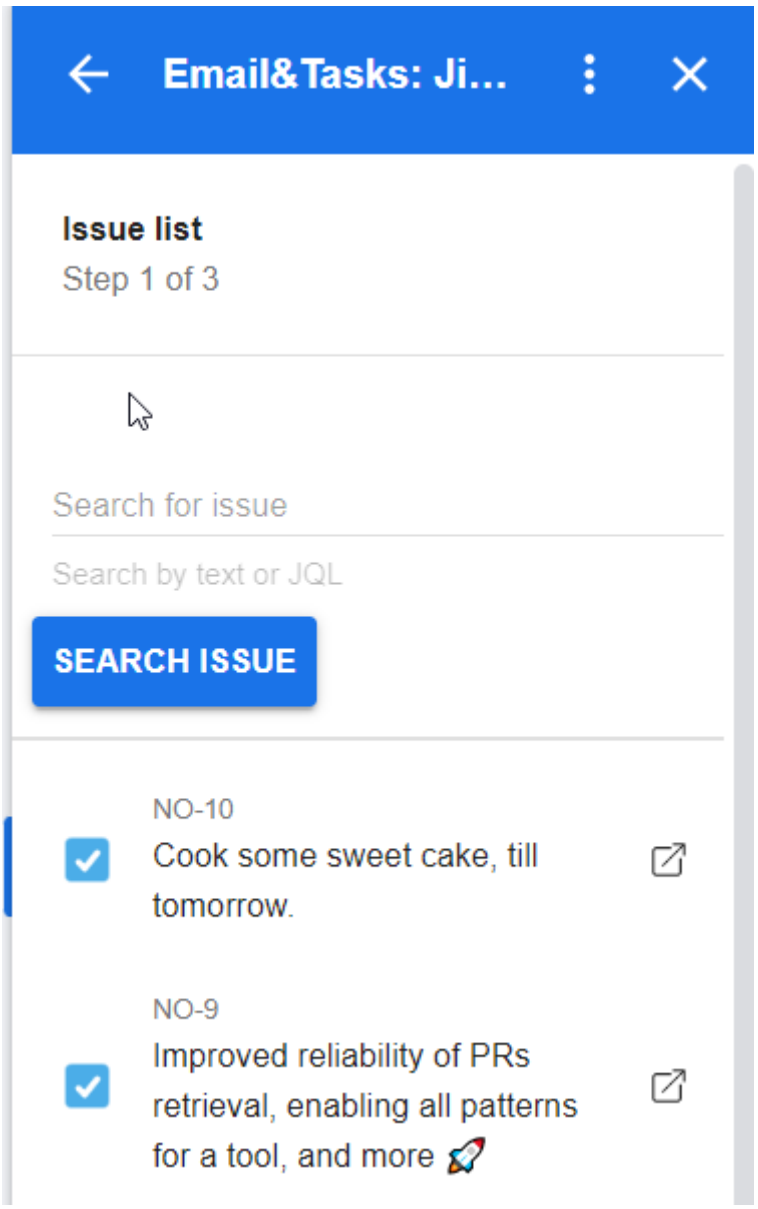

Рис.3.3. Картка пошуку задач

Також, в роботі даного плагіну використовується Google developer console, яка дозволяє йому взаємодіяти з сервісами Google.

3.3. Програмна реалізація плагіну для Jira

Для реалізації користувацьких плагінів для Jira є два способи:

- використовувати atlassian-connect
- Використовувати Forge

Даний плагін розроблений за першим способом, в його основі є клієнт і сервер, які пов’язані одним ключем додатку. Цей ключ має бути унікальним на Jira Marketplace. Для таких додатків обов’язковим є наявність файлу atlassian-connect.json в корені проекту. Цей файл містить наступні дані:

- key - ключ плагіну
- name - назву плагіну
- description - опис
- vendor - постачальник продукту
- baseUrl - посилання для встановлення плагіну
- authentication - метод аутентифікації користувачів в плагіні
- lifecycle - етапи використання плагіну
- scopes - список доступів плагіну в інстансі
- modules - кастомізація відображення плагіна в різних модулях Jira

Оскільки проект містить клієнтську і серверну частину, його необхідно зібрати, щоб оптимізувати, зменшити його фактичний розмір та задеплоїти на сервері. Для цього використовується webpack - збірник модулів.

Приклад фрагменту коду з основного файлу серверної частини:

```
import express from 'express';
import path from 'path'
import config from '../config'
import loggerOpts from '../logerOpts'
import { Connect } from 'sjconnect';
import db from "./service/db";

(async () => {
   const app = new Connect({
      config,
      loggerOpts
    });
```

```
    const { default: routes } = await import('./custom_routes');
    app.use(express.static(path.join(__dirname, '..', 'client', 'src')));
    app.use(routes);
    const dbConnect : any = await app.getDBConnection();
    db.init(dbConnect);
    await SJapp.start();
})();
```

За допомогою модуля Connect створюється екземпляр додатку, що містить в собі обширний функціонал для роботи з ним. Далі створюються роути серверу, які віддаватимуть основні сторінки та працюватимуть як API, для отримання користувачами даних. Нижче наведено приклад одного з роутів:

```
router.post('/api/give-feedback', async (req, res) => {
    const feedbackSystem = new FeedbackSystem('E2T', runConfig);
    await feedbackSystem.initRoute(req, res);
})
```

Клієнтська частина містить в собі структурований React.js проект, його структура показана на Рис.3.4. Додаток поділяється на реактивні компоненти, що відображаються на інтерфейсі користувача, локальне сховище даних та роутер з основними сторінками. На Рис.3.5 показано зовнішній вигляд основної сторінки плагіну, що містить невеликий опис програмної системи.

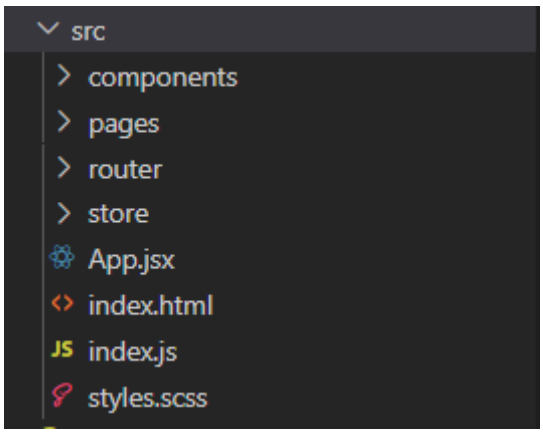

Рис.3.4. Структура клієнтської частини плагіну

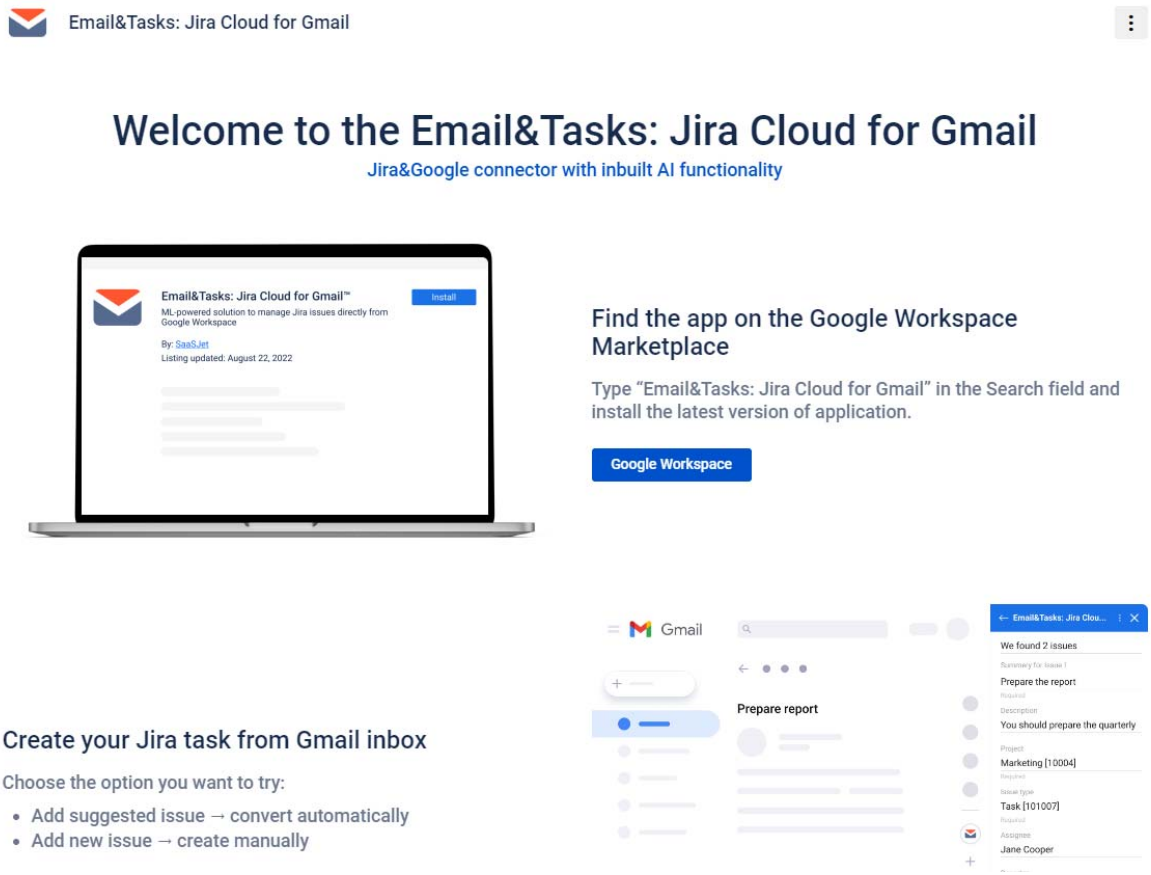

Рис.3.5. Вигляд основної сторінки плагіну для Jira

3.4. Програмна реалізація штучного інтелекту

Для початку розробки штучного інтелекту, що аналізуватиме листи, необхідно встановити Label Studio. Це можна зробити кількома способами,

одним з них є запуск команди pip install label-studio в терміналі. Процес встановлення зображений на Рис.3.6.

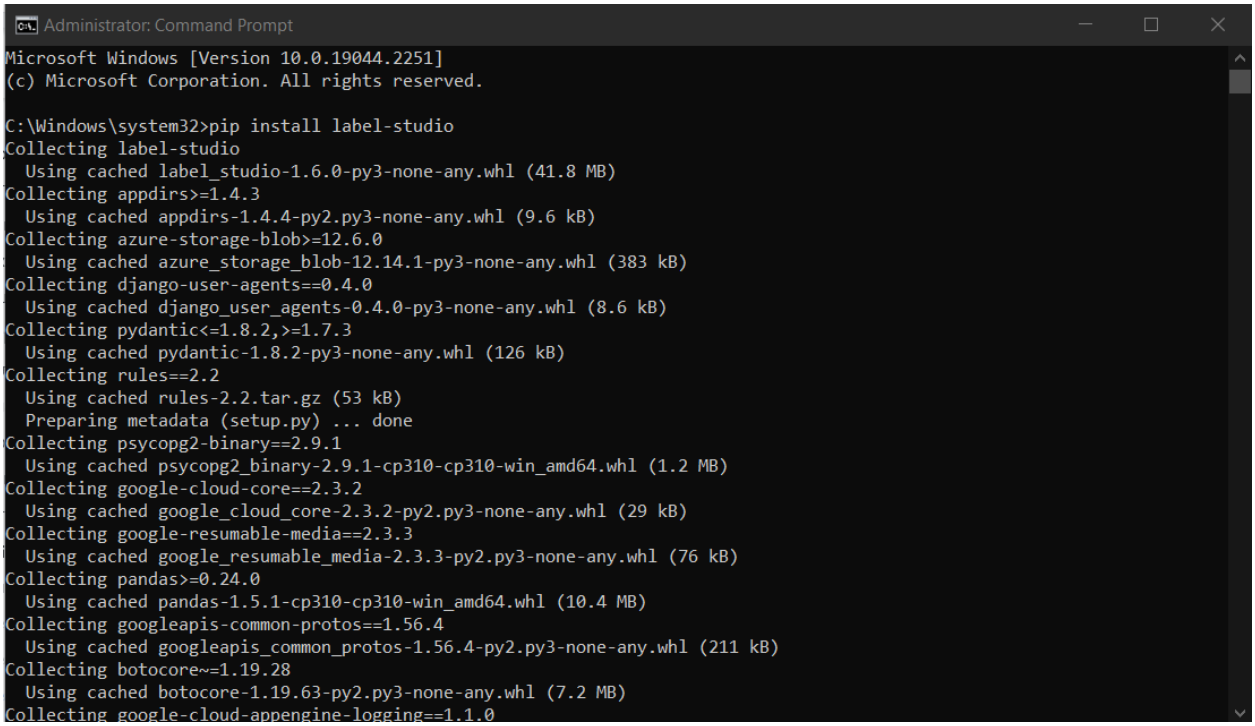

Рис.3.6. Процес встановлення Label Studio

Далі, для запуску використовуємо команду label-studio start. Після виконання, на локальному порті 8080 відкриється клієнт програмної системи, який зображений на Рис.3.7.

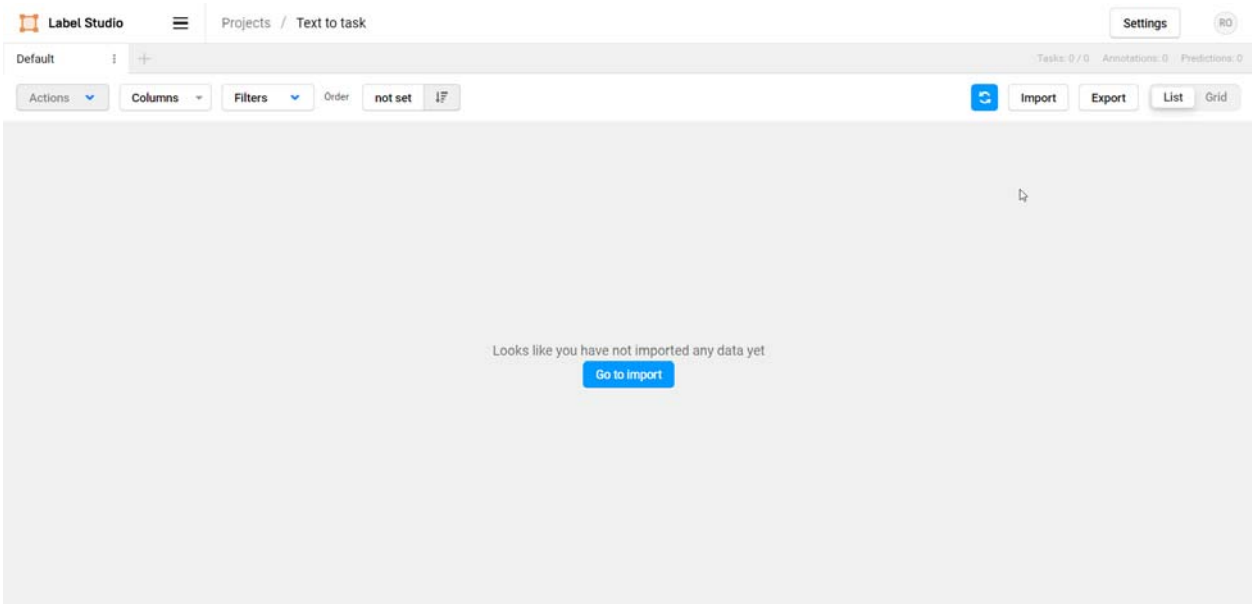

Рис.3.7. Інтерфейс Label Studio

Для продовження роботи, необхідно перейти в Settings, Labeling interface. Тут ми можемо створити користувацький інтерфейс для візуалізації проставлених лейблів до тексту, лейбли - умовні виділення тексту, який відповідає за конкретний контекст задачі, для прикладу, для такого тексту:

Hi, Mark. Can you complete your project till 12 november?

Можна проставити наступні лейбли:

- Mark - ASSIGNEE
- complete your project - SUMMARY/DESCRIPTION
- till 12 november - DUE-DATE

Для цього, необхідно створити необхідні лейбли за допомогою конструктора, або вручну, що дає більше можливостей для кастомізації. Щоб створити лейбли вручну, необхідно написати розмітку для користувацького інтерфейсу, нижче наведено приклад розмітки лейблів, а на Рис.3.8 зображено вигляд конструктора:

```
<Labels name="label" toName="text" showInline="false">
    <Label value="ASSIGNEE" background="red"/>
    <Label value="DUE-DATE" background="yellow"/>
    <Label value="DESCRIPTION" background="#FFA39E"/>
    <Label value="SUMMARY" background="#D4380D"/>
    <Label value="REPORTER" background="#FFC069"/>
    <Label value="STATUS" background="#AD8B00"/>
 </Labels>
```

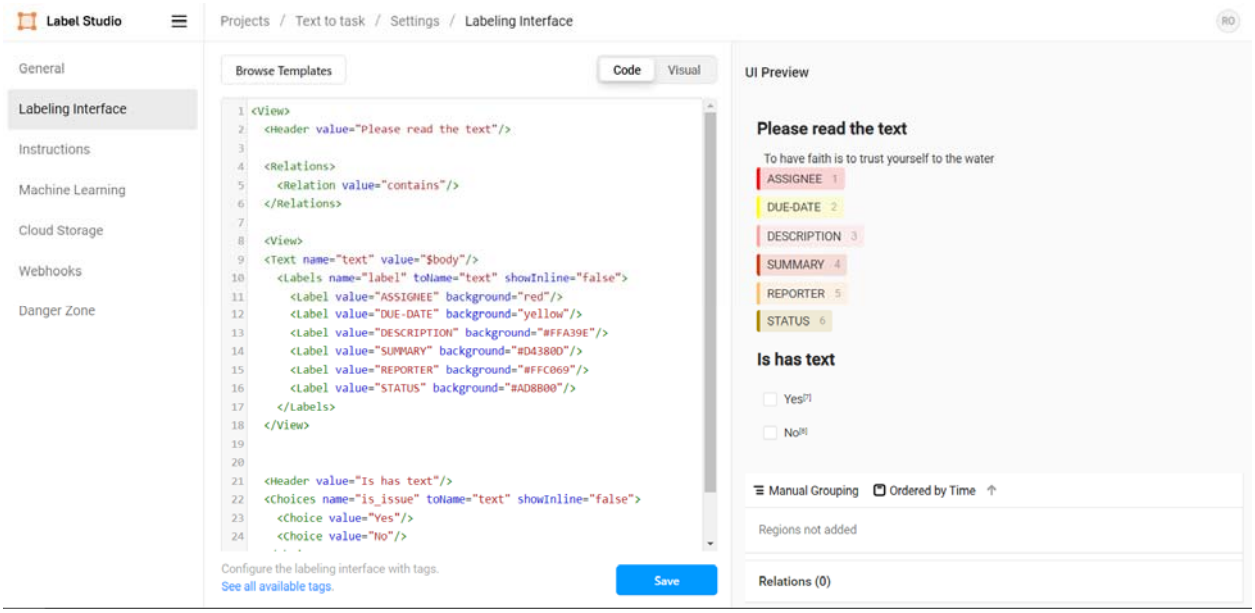

Рис.3.8. Вигляд конструктора користувацького інтерфейсу Label Studio

Після створення користувацького інтерфейсу, можна завантажити тестові дані для перевірки роботи, для цього використовується імпорт даних на головній сторінці. Підтримується імпорт даних різних типів, зокрема json, csv, xlsx. Після імпортування, можна зайти в будь який запис та проставити в ньому лейбли вручну, на Рис.3.9 зображено вигляд тексту з уже проставленими лейблами.

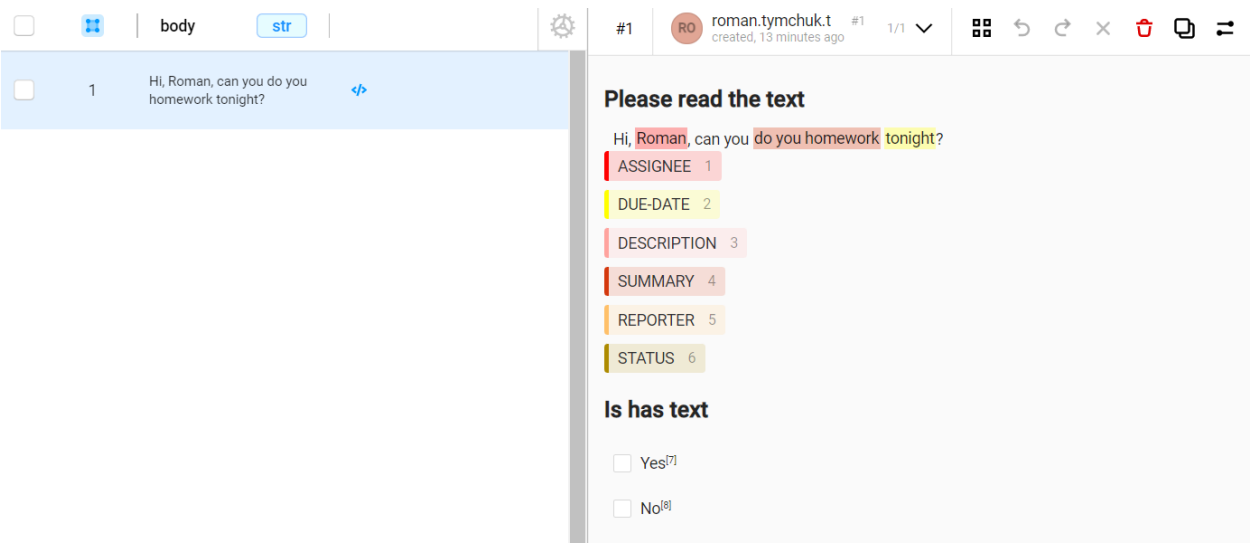

Рис.3.9. Вікно тексту з проставленими лейблами

Наступним кроком є підключення до Label Studio нейронної мережі, яка буде навчатись на отриманих даних - вона отримуватиме записи без лейблів - виставлятиме їх згідно поточного алгоритму і калібруватиметься після порівняння своїх результатів з очікуваними. Для цього, необхідно використати Label Studio API, щоб підключити сторонню програмну систему до необхідних даних.

Сама нейронна мережа виконуватиме завдання проставлення лейблів, після чого, ці лейбли будуть переведені в необхідні поля задачі і відправлені на сервер, для взаємодії з Jira API. Для навчання використовується модель GCFlair, з датасетом Co DDS & SUP з середньою кількістю циклів 500. Під час навчання калібрується велика кількість ваг нейронної мережі, нижче наведений список деяких з них:

- Weighted Precision
- Reporter Recall
- Assignee Precision
- Due Date Precision
- Reporter Title Support
- Date Context Recall

Значення кожного з ваг коливається між 0 та 1 і впливає на результат проставлення лейблів нейромережею. Окрім проставлення лейблів, ML має визначити чи є в тексті постановка задачі взагалі. Для цього є окремі ваги та моделі, в Label Studio з цією ціллю є поля “Is has text” та “Is task”.

Після кількох ітерацій навчання, можна побачити результати успішності штучного інтелекту в виконанні поставленої задачі. В таблиці 3.2 наведено результативність визначення деяких елементів в тексті (на основі тестування великої вибірки даних)

Таблиця 3.2

Відсоток точності визначення конкретних контекстів.

Speech act	Dataset size (emails)	Accuracy
Question	1000	92%
Regret	200	100%
Warning	1000	98%
Agreement	500	99%
Statement	1000	92%

Після кожної нової ітерації ML значення ваг можуть змінюватись, як до більш коректного значення, так і до помилкового, нижче наведено приклад зміни ваг “Rouge1 Precision”, що використовуються для визначення Description поля (в хронологічному порядку, від старіших ітерацій до актуальних):

- 0.3539050165
- 0.3473214845
- 0.4020513209
- 0.8577445774
- 0.7968051767
- 0.7753284673

Також, штучний інтелект, до створення задачі, повинен визначити проект в якому її потрібно створити. Наприклад: Development, Support, Admin, General, чи інші проекти які присутні на інстансі. Це непроста задача, оскільки на різних інстансах багато проектів з різними назвами, що ускладнює

визначення призначення того чи іншого проекту. Тому, за неможливості визначення, користувач сам обирає необхідний проект.

Висновки до третього розділу

У третьому розділі проведено розробку двох взаємопов’язаних плагінів для Jira та Gmail сервісів. Також розроблено штучний інтелект для лейблингу текстів, проведено його навчання та калібрацію.

Реалізація програмного продукту включає в себе два плагіни для сервісів Jira та Gmail. Інтерфейси програм розроблені за допомогою сучасних фреймворків та інструментів. Мовами програмування для реалізації програмного рішення є javascript, typescript та python.

ВИСНОВКИ

У магістерській роботі розроблено програмне забезпечення для розв’язання проблеми автоматизації створення задач Jira за допомогою аналізу листів нейромережею та проведено дослідження методів аналізу тексту електронних повідомлень для виявлення в ньому постановки задачі. Отримано такі наукові та практичні результати:

1. Проведено аналіз актуальності даного програмного забезпечення серед користувачів Jira.
2. Досліджено методи аналізу тексту листа за допомогою штучного інтелекту.
3. Побудовано алгоритм навчання нейромережі, а також проведено її тренування на реальних даних.
4. Розроблено програмне забезпечення для вирішення поставленої проблеми з максимальним коефіцієнтом корисної дії.