**Лейбюк Михайло Степанович,**

студент, Національний університет,

"Львівська політехніка", м. Львів, Україна,

ORCID: 0009-0007-9010-010X

**Науковий керівник: Коротєєва Тетяна Олександрівна**,

кандидат технічних наук, доцент,

Національний університет "Львівська політехніка",

м. Львів, Україна

**АЛГОРИТМ СТВОРЕННЯ ТЕХНІЧНОЇ ДОКУМЕНТАЦІЇ НА ОСНОВІ АНАЛІЗУ**

**ПРОГРАМНОГО КОДУ ЗАСОБАМИ ШТУЧНОГО ІНТЕЛЕКТУ**

Документація є невід'ємною частиною процесу розробки, оскільки вона надає важливу інформацію про функціональність, структуру і взаємозв'язки компонентів програмного продукту. Проте, написання документації є складним завданням, що вимагає великих затрат часу і зусиль з боку розробників. Для створення документації використовують різні засоби. З яких є відстежування та написання документацію власноруч, що потребує великої кількості часу та зусиль для її подальшої підтримки. Написання з використанням різних засобів які використовують додаткові маркери в коді, які потрібно власноруч залишати та оновлювати, це зручно тим що це можна додавати протягом розробки. Проте така документація буде більш технічною. Для розв'язання цієї проблеми пропонується використати штучний інтелект. За допомогою якого документація стане більш людською та зрозумілою для людей не що мають потрібних знань у відповідній сфері. Також штучному інтелекту можна задавати потрібний тип документації для кожної кодової бази окремо. Що дозволить документації бути більш специфічною та налаштовуватись під потребою програмної системи. Результати показали що алгоритм створює документацію менше ніж за хвилину, що в декілька разів швидше ніж створення її вручну та із застосуванням маркерів. Майбутні дослідження можуть передбачати вдосконалення алгоритму для роботи із пов'язаними файлами.

***Ключові слова:*** документація, штучний інтелект, проект, кодова база, технічна документація

**Вступ / Introduction**

Документація кодової бази є невід'ємною частиною кожного проєкту та є однією з ознак успішної програмної системи. Вона допомагає розробнику краще розуміти вже написаний код та пов'язані з ним властивості, що, своєю чергою, збільшує впевненість розробника в розробці високоякісного продукту. Це, своєю чергою, підвищує якість комунікації між розробниками, менеджерами та клієнтами.

Наразі на ринку є безліч різних систем, які допомагають у створенні та підтримці документації на основі кодової бази. Вони можуть використовувати різні алгоритми, такі як сканування коду, використання маркерів чи застосування методологій у процесі розробки програмної системи. Однак кожне з цих рішень потребує втручання людини в кодову базу та розуміння коду як такого і як він працює. На основі цього було визначено актуальність даної роботи з удосконалення алгоритму генерування документації із застосуванням штучного інтелекту.

*Об’єктом дослідження* є алгоритм створення технічної документації на основі аналізу програмного коду з використанням засобів штучного інтелекту.

*Предметом дослідження* є використання засобів штучного інтелекту для аналізу кодової бази та автоматичної генерації документації.

*Метою роботи* є розробка програмного додатка, який здатний автоматично генерувати документацію для програмного забезпечення на основі аналізу кодової бази з використанням штучного інтелекту.

*Завдання дослідження* що полягає в:

1. Огляд та аналіз наукових праць у сфері створення документації із застосування штучного інтелекту на основі кодової бази;
2. Проектування та розробка алгоритму який використовуватиме ШІ для генерації документації;
3. Розробка програмного забезпечення, яке реалізує розроблений алгоритм;
4. Аналіз результатів ефективності алгоритму.

**Аналіз останніх досліджень та публікацій.** Автори статей [1] та [2] визначили, що проекти з поганою документацією провалюються у 70% випадків. Через нечіткість документації вся команда може бути розчарована і навіть відмовитися від проекту. Схожі твердження були досліджені у роботі [3], де було досліджено вплив неякісної документації на витрати часу на розробку програмної системи. У дослідженні [4] стверджується, що чим якісніша документація, тим більше впевненості мають розробники, що покращує комунікацію та зменшує кількість помилок.

Автори статті [5] проаналізували вплив різних атрибутів документації. Залежно від атрибутів, кожна документація матиме унікальну зрозумілість та корисність. Подібне дослідження провели автори [6], які виявили, що додавання нових атрибутів до вже створеної документації потребує більше зусиль для підтримки застарілої документації.

Дослідження [7] розглянуло генерування документації із застосуванням GNN моделі та глибокого навчання. Така документація мала недолік у нестабільності тексту: його було то забагато, то замало. Автори дослідження [8] спробували розв'язати цю задачу за допомогою міток. Це підвищило якість моделі та покращило вибірку даних.

**Результати дослідження та їх обговорення**

Для реалізації алгоритму генерації документації із застосування штучного інтелекту. Було вирішено використовувати GPT-4 як штучний інтелект через те, що він має зручний інтерфейс для роботи. Також ця модель уже навчена на багатьох даних що дозволяє нам пропустити цей крок навчання.

Сам алгоритм складатиметься з 4 етапів:

1. Визначення структури проекту, його тип, мову на якій він написаний та які бібліотеки використовуються. Для цього перед початком роботи системи буде надіслано структура проекту до ШІ із запитом чи потрібна якась додаткова інформація для визначення типу проекту. Ця структура буде задаватись як контекст для покращення генерування документації
2. Обробка вибраних файлів користувачем перед надсилання до ШІ. З файлів потрібно взяти їх код видалити зайві відступи та символи, у разі необхідності розбити великі файли на менші для полегшення роботи з ШІ, це полегшить аналіз коду для штучного інтелекту.
3. Далі код з файлів буде надіслано до ШІ для його аналізу та генерування документації. Для цього використовуватиметься API інтерфейс за допомогою якого відбуватиметься спілкування з ШІ. Запит до ШІ включатиме код файлу, назву файлу, структуру документації та сам запит на генерацію документації
4. Останнім кроком буде отримання відповіді від штучного інтелекту та збереження даної інформації у відповідному місці. Через те, що кожний файл буде оброблено окремо потрібно вважати на правильність зберігання документації.

Отже, пропонується алгоритм, створення технічної документації на основі аналізу кодової бази із застосування штучного інтелекту. Блок-схему алгоритму подано на рисунку 1.



Рис. 1. Блок схема алгоритму.

На базі цього алгоритму було створено тестову програмну систему. Проведено різні експерименти з різними типами проектів.

**Обговорення результатів дослідження.** В ході експериментів із тестовою системою виявилось що для створення документації з одного файлу потребується від 30 секунд до 2 хвилин, залежно від кількості символів у файлів та швидкості інтернету. Таким чином в середньому на створення однієї документації в середньому потрібно 1 хвилину що значно зменшує витрати часу на створення документації в порівнянні зі створенням документації власноруч.

**Висновок**

 В даній роботі було проаналізовано актуальні джерела розглянуто вплив документації на проект, розглянуто різні застосування ШІ для генерації документації. Було розроблено алгоритм для генерації документації та програмну систему для тестування. В ході тестування було виявлено що на створення документації в середньому потрібно 1 хвилина що є значно швидше за ручне створення документації.

 Наступні дослідження можуть включати покращення алгоритму для роботи із більшими системами, додавання зв'язаності документація де документації посилається на уже створену документацію.

**Посилання**

1. Behutiye W., Rodríguez P., Oivo M. Quality requirement documentation guidelines for agile software development. *IEEE access*. 2022. Vol. 10. P. 70154–70173.
2. Cost, benefits and quality of software development documentation: a systematic mapping / J. Zhi et al. *Journal of systems and software*. 2015. Vol. 99. P. 175–198. URL: <https://doi.org/10.1016/j.jss.2014.09.042> (date of access: 16.05.2024).
3. Deep code comment generation / X. Hu et al. *ICSE '18: 40th international conference on software engineering*, Gothenburg Sweden. New York, NY, USA, 2018. URL: <https://doi.org/10.1145/3196321.3196334> (date of access: 19.05.2024).
4. Documentation matters: human-centered AI system to assist data science code documentation in computational notebooks / A. Y. Wang et al. *ACM transactions on computer-human interaction*. 2022. Vol. 29, no. 2. P. 1–33. URL: <https://doi.org/10.1145/3489465> (date of access: 19.05.2024).
5. Documentation matters: human-centered AI system to assist data science code documentation in computational notebooks / A. Y. Wang et al. *ACM transactions on computer-human interaction*. 2022. Vol. 29, no. 2. P. 1–33. URL: <https://doi.org/10.1145/3489465> (date of access: 19.05.2024).
6. Hassani R., EL Bouzekri EL Idrissi Y. Normalization of requirements specification document on software project management. *Journal of software*. 2018. Vol. 13, no. 4. P. 232–241. URL: <https://doi.org/10.17706/jsw.13.4.232-241> (date of access: 16.05.2024).
7. Hassani R., EL Bouzekri EL Idrissi Y. Normalization of requirements specification document on software project management. *Journal of software*. 2018. Vol. 13, no. 4. P. 232–241. URL: <https://doi.org/10.17706/jsw.13.4.232-241> (date of access: 16.05.2024).
8. The dataset nutrition label / S. Holland et al. *Data protection and privacy*. 2020. URL: <https://doi.org/10.5040/9781509932771.ch-001> (date of access: 19.05.2024).