УДК 004.8 DOI https://doi.org/10.35546/kntu2078-4481.2025.1.2.15

В. І. КОЦУН
кандидат технічних наук, доцент,
доцент кафедри комп'ютерних наук та програмної інженерії
ПВНЗ «Європейський університет»
ORCID: 0000-0003-2363-8157

Р. В. СЕНИШИН
аспірант
ПВНЗ «Європейський університет»
ORCID: 0009-0007-7900-8679

ДОСЛІДЖЕННЯ ВПРОВАДЖЕННЯ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПОДОЛАННЯ ПРОБЛЕМИ ВТРАТИ РЕСУРСІВ ТА ЧАСУ У СУЧАСНОМУ ЦИКЛІ РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ

У статті проаналізовано сучасний життєвий цикл розробки веб-застосунків, що поєднує методології Agile та практики DevOps, і виокремлено основні причини перевитрат ресурсів та часу. Автор звертає увагу на часто змінювані або нечіткі вимоги, накопичення технічного боргу внаслідок рутинних завдань і нестачі часу на рефакторинг, проблеми з автоматизацією тестування та складністю відтворення дефектів, а також на недосконалу комунікацію у великих чи розподілених командах. У контексті цих викликів розглянуто роль генеративного штучного інтелекту (ГШІ), що здатен значно прискорити виконання рутинних операцій, допомогти у написанні коду, тестуванні та навіть пропонувати проектні рішення.

Аналіз останніх досліджень засвідчує, що, попри широке впровадження гнучких (Agile) підходів і DevOps-практик, ефективна синергія з інструментами ГШІ залишається фрагментарно вивченою. У традиційних роботах із проектного менеджменту та розробки (Pressman & Maxim, Sommerville) велика увага приділяється процесам збирання вимог і управління змінами, проте роль генеративних моделей там або не розглядається, або згадується поверхнево. Натомість у публікаціях, присвячених ГШІ (Chen та ін., OpenAI, GitHub Copilot, Amazon CodeWhisperer), бракує комплексного аналізу впливу цих технологій на весь цикл веб-розробки – від постановки вимог до розгортання та супроводу.

Основна мета статті – оцінити, як саме генеративний ШІ здатен долати «вузькі місця», що зумовлюють найбільші втрати часу й ресурсів. Автор демонструє, що інтеграція ГШІ в Agile-спринти може пришвидшити формування backlog та генерацію тест-кейсів, а в контексті DevOps – полегшити автоматизацію CI/CD-процесів та інфраструктури. Окрім того, стаття висвітлює ризики: безпекові уразливості в згенерованому коді, неточність чи «галюцинації» моделей та потребу додаткових перевірок якості й безпеки.

Зроблено висновок, що генеративний штучний інтелект може стати вагомим каталізатором ефективності веб-розробки, суттєво скорочуючи рутинні операції, зменшуючи кількість дефектів і підвищуючи швидкість реагування на зміни. Подальші дослідження доцільно спрямувати на розробку методологій із вимірювання ROI від використання ГШІ, підвищення безпеки та точності згенерованого коду, а також на формування практичних рекомендацій щодо адаптації Agile/DevOps-процесів під можливості генеративних мовних моделей.

Ключові слова: *генеративний штучний інтелект, Agile, DevOps, технічний борг, автоматизація тестування, CI/CD, мовні моделі, GitHub Copilot, ChatGPT, ефективність розробки, веб-застосунки.*

V. I. KOTSUN
Candidate of Technical Sciences (Ph.D.), Associate Professor,
Head of the Department of Mathematics and Computer Sciences
Private Higher Educational Institution "European University"
ORCID: 0000-0003-2363-8157

R. V. SENYSHYN
Postgraduate Student
Private Higher Educational Institution "European University"
ORCID: 0009-0007-7900-8679

RESEARCH ON THE IMPLEMENTATION OF GENERATIVE ARTIFICIAL INTELLIGENCE TO OVERCOME THE ISSUE OF RESOURCE AND TIME LOSS IN THE MODERN WEB APPLICATION DEVELOPMENT CYCLE

In this article, a comprehensive analysis is presented of the modern web application development lifecycle, which typically combines Agile methodologies with DevOps practices, revealing the primary factors leading to resource and time overruns. The author points to ambiguous or frequently changing requirements, the accumulation of technical debt due to routine tasks and insufficient time for refactoring, inadequate test automation and challenges with reproducing defects, as well as communication difficulties in teams–especially distributed ones. Within this context, the article examines the role of generative artificial intelligence (GAI), capable of automating routine operations, writing or enhancing code, generating test cases and documentation, and in some cases providing design solutions.

A review of scientific and practical publications shows that although Agile methodologies, DevOps practices, and continuous integration and delivery (CI/CD) tools receive considerable attention, the synergy of these approaches with the capabilities of generative AI has not been adequately explored. Notably, there is no complete overview of how AI can influence all phases of web development – from requirement gathering and alignment to product deployment and support. The author underscores the necessity of a systemic approach that encompasses both technological and organizational dimensions, also highlighting the risks linked to GAI usage (security vulnerabilities, potential "hallucinations," and licensing issues).

The central goal of this article is to evaluate the effectiveness of generative AI in addressing the "bottlenecks" in web development that lead to significant time losses. It is demonstrated that integrating GAI into Agile sprints can expedite backlog formation and the generation of both code and tests, while its inclusion in DevOps workflows simplifies the automation of CI/CD pipelines and infrastructure configuration. In addition, the article provides practical recommendations for thorough code review and the adaptation of project management methodologies to accommodate AI-generated content.

The conclusion is that generative AI can substantially boost developer productivity, reduce repetitive tasks, and shorten release cycles when integrated with well-established Agile/DevOps processes. Future research should focus on real-world usage scenarios of GAI, the development of performance metrics (e.g., ROI), and the creation of guidelines for the safe and legally compliant use of language models. Given the rapid evolution of contemporary LLM solutions, establishing unified quality standards for generated code and standardized training for development teams remains highly relevant. Such a holistic approach will enable the full realization of GAI's potential while minimizing security and regulatory risks.

Key words: *generative artificial intelligence, Agile, DevOps, technical debt, test automation, CI/CD, language models, GitHub Copilot, ChatGPT, development efficiency, web applications.*

Постановка проблеми

У сучасному життєвому циклі розробки веб-застосунків, який часто базується на інтеграції гнучких (Agile) методологій із DevOps-практиками, все ще спостерігаються суттєві втрати ресурсів та часу. Серед найпоширеніших факторів:

- Нечіткі або часто змінювані вимоги, що призводить до переробок і перевитрат зусиль;
- Накопичення технічного боргу, обумовлене рутинними завданнями та браком часу на рефакторинг;
- Проблеми з тестуванням, зокрема низька автоматизація та складність відтворення дефектів;
- Недосконала комунікація у командах та складність масштабування процесів у великих чи розподілених командах.

На тлі цієї проблематики з'являється генеративний штучний інтелект (ГШІ), який здатен автоматизувати низку рутинних операцій, допомагати у написанні коду, генерувати документацію та навіть пропонувати проектні рішення. Однак масштаби впровадження ГШІ та його реальний вплив на оптимізацію ресурсів і часу поки що недостатньо досліджені. Тож мета цієї статті – проаналізувати роль генеративного ШІ в подоланні «вузьких місць» сучасної веб-розробки й оцінити потенційні вигоди від його застосування.

Аналіз останніх досліджень і публікацій

Актуальність проблематики оптимізації розробки веб-застосунків через мінімізацію втрат часу і ресурсів висвітлюється у низці наукових та практичних публікацій. Проте, як свідчить аналіз джерел, багато досліджень зосереджені переважно на окремих аспектах (наприклад, методології проектного менеджменту чи виключно автоматизованому тестуванні), тоді як комплексне вивчення ролі генеративного штучного інтелекту (ГШІ) у подоланні відомих «вузьких місць» розробки залишається фрагментарним. Нижче подано більш деталізований огляд літератури й виявлені обмеження існуючих праць:

Роботи Pressman & Maxim [1] та Sommerville [2] подають фундаментальний опис життєвого циклу ПЗ, починаючи від збору вимог до впровадження та супроводу продукту. У цих джерелах розкриваються класичні проблеми – невідповідність вимог, пізнє виявлення дефектів, складність управління змінами. Також автори висвітлюють переваги гнучких (Agile) підходів у порівнянні з каскадними моделями.

У зазначених працях роль інструментів штучного інтелекту розглядається або поверхнево, або взагалі не згадується. Хоча автори визнають важливість автоматизації, сфера генеративного ШІ залишається за рамками їхньої уваги.

Рис. 1. Найпоширеніші фактори в сучасному життєвому циклі розробки веб-застосунків

У працях Beck & Andres [3], Schwaber & Sutherland [4] ґрунтовно описано еволюцію гнучкої розробки, зокрема Scrum-підходи до спринтового управління, самоорганізації команд, швидкого зворотного зв'язку. У той же час DevOps-ініціативи (за Humble & Farley [5]) демонструють, як безперервна інтеграція та доставка (CI/CD) можуть скоротити цикл виходу продукту на ринок. Хоча в цих працях наведено низку рекомендацій для уникнення перевитрат і затримок, окремий аналіз впливу генеративного ШІ на скорочення часу виконання рутинних завдань чи на узгодження вимог у реальному масштабі поки що відсутній.

Дослідження щодо технічного боргу і складності проектів Boehm & Turner [7] та Larman & Vodde [6] наголошують, що неконтрольоване зростання технічного боргу, особливо в масштабних чи розподілених командах, призводить до значних перевитрат у майбутньому. Вони описують процеси оцінювання боргу та підходи до його зменшення (рефакторинг, контроль код-рев'ю), але здебільшого у традиційному контексті Agile/Lean. Ці дослідження обмежені тим, що не розглядають можливості ГШІ для автоматизованого виявлення зон технічного боргу чи автогенерації тест-кейсів і рекомендацій щодо рефакторингу коду.

Whittaker [8] (досвід Google) і Meszaros [9] (xUnit Test Patterns) широко висвітлюють питання побудови тестових стратегій: від модульного і інтеграційного тестування до складних end-to-end сценаріїв. Автори акцентують на економії ресурсів завдяки ефективній автотест-стратегії. Попри детальний опис патернів та інструментів тестування, взаємодія ГШІ з процесом тестування поки що майже не досліджена, а можливість генеративної моделі створювати та оновлювати тести на основі змін у коді залишається поза межами аналізу.

Chen та ін. [10] досліджували ефективність мовних моделей (Codex) у генерації програмного коду, демонструючи позитивний вплив на швидкість написання тестових і допоміжних фрагментів. OpenAI [11], GitHub Copilot [12] і Amazon CodeWhisperer [13] також представляють результати власних експериментів, показуючи потенціал економії часу розробників. Хоча ці праці й оглядають можливості ГШІ, дослідження здебільшого фрагментарні. Зокрема, немає повного аналізу, як генеративні моделі змінюють увесь життєвий цикл веб-розробки (від збору вимог до супроводу), які саме метрики ефективності (ROI, швидкість релізів, зменшення дефектів) найбільш релевантні, і наскільки стабільним є цей ефект у довгостроковій перспективі.

Окремі статті та блоги (Kim et al. [14], Sea & Nguyen [15]) згадують «інтелектуальний DevOps», де машинне навчання допомагає аналізувати логи виробничих середовищ і налаштовувати пайплайни CI/CD. Але про роль генеративних моделей у проєктуванні архітектури, створенні документації чи управлінні вимогами інформації мало. У більшості випадків автори зосереджуються на аналізі даних чи прогнозних алгоритмах, тоді як генеративний аспект (автоматичне створення коду, тестів, документації) залишається мало дослідженим.

Неповнота існуючих досліджень зображена на рисунку 2.

Таким чином, попри те, що в останні роки у науковому середовищі й практичній індустрії активно обговорюються питання автоматизації, Agile/DevOps та окремі аспекти застосування мовних моделей у розробці, повного

Рис. 2. Неповнота існуючих досліджень

та всебічного опису інтеграції генеративного ШІ у цикл веб-розробки поки що не запропоновано. Це обґрунтовує необхідність додаткового дослідження, спрямованого на розробку методологічного підходу до впровадження ГШІ, оцінку його впливу на різні етапи проєкту та формування рекомендацій для розробників і менеджменту.

Формулювання мети дослідження

Виявити ключові «вузькі місця» сучасного циклу розробки веб-застосунків, де відбувається найбільша втрата ресурсів та часу. Проаналізувати наявні генеративні рішення (GitHub Copilot, ChatGPT, CodeWhisperer тощо) й оцінити їхню роль у подоланні виявлених проблем. Запропонувати підходи та рекомендації щодо інтеграції генеративного ШІ в існуючі процеси розробки з метою мінімізації технічного боргу, прискорення тестування та зменшення рутинних завдань.

Викладення основного матеріалу дослідження

Основні чинники «загублення» ресурсів у життєвому циклі розробки

Для виявлення ділянок, де губляться ресурси, розглянемо традиційну структуру життєвого циклу веб-розробки з труднощами які виникають на кожному з етапів зображеному на рисунку 3.

Генеративний штучний інтелект (ГШІ), побудований на великих мовних моделях (LLM), має потенціал впливу та покращення різних етапів розробки веб-застосунків (Рис. 4).

Таким чином, ГШІ може стати каталізатором скорочення часу на рутинні завдання, зменшити кількість дефектів, сприяти швидшому реагуванню на зміни і покращити загальну продуктивність. Мною були виділені такі приклади інтеграції ГШІ у сучасні методології (Agile + DevOps):

Agile-спринт з ГШІ. На етапі планування спринту штучний інтелект може допомагати формувати backlog, генерувати детальні user stories та тест-кейси. Під час розробки інструмент на кшталт GitHub Copilot пропонує фрагменти коду, а ChatGPT дає поради щодо покращення алгоритмів або архітектурних рішень.

DevOps-пайплайн. Continuous Integration: ГШІ може згенерувати скрипти або YAML-файли для побудови збірок, запуску тестів і статичного аналізу коду.

DevOps-пайплайн. Continuous Deployment: автоматичне розгортання у хмарному середовищі, створення інфраструктури як коду (IaC).

Ретроспектива та вдосконалення. Застосування ГШІ для аналізу журналів (логів), тестових звітів і показників продуктивності дає змогу швидше виявляти системні помилки та «вузькі місця».

Попри відчутні переваги які несе використання ГШІ в повному циклі розробки веб-застосунків хочу також відмітити наступні проблеми та обмеження використання генеративного ШІ:

Безпека та якість коду. Генеративні моделі можуть пропонувати код із вразливостями або некоректними алгоритмами. Потрібен додатковий рівень перевірок (рецензія коду, інструменти статичного аналізу).

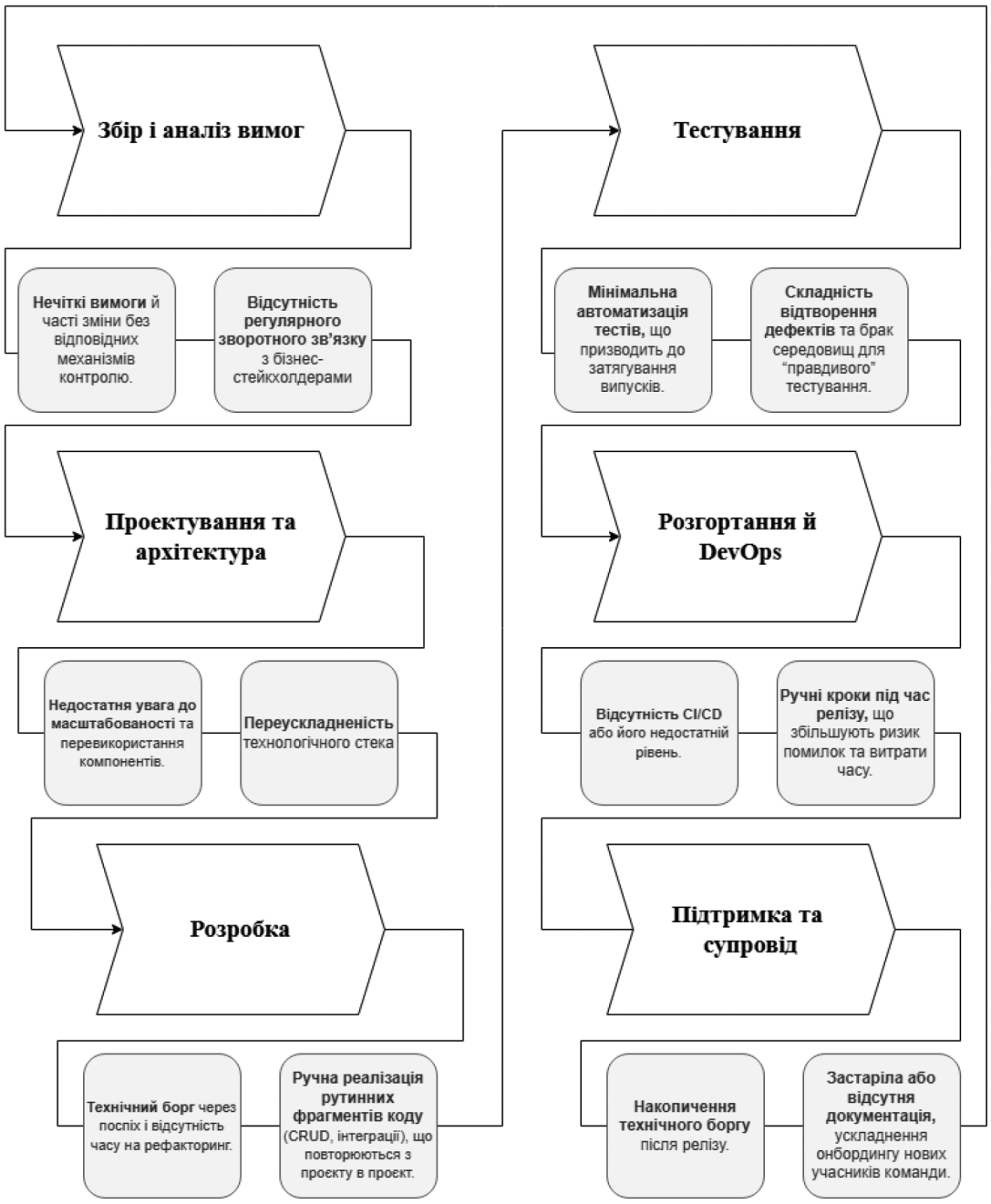

Рис. 3. Структура життєвого циклу веб-розробки

Інтелектуальна власність та ліцензування. Якщо модель навчена на відкритих репозиторіях, виникають суперечливі питання щодо використання уривків коду з різними ліцензіями.

Точність та релевантність. Можливі «галюцинації» моделей (коли вони генерують правдоподібний, але неправильний код). Потрібен досвідчений розробник, щоб фільтрувати пропозиції ГШІ.

Навчання та адаптація. Для точних підказок у специфічному домені (наприклад, фінтех, медичні дані) бажаний донавчальний етап моделі на внутрішньому коді та документації.

Висновки

Сучасний цикл розробки веб-застосунків часто страждає від перевитрат часу та ресурсів на етапах збору вимог, написання рутинних частин коду, тестування й розгортання.

Генеративний штучний інтелект відкриває можливості для оптимізації більшості стадій, починаючи від автоматизованого створення шаблонів коду та тест-кейсів до генерації конфігурацій для DevOps-середовищ. Найбільший ефект ГШІ спостерігається в галузях, де рутинні й повторювані завдання становлять великий

Рис. 4. Покращення етапів розробки веб-застосунків

відсоток роботи (CRUD-операції, типові інтеграції, тестування), а також у проектах з високим ступенем складності та швидко змінюваними вимогами.

Для успішного впровадження потрібно подбати про додаткові перевірки якості згенерованого коду (код-рев'ю, статичний аналіз, безпекові сканери), механізми відповідального використання (дотримання ліцензій, конфіденційності даних) та навчання команди та адаптацію процесів (Agile + DevOps + ГШІ).

Таким чином, інтеграція генеративного штучного інтелекту може суттєво скоротити втрати часу та ресурсів у розробці веб-застосунків, особливо якщо поєднати її з усталеними гнучкими методологіями та DevOps-практиками. Подальші дослідження у цій сфері можуть бути спрямовані на оцінку ефективності ГШІ в реальних проєктах, покращення точності та безпеки згенерованого коду, а також розробку методів вимірювання ROI від впровадження інтелектуальних моделей.

Список використаної літератури

1. Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner's Approach*. 9th Ed. McGraw-Hill Education.
2. Sommerville, I. (2016). *Software Engineering*. 10th Ed. Pearson.
3. Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change*. 2nd Ed. Addison-Wesley.
4. Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. [Онлайн]. Доступно: https://scrumguides.org/

5. Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley.
6. Larman, C., & Vodde, B. (2008). *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison-Wesley.
7. Boehm, B., & Turner, R. (2005). *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley.
8. Whittaker, J.A. (2012). *How Google Tests Software*. Addison-Wesley.
9. Meszaros, G. (2007). *xUnit Test Patterns: Refactoring Test Code*. Addison-Wesley.
10. Chen, M., Tworek, J., Jun, H., et al. (2021). Evaluating Large Language Models Trained on Code. *arXiv:2107.03374 [cs.LG]*.
11. OpenAI ChatGPT Official Documentation. [Онлайн]. Доступно: https://platform.openai.com/docs/introduction
12. GitHub Copilot. [Онлайн]. Доступно: https://github.com/features/copilot
13. Amazon CodeWhisperer. [Онлайн]. Доступно: https://aws.amazon.com/codewhisperer/
14. Kim, M., Kim, S., & Murugesan, S. (2021). Intelligent DevOps: Applying AI/ML to CI/CD pipelines. *IEEE Software, 38*(5), 81–87.
15. Sea, C., & Nguyen, T. (2022). Automated CI/CD pipeline generation using generative AI. *arXiv:2207.01123 [cs.SE]*.
16. Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution.
17. Sandoval, E. (2022). Intellectual Property Rights and AI-generated Code. *Journal of Business Law, 45*(2), 73–91.

References

1. Pressman, R.S., & Maxim, B.R. (2019). *Software Engineering: A Practitioner's Approach*. 9th Ed. McGraw-Hill Education.
2. Sommerville, I. (2016). *Software Engineering*. 10th Ed. Pearson.
3. Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change*. 2nd Ed. Addison-Wesley.
4. Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. [Online]. Available: https://scrumguides.org/
5. Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley.
6. Larman, C., & Vodde, B. (2008). *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison-Wesley.
7. Boehm, B., & Turner, R. (2005). *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley.
8. Whittaker, J.A. (2012). *How Google Tests Software*. Addison-Wesley.
9. Meszaros, G. (2007). *xUnit Test Patterns: Refactoring Test Code*. Addison-Wesley.
10. Chen, M., Tworek, J., Jun, H., et al. (2021). Evaluating Large Language Models Trained on Code. *arXiv:2107.03374 [cs.LG]*.
11. OpenAI ChatGPT Official Documentation. [Online]. Available: https://platform.openai.com/docs/introduction
12. GitHub Copilot. [Онлайн]. Доступно: https://github.com/features/copilot
13. Amazon CodeWhisperer. [Online]. Available: https://aws.amazon.com/codewhisperer/
14. Kim, M., Kim, S., & Murugesan, S. (2021). Intelligent DevOps: Applying AI/ML to CI/CD pipelines. *IEEE Software, 38*(5), 81–87.
15. Sea, C., & Nguyen, T. (2022). Automated CI/CD pipeline generation using generative AI. *arXiv:2207.01123 [cs.SE]*.
16. Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution.
17. Sandoval, E. (2022). Intellectual Property Rights and AI-generated Code. *Journal of Business Law, 45(2),* 73–91.