УДК 004.4

ПІДСИСТЕМА ПУБЛІКУВАННЯ ОКРЕМИХ ТЕСТІВ НА DMOJ ЗА ЗАПИТАМИ КОРИСТУВАЧІВ

Шафієв Д. Ю.
shafiiev.daniil1122@vu.cdu.edu.ua
Черкаський національний університет
імені Богдана Хмельницького
Порубльов І. М.
м. Черкаси, Україна

Системи автоматичної перевірки задач з програмування стали важливою складовою підготовки майбутніх фахівців з інженерії програмного забезпечення, оскільки забезпечують автоматизовану перевірку розв'язків, об'єктивність оцінювання та зручну взаємодію між користувачем і платформою. Однією з відкритих систем такого типу є DMOJ – Modern Online Judge, що підтримує багато мов програмування, може бути розгорнута локально та допускає розширення функціональності за рахунок додаткових модулів [1].

Актуальність дослідження зумовлена тим, що більшість сучасних онлайн-суддів повністю або частково приховують тестові дані. Такий підхід є виправданим у змагальному середовищі, однак у навчальному процесі він ускладнює аналіз помилок і знижує ефективність налагодження програм. Користувач зазвичай бачить лише узагальнений результат перевірки, наприклад Wrong Answer або Time Limit Exceeded, але не має змоги встановити, які саме вхідні дані спричинили помилку. Тому розробка механізму контрольованого надання окремих тестів за запитами користувачів є актуальним практичним завданням.

Метою роботи є розробка підсистеми публікування окремих тестів у системі DMOJ за запитами користувачів шляхом створення програмного модуля з механізмами керування доступом. Запропоноване рішення має поєднати дві вимоги: надати користувачеві більше даних для аналізу власного розв'язку та водночас не допустити довільного розкриття повного набору перевірочних матеріалів [1].

На першому етапі дослідження було проаналізовано сучасні системи автоматичної перевірки задач з програмування та підходи до керування тестовими даними. Доцільно розмежовувати централізовані онлайн-платформи, зокрема LeetCode, HackerRank та EOlymp [4–6], і open-source judge-системи, серед яких DMOJ, DOMjudge, Judge0 та PC² [1–3; 7]. Централізовані платформи надають готовий сервіс, але не дають змоги змінювати внутрішню логіку перевірки або впроваджувати власні підсистеми. Натомість open-source рішення можуть бути розгорнуті у власній інфраструктурі, підтримують адаптацію архітектури та є придатними для створення спеціалізованих розширень [1–3; 7].

Проведений аналіз показав, що жодна з розглянутих систем у типовій конфігурації не забезпечує гнучкого механізму вибіркового публікування окремих тестів за запитами користувачів. Водночас саме DMOJ має архітектурні передумови для реалізації такої функціональності: відкритий код, наявність ролей користувачів, інтеграцію із серверною логікою перевірки та роботу з файловою системою тестових даних [1]. Саме тому цю платформу обрано як базове середовище для проєктування та подальшої реалізації підсистеми.

На другому етапі було сформовано вимоги до підсистеми публікування окремих тестів. Моделювання бізнес-процесу показало, що новий функціонал не повинен змінювати стандартний механізм оцінювання посилки, а має розширювати сценарій роботи зі сторінкою результатів. Після завершення перевірки та виявлення неуспішних тестових випадків користувач повинен мати змогу ініціювати запит лише для конкретного тесту, для якого публікування дозволено встановленими правилами.

На основі користувацьких історій визначено ключові функціональні вимоги до підсистеми. Вона повинна відображати доступність запиту лише для тих тестів, які одночасно є неуспішними в межах конкретної посилки та позначені як відкриті до публікування. Після натискання відповідної кнопки система має автоматично визначити задачу, знайти потрібний ZIP-архів тестових даних, отримати вхідні дані та еталонний вихід для вибраного тестового випадку, додати фактичний результат користувача й сформувати текстовий файл для

завантаження. У результаті користувач отримує звіт, придатний для подальшого аналізу, без додаткових ручних дій.

Окрему увагу приділено нефункціональним вимогам. Підсистема повинна бути безпечною, тобто не допускати довільного доступу до файлів поза встановленою структурою архівів і не дозволяти публікування даних для тестів, які не відповідають визначеним умовам. Важливими також є продуктивність, надійність, сумісність із наявною архітектурою DMOJ, зручність використання та супроводжуваність [1]. Це означає, що формування файлу має відбуватися без помітної затримки для користувача, а внутрішня структура модуля повинна допускати подальше розширення правил доступу та формату звіту.

Для підтримки запропонованого функціоналу обґрунтовано розширення моделі даних. Підсистема спирається на вже наявні у DMOJ сутності користувача, задачі, посилки, тестового випадку та результату виконання окремого тесту [1]. Разом із цим до логічної схеми доцільно додати окрему таблицю правила публікування тесту, яка зберігатиме ідентифікатор тестового випадку, ознаку дозволу на публікування, момент останнього оновлення та користувача, який виконав зміну. Такий підхід дає змогу інтегрувати нову функціональність без дублювання наявних даних і без порушення стандартної логіки роботи платформи.

Отже, у результаті проведеного дослідження обґрунтовано доцільність розробки підсистеми публікування окремих тестів на DMOJ за запитами користувачів, визначено її місце в архітектурі платформи та сформовано основні вимоги до реалізації. Запропоноване рішення має практичне значення для освітнього процесу, оскільки поєднує контрольований доступ до тестових даних із збереженням принципів чесного оцінювання [1]. Упровадження такої підсистеми сприятиме глибшому аналізу помилок, підвищенню якості студентських розв’язків і розширенню функціональних можливостей сучасних систем автоматичної перевірки програм.

Список використаних джерел:

1. DMOJ: Modern Online Judge. URL: https://dmoj.ca/ (дата звернення: 20.03.2026).
2. DOMjudge. Programming Contest Jury System. URL: https://www.domjudge.org/ (дата звернення: 20.03.2026).
3. Judge0. Open-source online code execution system. URL: https://judge0.com/ (дата звернення: 20.03.2026).
4. LeetCode. The world’s leading online programming learning platform. URL: https://leetcode.com/ (дата звернення: 20.03.2026).
5. HackerRank. Online coding tests and technical interviews. URL: https://www.hackerrank.com/ (дата звернення: 20.03.2026).
6. EOlymp. Website dedicated to competitive programming, algorithms and problem solving. URL: https://eolymp.com/ (дата звернення: 20.03.2026).
7. PC² Contest Control System. URL: https://pc2ccs.github.io/ (дата звернення: 20.03.2026).

УКД 004.415.2

АРХІТЕКТУРНІ ПІДХОДИ ДО ПРОЄКТУВАННЯ ЧАТ-БОТІВ У СУЧАСНИХ ПРОГРАМНИХ СИСТЕМАХ

Антоненко А.Я
annantonenko4567@gmail.com
Черкаський державний фаховий бізнес-коледж
Немченко В.Ю.
м. Черкаси, Україна

Активний розвиток технологій штучного інтелекту, обробки природної мови та хмарних обчислень сприяв впровадженню чат-ботів у сучасні програмні системи. Сьогодні такі рішення використовуються для автоматизації взаємодії з користувачами, надання довідкової інформації, підтримки клієнтів та виконання типових операцій у веб-сервісах, мобільних застосунках і корпоративних платформах. У зв’язку з цим особливого значення набуває архітектурне проєктування чат-ботів, оскільки саме структура програмної системи визначає її