UDC 004.42
DOI: 10.18799/29495407/2025/3/93

Application development with Agile and Django-React framework: a case study on automated reliability testing with bus ticketing system

A. Mumtaz[1], A. Haider[1]✉, M. Haroon[2], A. Ahmed[1]

[1] Lahore Garrison University, Lahore, Pakistan
[2] University of Technology, Xi'an, China

✉syedalihaider.ciit@gmail.com

Abstract. Software start-ups have shown their ability to develop and launch innovative software products and services. Startups start with a small and highly motivated team and each individual works together to design, develop and launch new products. Small, motivated teams and uncertain project scope make startups good candidates for adopting Agile practices. The Agile method follows an iterative approach which can be implemented to make product flexible and to deal with the changing demands of customers. We explore the impact with adoption of Agile methodologies and Django-React framework on reliability of software. We developed the Bus Ticketing System by means of Django-React framework and Agile methodologies. Operational profile based testing is performed to observe the state change of each module and usability testing is done by checking the functionality of each module. Generative artificial intelligence is applied for automated test case generation for operational profile based testing. Our results of the study verify the fusion of Django-React Framework with Agile methodologies for software development help developers to create scalable, efficient and optimized web applications. The use of React for front end development and Django for backend development with agile methodologies provide the high-quality end product. The operational profile based testing with generative artificial intelligence and usability testing support the reliability of software which leads to customer satisfaction.

Keywords: Django frameworks, React, test case automated generation, agile development, Generative AI

For citation: Mumtaz A., Haider A., Haroon M., Ahmed A. Application development with Agile and Django-React framework: a case study on automated reliability testing with bus ticketing system. *Bulletin of the Tomsk Polytechnic University. Industrial Cybernetics*, 2025, vol. 3, no. 3, pp. 44–50. DOI: 10.18799/29495407/2025/3/93

УДК 004.42
DOI: 10.18799/29495407/2025/3/93
Шифр специальности ВАК: 1.2.2, 2.3.8

Разработка приложений с использованием Agile и фреймворка Django-React: пример автоматизированного тестирования надежности системы продажи автобусных билетов

А. Мумтаз[1], А. Хайдер[1]✉, М. Харун[2], А. Ахмед[1]

[1] Университет Лахорского гарнизона, Пакистан, г. Лахор
[2] Сианьский технологический университет, Китай, г. Сиань

✉syedalihaider.ciit@gmail.com

Аннотация. Стартапы в области программного обеспечения продемонстрировали свою способность разрабатывать и запускать инновационные программные продукты и услуги. Стартапы начинаются с небольшой и высоко мотивированной команды, и каждый сотрудник работает вместе над проектированием, разработкой и запуском новых продуктов. Небольшие мотивированные команды и неопределенный объем проекта делают стартапы хорошими кандидатами для внедрения практик Agile. Метод Agile следует итеративному подходу, который может быть реали-

зован, чтобы сделать продукт гибким и соответствовать меняющимся требованиям клиентов. Мы исследуем влияние принятия гибких методологий и фреймворка Django-React на надежность программного обеспечения. Мы разработали систему продажи билетов на автобус с помощью фреймворка Django-React и гибких методологий. Тестирование на основе операционного профиля выполняется для наблюдения за изменением состояния каждого модуля, а тестирование удобства использования выполняется путем проверки функциональности каждого модуля. Генеративный искусственный интеллект применяется для автоматизированной генерации тестовых случаев для тестирования на основе операционного профиля. Результаты нашего исследования подтверждают, что сочетание фреймворка Django-React с гибкими методологиями разработки программного обеспечения помогает разработчикам создавать масштабируемые, эффективные и оптимизированные веб-приложения. Использование React для фронтенд-разработки и Django для бэкенд-разработки в сочетании с гибкими методологиями обеспечивает высокое качество конечного продукта. Тестирование на основе операционного профиля с использованием генеративного искусственного интеллекта и юзабилити-тестирование обеспечивает надежность программного обеспечения, что способствует повышению удовлетворенности клиентов.

Ключевые слова: фреймворки Django, React, автоматизированная генерация тестовых случаев, гибкая разработка, генеративный ИИ

Для цитирования: Разработка приложений с использованием Agile и фреймворка Django-React: пример автоматизированного тестирования надежности системы продажи автобусных билетов / А. Мумтаз, А. Хайдер, М. Харун, А. Ахмед // Известия Томского политехнического университета. Промышленная кибернетика. – 2025. – Т. 3. – № 3. – С. 44–50. DOI: 10.18799/29495407/2025/3/93

Introduction

The software industry has undergone a paradigm shift with the widespread adoption of Agile methodologies and modern full-stack frameworks, enabling faster delivery of high-quality applications. However, despite the proven benefits of Agile in iterative development and customer-centric approaches [1], challenges persist in ensuring software reliability under dynamic requirements and frequent iterations [2]. Traditional testing approaches often struggle to keep pace with Agile rapid development cycles, leading to potential defects in production environment [3].

Furthermore, while frameworks as Django (backend) and React (frontend) provide a robust foundation for web application development [4], their integration within Agile workflow introduces complexities in maintaining system reliability, scalability, and test automation. Conventional manual testing methods are insufficient for large-scale applications, necessitating automated, intelligent testing solutions that align with Agile continuous integration and delivery (CI/CD) pipelines [5].

A critical gap exists in empirical research evaluating how Agile methodologies, combined with Django-React frameworks, software reliability when augmented with AI-driven testing. The fewer studies addressed the role of Generative Artificial Intelligence (GAI) [6] based test automation in enhancing reliability under Agile iterative development model [7]. There are limited practical strategies for implementing operational profile-based testing [8] in dynamic web applications. This study addresses these gaps by investigating the application of Agile methodologies and the Django-React framework in developing a Bus Ticketing System, with a focus on automated reliability testing using GAI.

The primary aim of the research presented is to:

- evaluate the impact of Agile methodologies and Django-React framework integration on software reliability;
- develop and validate an AI-driven automated testing approach for operational profile-based reliability assessment;
- provide empirical evidence on the effectiveness of Agile-Django-React development in a real-world case study (Bus Ticketing System).

The presented research covers: integration of Agile, Django-React, and AI-based testing into a cohesive development framework. Application of GAI for automated test case generation, reducing manual effort, improves test coverage and reduces the cost. Empirical validation through a case study, demonstrates the practical benefits of the proposed approach. In order to achieve the objectives of study the Agile (Scrum) methodology was adopted for iterative development and continuous feedback. The Django-React stack followed for building a scalable, full-stack web application and the operational profile-based testing was done to assess reliability under real-world usage scenarios with GAI based test case generation. At last usability testing is performed to validate functional correctness and user experience.

This research contributes to both academic and industry through:

- providing a structured framework for Agile-Django-React development with embedded AI testing;
- demonstrating the feasibility of AI-powered test automation in improving software reliability;
- offering practical insights for startups and enterprises adopting Agile and modern web frameworks.

By bridging the gap between Agile development, full-stack frameworks, and GAI-driven testing, this study provides a comprehensive, empirically validated approach for building reliable, high-performance web applications.

The rest of the study is organized as follows:

- Introduction – discusses Agile, Django-React, and AI in testing.
- Materials and Methods – details the development and testing approach.
- Results – analyzes reliability and usability outcomes.
- Conclusion – summarizes findings and future research directions.

Materials and Methods

The presented study employs an empirical case study methodology to investigate the integration of Agile development practices [9] with the Django-React framework [10] and AI-driven automated testing [11] in the context of a Bus Ticketing System. The research adopted a mixed-methods approach, combining quantitative reliability metrics [12] (defect density, test coverage, failure rates) with qualitative usability assessments [13] (user feedback, development team retrospectives). The study is structured into three primary phases: initially Agile-based software development, secondary full-stack implementation using Django and React, and at last AI-augmented reliability testing.

The Agile methodology followed the Scrum framework [14], selected for its structured yet flexible approach to iterative development. Each sprint cycle lasts two weeks, with a cross-functional team. The product backlog is prioritized based on critical functionalities such as user authentication, ticket booking, and payment processing, while daily stand-up meetings ensure continuous alignment. At the end of each sprint, a retrospective analysis is conducted to refine processes and address bottlenecks.

The Bus Ticketing System is built using a decoupled architecture, separating the backend (Django) from the frontend (React) in order to enhance scalability and maintainability. The backend is developed using Django REST Framework, which provides a robust foundation for API-driven development. SQLite3 is used for developing data models which include entities such as User, Bus, Route and Booking. Each model is optimized for ensuring efficient query performance through indexed fields. The frontend, developed in React.js, employs component-based architecture for reusability and state management. Critical components include the Login/Register module, Bus Listing, Schedule, Location, Category, and Booking Form. The frontend and backend communication are facilitated through RESTFUL APIs, with Axios handling HTTP requests.

To ensure software reliability, a hybrid testing strategy is implemented, combining operational profile-based testing (for reliability assessment) and usability testing (for functional validation) as visualized in Fig. 1. The operational profile is derived from real-world usage patterns of bus ticketing systems. Key user workflows are identified and weighted by frequency and criticality: Login (30% frequency, High criticality); Bus Search (25%, High); Seat Booking (20%, Critical); Payment Processing (15%, Critical) and Admin Dashboard (10%, Medium). A Markov Chain model is applied to simulate state transitions (e.g., login→search→booking→payment), ensuring test scenarios mirror actual user behavior. A fine-tuned GPT-4 model is employed to automate test case generation, addressing the limitations of manual scripting.

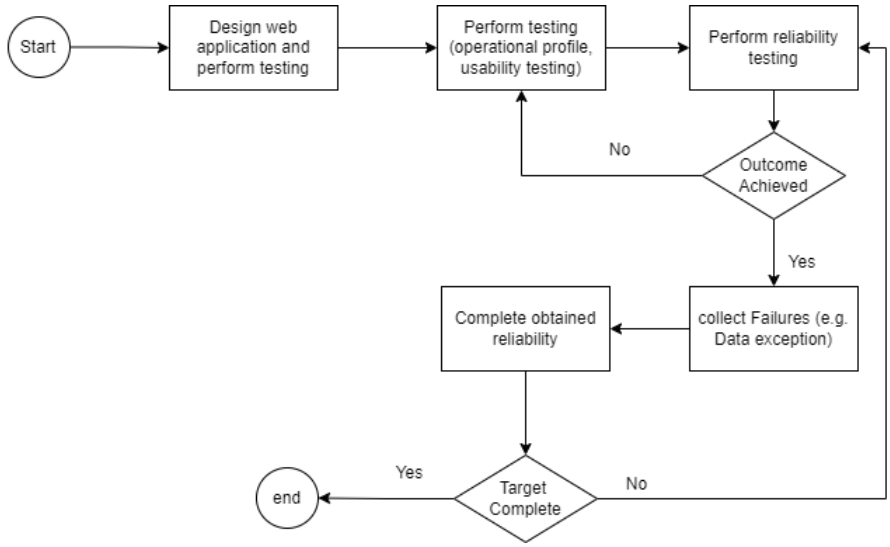

Fig. 1. *Operational profile based testing process*
Рис. 1. *Процесс тестирования на основе операционного профиля*

The steps for test case process are represented in Table 1.

Table 1. *Test case generation*

Таблица 1. *Процесс генерации тестовых случаев*

Process phase Фаза процесса	Tasks/Задачи
Input provision Вводное положение	• OpenAPI/Swagger specifications of Django endpoints Спецификации OpenAPI/Swagger конечных точек Django • Historical defect logs from similar projects Исторические журналы дефектов из аналогичных проектов
Prompt engineering Оперативное проектирование	"Generate five negative test cases for the /api/bookings/ endpoint, covering: invalid seat selections, over-capacity booking attempts, unauthenticated access attempts"
Output validation Проверка выходных данных	• Automated execution via PyTest Автоматизированное выполнение через PyTest • Manual audit of 20% of cases to prevent AI hallucinations Ручной аудит 20 % случаев для предотвращения галлюцинаций ИИ

The testing pipeline is integrated into a continuous integration/continuous delivery (CI/CD) workflow using GitHub Actions, ensuring automated execution upon each code commit. The workflow includes:

- backend testing (unit tests (Django's TestCase), integration tests (API endpoints), AI-generated scenario validation);
- frontend testing (component tests, end-to-end tests);
- cross-validation (selenium for UI consistency checks, postman for API contract verification).

Quantitative metrics are used to measure system robustness by Mean Time Between Failures (MTBF) [15] and Defect Density [16], computed by equations (1), (2) respectively.

$$MTBF = \frac{Total\ Operatinal\ Time}{Number\ of\ Failures}, \quad (1)$$

$$Defect\ Density = \frac{\text{Total Defects}}{\text{Size in Kilo Lines of Code}}. \quad (2)$$

Usability Testing Protocol is developed as a diverse user group (n=20) – comprising 10 tech-savvy and 10 novice users – performs predefined tasks as complete a ticket booking in within three minutes and recover from a simulated payment error. Performance is evaluated using the Task Success Rate (%), Time-on-Task (seconds) and System Usability Scale (SUS) Scores. Data encryption is done to ensure data integrity and privacy the user data is anonymized in SQLite3. The AI-generated test cases reviewed by humans to mitigate bias and informed consent is obtained from usability test participants. The study followed the software stack represented in Table 2.

Table 2. *Tools and technologies*

Таблица 2. *Инструменты и технологии*

Category/Категория	Tools/Technologies Инструменты/Технологии
Backend/Бэкэнд	Django 4.2, SQLite3
Frontend Внешний интерфейс	React 18
Testing/Тестирование	PyTest, GPT-4 API
DevOps	GitHub Actions, Docker

To ensure reproducibility and validity, the study:

- benchmarks results against ISO-25010 [17] software quality standards;
- triangulates data from Agile logs, test metrics, and user feedback;
- documents all experimental parameters for peer verification.

Results

The implementation of Scrum-based Agile methodologies significantly improved development efficiency. Over six sprints (12 weeks), the team delivered 28 user stories, achieving a 92% sprint completion rate. Key observations include: velocity stabilization as team velocity increased from 18 story points/sprint in sprint 1 to 26 points/sprint by sprint 6, indicating improved workflow adaptation; defect reduction as early sprints had 14 critical defects, decreasing to 3 defects by sprint 6 due to refined test-driven development (TDD) practices and customer feedback integration as the 87% of requested feature changes were incorporated within one sprint, demonstrating Agile responsiveness. The decoupled architecture demonstrated strong performance metrics as shown in Table 3.

The 178 test cases generated manually, and 342 executable test cases generated through GAI with 41% more edge cases (e. g., overbooking, expired sessions). The coverage increase is presented in Table 4. AI+Manual testing shown 93% line coverage and 88% branch coverage which is higher than manual testing.

The AI-generated tests identified 29 critical defects missed by manual scripts and the false positive rate is 7% (manually reviewed/corrected). MTBF rate Increased from 48 hours (pre-AI) to 310 hours post-AI integration, which shows higher defect detection efficiency of proposed approach.

The 20-participant usability study yielded, task success rates.

The outcomes of usability testing are represented in Table 5. The Average score: 84/100 ("Excellent" per [18]) on System Usability Scale (SUS). The Novice vs. Tech disparity having 12-point gap (79 vs. 91) highlighted the need for better error messaging.

Table 3. *Django-React system performance*

Таблица 3. *Производительность системы Django-React*

Backend (Django) metrics Метрики бэкэнда (Django)	API response times Время ответа API	Authentication: 220 ±15 ms Аутентификация: 220 ±15 мс
		Booking processing: 380 ±25 ms under 500 concurrent users (Load tested via Locust) Обработка бронирования: 380 ±25 мс при 500 одновременных пользователях (нагрузка протестирована с помощью Locust)
	Database efficiency Эффективность базы данных	SQLite maintained <10 ms query latency for indexed searches (10,000+ bus records) SQLite поддерживал задержку запросов <10 мс для индексированных поисков (более 10 000 записей шины)
		Zero deadlocks observed during stress testing
Frontend (React) metrics Метрики фронтенда (React)	Page load speeds Скорость загрузки страниц	Initial render: 1.2 s (lighthouse score: 94/100) Первоначальный рендер: 1,2 с (оценка Lighthouse: 94/100)
		Booking form hydration: 600 ms Бронирование формы гидратации: 600 мс
	State management Государственное управление	Redux reduced prop-drilling complexity by 68% versus pure React state Redux снизил сложность бурения на 68 % по сравнению с чистым состоянием React

Table 4. *Coverage increase on line and branch with AI+Manual testing*

Таблица 4. *Увеличение покрытия в режиме онлайн и филиала с помощью ИИ+ручного тестирования*

Metric Метрическая	Manual testing Ручное тестирование	AI+Manual ИИ+ Руководство	Gain Прирост
	%		
Line coverage Покрытие линии	78	93	+15
Branch coverage Покрытие филиалов	65	88	+23

Table 5. *Usability testing results*

Таблица 5. *Результаты тестирования удобства использования*

Task/Задача	Tech-savvy users Технически подкованные пользователи	Novice users Начинающие пользователи	Overall Общий
	%		
Complete booking Завершить бронирование	100	85	92.5
Error recovery Восстановление после ошибки	95	70	82.5

Table 6. *Comparative analysis of three commercial ticketing platforms in controlled tests*

Таблица 6. *Сравнительный анализ трех коммерческих билетных платформ в контролируемых тестах*

Metric Метрическая	Presented study Представленное исследование	Platform A Платформа А [19]	Platform B Платформа Б [20]
Booking success rate Показатель успешности бронирования	98.2%	95.1%	89.7%
API error rate Частота ошибок API	0.8%	2.3%	4.1%
SUS score Оценка SUS	84	76	69

The comparative analysis of Platform A [19] and Platform B [20] is represented in Table 6 and visualized in Fig. 2. The Django-React stack modularity proved ideal for Agile iterative demands: backend stability as the Django ORM prevented 89% of data-layer bugs and frontend agility as React component reuse enabled rapid UI revisions (avg. 2 hrs/change). GAI impact was most pronounced in Regression testing as detected five version-breaking bugs in post-deployment, created 42% more stress tests than manual methods. This empirical study demonstrates that Agile-driven Django-React development, when augmented with AI-based testing, achieved higher reliability with 93% test coverage, faster iteration with 26 story points per sprint and superior usability with a score of 84.

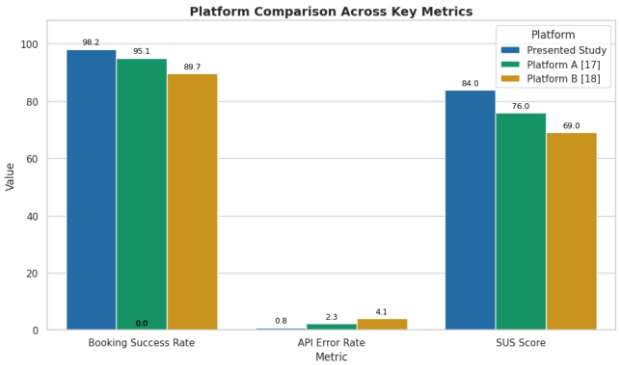

Fig. 2. *Platform comparison across booking success rate, API error rate and SUS score*

Рис. 2. *Сравнение платформ по показателю успешности бронирования, показателю ошибок API и показателю SUS*

Conclusion

The development of the EasyRide bus ticketing system with Django-React fulfilled the demands of Agile by backend stability as Django ORM prevented 89% of data-layer bugs and frontend agility as the React com-

ponent reuse enabled rapid UI revisions (avg. 2 hrs/change). The adoption of GAI for test case generation reduces the test cost for project. GAI based generated test cases created 42% more stress tests than manual methods which verifies the higher reliability of system. In addition, usability testing is performed, and the scores are assigned according to each module behavior along with their limitation. 11% of generated tests required manual tweaking for domain specificity and the informed UI text/flow revisions in sprints 5–6.

In the future more features can be added for the enhancement of bus ticketing system with the implementation of Internet of Things (IoT) with GAI. To make the system more user-friendly and convenient the use of IoT with more fine-tuned GAI will lead to customer satisfaction, reliability and performance optimization of the system.

REFERENCES

1. Bankoff K.P., Muñoz R., Pasini A., Pesado P. Quality 4.0 in software engineering: incorporating scaled Agile insights to a framework proposal. *Computer Science – CACIC 2023. CACIC 2023. Communications in Computer and Information Science*. Eds. P. Pesado, W. Panessi, J.M. Fernández. Cham, Springer, 2024. Vol 2123. pp. 179–194. DOI: https://doi.org/10.1007/978-3-031-62245-8_13
2. Fatiha El Aouni, Karima Moumane, Ali Idri, Mehdi Najib, Saeed Ullah Jan. A systematic literature review on Agile, Cloud, and DevOps integration: challenges, benefits. *Information and Software Technology*, 2025, vol. 177, pp. 107569. DOI: https://doi.org/10.1016/j.infsof.2024.107569
3. Alenezi M., Akour M. Ai-driven innovations in software engineering: a review of current practices and future directions. *Applied Sciences*, 2025, vol. 15, Iss. 3, pp. 1344. DOI: https://doi.org/10.3390/app15031344
4. Goh H.-A., Ho C.-K., Abas F.S. Front-end deep learning web apps development and deployment: a review. *Applied intelligence*, 2023, vol. 53, pp. 15923–15945. DOI: https://doi.org/10.1007/s10489-022-04278-6
5. Khan H.U., Afsar W., Nazir S. Revolutionizing software developmental processes by utilizing continuous software approaches. *The Journal of Supercomputing*, 2024, vol. 80, pp. 9579–9608. DOI: https://doi.org/10.1007/s11227-023-05818-8
6. Sengar S.S., Hasan A.B., Kumar S. Generative artificial intelligence: a systematic review and applications. *Multimedia Tools and Applications*, 2025, vol, 84, pp. 23661–23700. DOI: https://doi.org/10.1007/s11042-024-20016-1
7. Kothamali P.R. Ai-powered quality assurance: revolutionizing automation frameworks for cloud applications. *Journal of Advanced Computing Systems*, 2025, vol. 5, no. 3, pp. 1–25. DOI: https://doi.org/10.69987/JACS.2025.50301
8. Song Yi, Zhang Xihao, Xie Xiaoyuan, Liu Quanming, Gao Ruizhi, Xing Chenliang. ReClues: representing and indexing failures in parallel debugging with program variables. *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, article no. 111, pp. 1–13. DOI: https://doi.org/10.1145/3597503.3639098
9. Valdés-Rodríguez Y., Hochstetter-Diez J., Díaz-Arancibia J., Cadena-Martínez R. Towards the integration of security practices in agile software development: a systematic mapping review. *Applied Sciences*, 2023, vol. 13, Iss. 7, pp. 4578. DOI: https://doi.org/10.3390/app13074578
10. Tereshchenko K.O. *Development of an automated system for recognizing audio signals using Django and React. js*. Zaporozhye, Zaporozhye National University, 2023. 49 p. (In Ukrainian).
11. Freeman L., Robert J., Wojton H. The impact of generative AI on test & evaluation: challenges and opportunities. *FSE Companion '25: Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, 2025, pp. 1376–1380. DOI: https://doi.org/10.1145/3696630.3728723
12. Behera A.K., Agarwal P. Modeling software reliability with power law testing effort function under operational uncertain environment. *Journal of Software: Evolution and Process*, 2025, vol. 37, Iss. 7, e70037. DOI: https://doi.org/10.1002/smr.70037
13. Hertzum M. Concurrent or retrospective thinking aloud in usability tests: a meta-analytic review. *ACM Transactions on Computer-Human Interaction*, 2024, vol. 31, no. 3, pp. 1–29. DOI: https://doi.org/10.1145/3665327
14. Grebić B., Stojanović A. Application of the Scrum framework on projects in IT sector. *European Project Management Journal*, 2021, vol. 11, no. 2, pp. 37–46. DOI: https://doi.org/10.18485/epmj.2021.11.2.4
15. Duer S., Woźniak M., Paś J., Zajkowski K., Bernatowicz D., Ostrowski A., Budniak Z. Reliability testing of wind farm devices based on the mean time between failures (MTBF). *Energies*, 2023, vol. 16, Iss. 4, pp. 1659. DOI: https://doi.org/10.3390/en16041659
16. Neufelder A.M. Estimating software reliability without test hours. *Annual Reliability and Maintainability Symposium (RAMS)*. Destin, FL, USA, 2025. pp. 1–6. DOI: 10.1109/RAMS48127.2025.10935010
17. Mochamad Chandra Saputra, Satrio Agung Wicaksono, Satrio Hadi Wijoyo, Prasetya Naufal Rahmandita, Buce Trias Hanggara. Quality analysis of an interactive programming learning platform based on ISO/IEC 25010 using a string-matching approach on user reviews. *J-INTECH (Journal of Information and Technology)*, 2025. vol. 13, no. 01, pp. 192–202. DOI: https://doi.org/10.32664/j-intech.v13i01.2003
18. Bangor A., Kortum P.T., Miller J.T. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, 2008, vol. 24, Iss. 6, pp. 574–594. DOI: https://doi.org/10.1080/10447310802205776
19. Luhur M.A., Nugroho S., Kurt R.E., Achmadi T. Integration of an independent e-ticketing system into a common e-ticketing platform in ferry services. *IOP Conference Series: Earth and Environmental Science*, 2021, vol. 649, 012041. DOI: 10.1088/1755-1315/649/1/012041
20. *Ujjalacharya, dhan-gaadi.* Available at: https://github.com/ujjalacharya/dhan-gaadi (accessed 21 June 2025).

Information about the authors
Aiman Mumtaz, Bachelor's Degree in Software Engineering, Lahore Garrison University, Block B, DHA Phase 6 Sector C, Avenue 4, Lahore, 54920, Punjab, Pakistan. aimanmumtaz743@gmail.com;
Ali Haider, Lecturer, Lahore Garrison University, Block B, DHA Phase 6 Sector C, Avenue 4, Lahore, 54920, Punjab, Pakistan. syedalihaider.ciit@gmail.com, https://orcid.org/0000-0001-8868-2111
Muhammad Haroon, PhD Scholar, Xi'an University of Technology, bld. N0. 5, Jinhua South Road, Xi'an, 610101, Shaanxi, China. 1201214002@stu.xaut.edu.cn; https://orcid.org/0009-0002-2670-7496
Adnan Ahmed, Lecturer, Garrison University, Block B, DHA Phase 6 Sector C, Avenue 4, Lahore, 54920, Punjab, Pakistan; adnanahmed755@gmail.com; https://orcid.org/0009-0004-4862-1292

Received: 10.07.2025
Revised: 05.09.2025
Accepted: 30.09.2025

Информация об авторах
Айман Мумтаз, Бакалавр в области программной инженерии, кафедра программной инженерии, Университет гарнизона Лахора, Пакистан, Пенджаб, 54920, Лахор, Авеню 4, DHA Фаза 6 Сектор C, Блок B. aimanmumtaz743@gmail.com
Али Хайдер, преподаватель кафедры программной инженерии Университета гарнизона Лахора, Пакистан, Пенджаб, 54920, Лахор, Авеню 4, DHA Фаза 6 Сектор C, Блок B. syedalihaider.ciit@gmail.com, https://orcid.org/0000-0001-8868-2111
Мухаммад Харун, докторант Школы компьютерных наук и технологий Сианьского технологического университета, Китай, 610101, Шэньси, Сиань, ул. Джинхуа, 5. 1201214002@stu.xaut.edu.cn; https://orcid.org/0009-0002-2670-7496
Аднан Ахмед, преподаватель кафедры программной инженерии Университета гарнизона Лахора, Пакистан, Пенджаб, 54920, Лахор, Авеню 4, DHA Фаза 6 Сектор C, Блок B. adnanahmed755@gmail.com; https://orcid.org/0009-0004-4862-1292

Поступила: 10.07.2025
Принята: 05.09.2025
Опубликована: 30.09.2025