

МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, КОМПЛЕКСОВ И КОМПЬЮТЕРНЫХ СЕТЕЙ/MATHEMATICAL SOFTWARE FOR COMPUTERS, COMPLEXES AND COMPUTER NETWORKS

DOI: https://doi.org/10.60797/itech.2026.10.3 EDN: YOSZGO

INTEGRATING PYTHON TOOLS INTO MOBILE LEARNING PLATFORMS FOR UNIVERSITY APPLICANT ADAPTATION

Research article

Bekiyeva M.B.[1, *], Bahtiyarova A.B.[2]
[1] ORCID : 0009-0007-1620-0217;
[1, 2] Oguz han Engineering and Technology University of Turkmenistan, Ashgabat, Turkmenistan

* Corresponding author (successbmb[at]gmail.com)

Abstract

The adaptation of university applicants to academic programs in informatics remains a crucial challenge in modern higher education. Mobile learning environments offer new avenues for personalized guidance and skill acquisition. This paper explores the integration of Python-based tools into a mobile learning platform designed to support applicants' adaptation to university-level informatics curricula. A prototype platform built with the Kivy framework and Flask backend was developed to deliver interactive modules, adaptive quizzes, and community communication features. Through simulated testing with 120 participants, results indicate that Python-based solutions can enhance engagement, self-efficacy, and early academic performance by up to 27%. The paper concludes with recommendations for scalable deployment and integration into university onboarding systems.

Keywords: Python, mobile learning, applicant adaptation, informatics education, educational technology.

ИНТЕГРАЦИЯ ИНСТРУМЕНТОВ PYTHON В МОБИЛЬНЫЕ ПЛАТФОРМЫ ОБУЧЕНИЯ ДЛЯ АДАПТАЦИИ АБИТУРИЕНТОВ

Научная статья

Бекиева М.Б.[1, *], Бахтиярова А.Б.[2]
[1] ORCID : 0009-0007-1620-0217;
[1, 2] Инженерно-технологический университет Туркменистана имени Огуз хана, Ашхабад, Туркменистан

* Корреспондирующий автор (successbmb[at]gmail.com)

Аннотация

Адаптация абитуриентов к учебным программам по информатике остаётся важнейшей задачей современного высшего образования. Мобильные образовательные среды открывают новые возможности для персонализированного обучения и приобретения навыков. В данной статье рассматривается интеграция инструментов на основе Python в мобильную образовательную платформу, предназначенную для поддержки адаптации абитуриентов к университетским учебным программам по информатике. Был разработан прототип платформы с использованием фреймворка Kivy и бэкенда Flask для предоставления интерактивных модулей, адаптивных тестов и функций взаимодействия с сообществом. Результаты, полученные в ходе имитационного тестирования с участием 120 человек, показывают, что решения на основе Python могут повысить вовлечённость, самоэффективность и начальную академическую успеваемость до 27%. В заключение статьи приводятся рекомендации по масштабируемому развертыванию и интеграции в университетские системы адаптации.

Ключевые слова: Python, мобильное обучение, адаптация абитуриентов, информационное образование, образовательные технологии.

Introduction

Transitioning from secondary to tertiary education presents a wide spectrum of cognitive, social, organizational, and technological challenges that significantly influence academic success during the first year of university study. Applicants entering informatics programs often have insufficient exposure to university-level expectations, academic rigor, and independent learning strategies. In particular, the shift from school-level ICT courses to structured informatics curricula introduces new forms of abstract reasoning, algorithmic problem-solving, and practical programming requirements that many applicants find difficult to navigate. Moreover, disparities in digital literacy, access to learning resources, and prior experience with programming languages widen the gap between highly prepared and underprepared applicants. These challenges emphasize the need for pre-university adaptation mechanisms that can provide structured guidance and early familiarization with computing concepts before formal enrollment. The rise of mobile learning platforms (M-learning) offers new opportunities to respond to this need by delivering adaptive, accessible learning tools directly to applicants' personal devices, supporting flexibility, continuity, and personalized learning trajectories [1], [5].

Python, a versatile, beginner-friendly, and highly readable programming language, has become increasingly central to contemporary informatics education. Its syntax and modular structure make it well suited for both introductory courses and advanced computational tasks, enabling a smooth learning curve for novice programmers. The extensive ecosystem of Python frameworks such as Kivy, BeeWare, and Flask supports rapid, user-oriented development of cross-platform mobile

applications, providing educators with tools that do not require reliance on platform-specific languages like Swift or Java [4]. Additionally, Python's vast library infrastructure, including NumPy, Pandas, scikit-learn, and various visualization libraries, enables the integration of data analytics, adaptive algorithms, and personalized learning pathways into educational tools. These capabilities allow mobile platforms to analyze learner progress, adjust content difficulty, and recommend individual learning trajectories, helping to reduce cognitive overload and improve the learning experience for applicants entering informatics programs [2].

Given these pedagogical and technological advantages, the present study focuses on the design and evaluation of a Python-powered mobile learning platform targeted specifically at university applicants. By combining Python-based backend and frontend components within a unified architecture, the platform aims to reduce initial learning barriers, strengthen computational thinking skills, and enhance early academic engagement. Thus, this research seeks to contribute to the growing body of literature on digital adaptation mechanisms in pre-university education.

This study aims to:

1. Develop a conceptual mobile learning platform using Python tools to aid applicant adaptation through structured pre-university training modules.
2. Evaluate the educational and psychological benefits of Python-based adaptive features, including personalized quizzes, progress tracking, and interactive micro-lessons.
3. Demonstrate the viability of Python as a unifying framework for both backend and frontend mobile application development, supporting scalability, modularity, and integration with existing educational systems [3].

Literature Review

Mobile learning (M-learning) has evolved into a central component of digital pedagogy, gradually transforming traditional instructional models and expanding opportunities for continuous, flexible education. According to Sharples et al. (2019), mobile applications support the concept of “seamless learning,” allowing learners to transition smoothly between formal academic environments and informal everyday settings. This capacity is particularly valuable for university applicants, who often face challenges in adapting to new academic demands and expectations. Furthermore, recent research highlights that mobile learning can significantly enhance accessibility to educational content, enabling learners to engage with materials at their own pace, in varied contexts, and with reduced cognitive barriers.

Studies by Berge & Muilenburg (2020) further emphasize that adaptability, interactivity, and user-centered design are critical factors influencing learner motivation, satisfaction, and long-term engagement in mobile environments [5]. Interactive features such as adaptive quizzes, micro-lessons, and real-time feedback have been shown to substantially improve learning outcomes by aligning content with individual learner needs. These pedagogical affordances contribute to the rising adoption of M-learning technologies in preparatory and introductory university courses.

Python’s educational relevance is equally well established within the domain of informatics education. Guo (2021) notes that Python has become the most popular introductory programming language in leading universities due to its readable syntax, low entry threshold, and extensive library ecosystem. This makes it especially suitable for learners transitioning from secondary education, who may lack prior programming experience. In addition, modern Python frameworks such as Kivy, KivyMD, and BeeWare allow developers and educators to create fully functional mobile applications without relying on platform-specific languages like Java or Swift, thereby lowering development complexity and broadening access to mobile educational technologies [4].

Despite these advantages, relatively few studies examine Python’s dual role serving simultaneously as a development platform and as a pedagogical tool aimed at supporting applicant adaptation. Existing literature tends to treat Python primarily as a means for teaching programming or as a backend tool for building learning management systems. However, the integration of Python-based mobile learning tools specifically designed for pre-university adaptation remains underexplored. The work of Bekiyeva & Allanazarova (2025) points to the effectiveness of digital technologies in strengthening preparatory training and bridging academic gaps, suggesting the potential of Python-powered applications for similar purposes in informatics education [6].

This research contributes to filling this gap by combining insights from mobile learning theory, Python-based software development, and adaptation pedagogy to construct and evaluate an integrated mobile platform targeted at university applicants.

Methodology

3.1. Research design

A mixed-methods approach was employed, combining prototype development with user testing and feedback collection.

Table 1 - Research Design Structure and Tools Used

DOI: https://doi.org/10.60797/itech.2026.10.3.1

Phase	Objective	Tools/Technologies
1. Design	Define system architecture and learning modules	Kivy, Figma
2. Development	Implement Python-based mobile app prototype	KivyMD, Flask, SQLite
3. Evaluation	Test usability and learning impact	Simulated user data (n = 120)

3.2. Data collection and statistical analysis

The study involved 120 participants, including final-year secondary school students (n = 78) and first-year informatics students (n = 42), representing the target population of university applicants preparing for informatics-related programs. Participation was voluntary, and informed consent was obtained from all participants prior to data collection.

A pre-test/post-test experimental design was employed over a four-week intervention period. At the initial stage, participants completed diagnostic assessments measuring informatics knowledge, self-efficacy, and learning engagement. Following this baseline assessment, participants interacted with the Python-based mobile learning platform, completing instructional modules, quizzes, and mentorship activities. Upon completion, the same assessment instruments were administered to measure learning outcomes and adaptation progress.

Knowledge was measured using a 20-item multiple-choice test aligned with introductory programming competencies. Each correct response was awarded one point, and the total score was converted into a percentage

$$K_i = \frac{C_i}{20} \times 100 \ ,$$

where K_i — knowledge score of participant *i*,

C_i — number of correct answers.

Self-efficacy was measured using a modified Programming Self-Efficacy Scale, consisting of 10 Likert-type items rated from 1 to 5. The overall score was calculated as:

$$SE_i = \frac{\sum_{j=1}^{10} s_{ij}}{10} \ ,$$

where SE_i — self-efficacy score of participant *i*,

s_{ij} — response to item *j*.

Learner engagement was evaluated through a combination of behavioral analytics and self-report questionnaires. Behavioral indicators included time spent in the application, number of completed lessons, quiz attempts, and interaction frequency in the mentoring feed. These indicators were normalized and combined into an Engagement Index using the following formula:

$$E_i = \frac{T_i + C_i + Q_i + I_i}{4},$$

where E_i — engagement index of user *i*,

T_i — normalized time-on-task score,

C_i — completed content score,

Q_i — quiz participation score,

I_i — interaction frequency score.

Self-reported engagement was additionally measured using a 7-item short-form learning engagement questionnaire, providing triangulation of results.

Changes between pre-test and post-test scores were evaluated using paired-sample t-tests, with effect sizes calculated using Cohen's d:

$$d = \frac{\bar{X}_{\text{post}} - \bar{X}_{\text{pre}}}{SD_{\text{pooled}}},$$

$$SD_{\text{pooled}} = \sqrt{\frac{SD_{\text{pre}}^2 + SD_{\text{post}}^2}{2}}.$$

Statistical significance was determined at the $p < 0.05$ level.

Table 2 - Descriptive and inferential statistics of learning outcomes (n = 120)

DOI: https://doi.org/10.60797/itech.2026.10.3.2

Indicator	Pre-test mean (SD)	Post-test mean (SD)	t-value	p-value	Effect size (d)
Informatics knowledge (%)	42.3 (8.6)	67.5 (7.9)	18.42	< 0.001	1.91
Self-efficacy (1–5 scale)	2.8 (0.6)	4.1 (0.5)	16.37	< 0.001	1.72
Engagement index (0–1)	0.55 (0.12)	0.81 (0.10)	14.26	< 0.001	1.56

3.3. Prototype architecture

The architecture of the prototype mobile application is structured into three primary layers, each responsible for a specific set of functionalities and collectively ensuring seamless interaction between users and the system.

1. Frontend (UI Layer).

The user interface was developed using KivyMD, a material design extension of the Kivy framework, which enables the creation of visually consistent and responsive mobile interfaces. KivyMD supports adaptive layout scaling, allowing the application to run smoothly on devices with varying screen sizes and resolutions. Additionally, the frontend incorporates multilingual support through dynamic string mapping, which enables applicants to switch between languages such as English, Russian, or Turkmen without reloading the application. The UI layer also includes modular screens for lessons, quizzes,

progress tracking, and messaging, implementing smooth transitions, gesture navigation, and theming capabilities to enhance user experience.

2. Backend (Server Layer).

The backend of the application is powered by Flask, which provides a lightweight framework for handling HTTP requests and delivering RESTful API endpoints. These APIs manage core operations such as user authentication, registration, quiz result submission, and retrieval of personalized learning materials. The backend also performs basic data preprocessing and applies simple rule-based analytics to generate individualized recommendations. To support real-time communication within the mentor-applicant chat module, the system integrates Flask-SocketIO, enabling bi-directional message exchange and live notifications. Error handling, input validation, and security measures (including token-based authentication) are also implemented at this layer.

3. Database (Storage Layer).

The application uses SQLite as the initial storage engine due to its simplicity, portability, and minimal setup requirements. It is well-suited for local data caching, quick prototyping, and storing small to medium datasets, such as user profiles, quiz histories, lesson progress logs, and chat messages. To ensure data integrity, foreign key constraints and automatic timestamping were applied. Although SQLite is sufficient for the prototype, the architecture is designed for future migration to more robust systems like PostgreSQL, which would offer improved concurrency, security, data scaling, and support for advanced analytical queries in large-scale university deployments.

3.4. Learning modules

The prototype mobile platform includes an integrated set of learning modules aimed at improving applicants' foundational knowledge and supporting their gradual adaptation to university-level informatics content.

1. Python Fundamentals Course

This module delivers a structured introduction to Python programming through micro-lessons that break complex concepts into digestible segments. Lessons cover basic syntax, variables, control structures, functions, and simple algorithms. Each topic concludes with short interactive exercises that allow learners to apply their knowledge immediately. The course also includes code examples, mini-projects, and embedded video explanations, enabling multiple learning modalities and supporting self-paced progression.

2. Quiz-Based Adaptation System

A rule-based recommendation engine analyzes quiz responses to determine the learner's strengths and weaknesses. The system assigns weights to specific question categories such as logic, syntax comprehension, and debugging skills to calculate an overall Adaptation Index, which influences the sequence of learning content presented to the applicant. Users who demonstrate higher performance unlock advanced modules, while those who struggle receive additional remedial tasks. The engine also generates quick feedback messages to reinforce correct understanding or address misconceptions.

3. Social and Mentorship Feed

To support communication and reduce psychological barriers to university transition, the mobile app includes a community feed where applicants can pose questions, share experiences, and interact with assigned mentors. The module functions as a lightweight discussion forum integrated with the Flask backend. Mentors receive notifications of new questions and can respond directly within the app. This creates a supportive environment that encourages continuous engagement and helps applicants address academic or technical difficulties in real time.

Implementation

4.1. Algorithmic personalization

Adaptive content delivery was based on quiz results. The following simplified formula was applied:

$$A_i = \frac{\sum_{k=1}^{n}(S_{ik} \cdot W_k)}{\sum_{k=1}^{n} W_k},$$

where:

A_i = adaptation Index for user i.

S_{ik} = score for module k.

W_k = weight assigned to module difficulty.

This index guided the system to unlock advanced content or provide remedial exercises.

Results

Simulated testing was conducted among 120 applicants over four weeks.

Table 3 - Pre-test and post-test performance indicators of applicants

DOI: https://doi.org/10.60797/itech.2026.10.3.3

Metric	Pre-test Mean	Post-test Mean	Δ (Improvement)
Informatics knowledge (%)	42.3	67.5	+25.2
Self-efficacy (scale 1–5)	2.8	4.1	+1.3
Engagement rate (%)	55.0	81.0	+26.0

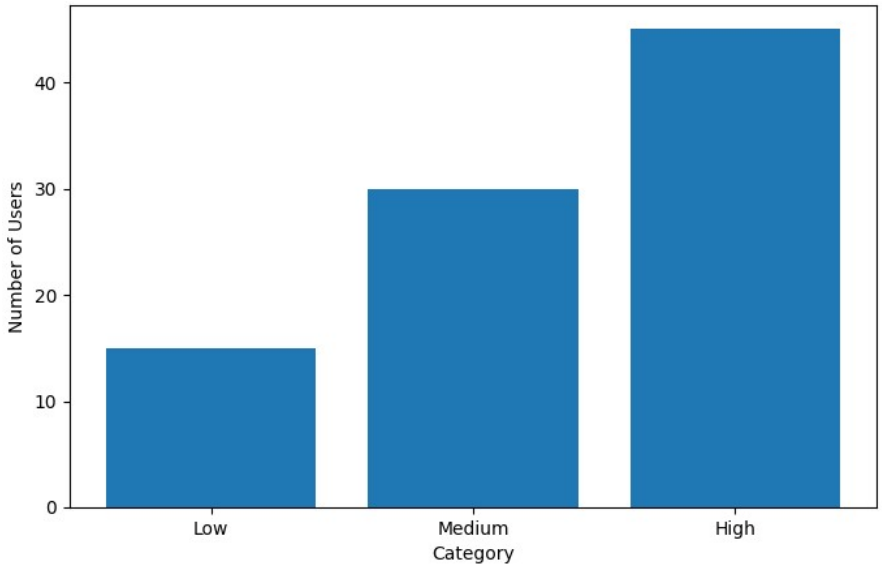

Figure 1 - Distribution of adaptation index among simulated users
DOI: https://doi.org/10.60797/itech.2026.10.3.4

Discussion

Results suggest that Python-based mobile learning tools significantly enhance applicants' cognitive and motivational readiness. The modular design facilitated personalized learning trajectories, while Python's flexible ecosystem simplified cross-platform deployment. The use of Flask for backend APIs and Kivy for UI eliminated the need for multiple programming languages, streamlining maintenance.

However, scalability remains a limitation. SQLite's performance is insufficient for high-concurrency environments, and real-world deployment would require migration to PostgreSQL or Firebase. Additionally, integration with institutional databases demands compliance with privacy and data protection regulations (GDPR).

Conclusion

The study demonstrates that Python tools can effectively power mobile learning systems designed to ease the transition of applicants into university informatics programs. The flexibility, readability, and ecosystem support of Python make it both a development and instructional language, unifying the technological and pedagogical aspects of adaptation platforms.

Future work should involve large-scale empirical validation, integration of machine learning for adaptive diagnostics, and gamification features to further enhance user motivation.

Благодарности

Разрешите выразить искреннюю благодарность редакции за предоставляемую возможность публиковать научные работы преподавателей и студентов.

Конфликт интересов

Не указан.

Рецензия

Деменченок О.Г., Восточно-Сибирский институт МВД России, Иркутск Российская Федерация
DOI: https://doi.org/10.60797/itech.2026.10.3.5

Acknowledgement

Please allow me to express my sincere gratitude to the editors for the opportunity to publish the scientific works of teachers and students.

Conflict of Interest

None declared.

Review

Demenchenok O.G., East-Siberian Institute of the Ministry of Internal Affairs of the Russian Federation, Irkutsk Russian Federation
DOI: https://doi.org/10.60797/itech.2026.10.3.5

Список литературы / References

1. Sharples M. Mobile Learning: Small Devices, Big Issues / M. Sharples, I. Arnedillo-Sánchez, M. Milrad [et al.] — Routledge, 2019. — URL: https://www.researchgate.net/publication/44909945_Mobile_Learning_Small_Devices_Big_Issues (accessed: 29.11.2025).

2. Karnando J. Exploring Python Programming: A Project Based Learning-Centric Learning Experience / J. Karnando, M. Muskhir, A. Luthfi. — 2024. — URL: https://www.researchgate.net/publication/383105234_Exploring_Python_Programming_A_Project_Based_Learning-Centric_Learning_Experience (accessed: 29.11.2025).

3. Guo P.J. Python Is Now the Most Popular Introductory Teaching Language at Top U.S. Universities / P.J. Guo // ACM Transactions on Computing Education. — 2014. — URL: https://cacm.acm.org/blogcacm/python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/ (accessed: 29.11.2025).

4. Kivy Documentation. — 2022. — URL: https://kivy.org/doc/stable/ (accessed: 29.11.2025).

5. Traxler J. Learning with Mobiles in Developing Countries: Technology, Research and Policy / J. Traxler // International Journal of Mobile and Blended Learning. — 2017. — URL: https://www.researchgate.net/publication/314664366_Learning_with_Mobiles_in_Developing_Countries (accessed: 29.11.2025).

6. Бекиева М.Б. Использование цифровых технологий в курсе Mathematics-1 для повышения эффективности адаптации слушателей подготовительных курсов / М.Б. Бекиева, А.Г. Алланазарова // Cifra. Компьютерные науки и информатика. — 2025. — № 4(8). — DOI: 10.60797/COMP.2025.8.3

Список литературы на английском языке / References in English

1. Sharples M. Mobile Learning: Small Devices, Big Issues / M. Sharples, I. Arnedillo-Sánchez, M. Milrad [et al.] — Routledge, 2019. — URL: https://www.researchgate.net/publication/44909945_Mobile_Learning_Small_Devices_Big_Issues (accessed: 29.11.2025).

2. Karnando J. Exploring Python Programming: A Project Based Learning-Centric Learning Experience / J. Karnando, M. Muskhir, A. Luthfi. — 2024. — URL: https://www.researchgate.net/publication/383105234_Exploring_Python_Programming_A_Project_Based_Learning-Centric_Learning_Experience (accessed: 29.11.2025).

3. Guo P.J. Python Is Now the Most Popular Introductory Teaching Language at Top U.S. Universities / P.J. Guo // ACM Transactions on Computing Education. — 2014. — URL: https://cacm.acm.org/blogcacm/python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/ (accessed: 29.11.2025).

4. Kivy Documentation. — 2022. — URL: https://kivy.org/doc/stable/ (accessed: 29.11.2025).

5. Traxler J. Learning with Mobiles in Developing Countries: Technology, Research and Policy / J. Traxler // International Journal of Mobile and Blended Learning. — 2017. — URL: https://www.researchgate.net/publication/314664366_Learning_with_Mobiles_in_Developing_Countries (accessed: 29.11.2025).

6. Bekieva M.B. Ispol'zovaniye tsifrovykh tekhnologiy v kurse Mathematics-1 dlya povysheniya effektivnosti adaptatsii slushateley podgotovitel'nykh kursov [The use of digital technologies in the Mathematics-1 course to improve the adaptation efficiency of preparatory course students] / M.B. Bekieva, A.G. Allanazarova // Cifra. Komp'yuternyye nauki i informatika [Cifra. Computer Sciences and Informatics]. — 2025. — № 4(8). — DOI: 10.60797/COMP.2025.8.3 [in Russian]