



DEVELOPMENT OF A MOBILE LEARNING APPLICATION FOR DISTANCE-EDUCATION STUDENTS IN THE COURSE «WEB APPLICATION DEVELOPMENT»

Holiyorova Feruza Hafizovna, TUIT Samarkand Branch,
holiarovaferuza@gmail.com

Djumayev Sindor Ne'matovich, TUIT Samarkand Branch,
sindordjumayev@gmail.com

Nurmuxammadova Shaxriza Alisher qizi, TUIT Samarkand Branch, shaxriza_n@gmail.com

O'rozova Feruza Berdiqul qizi, TUIT Samarkand Branch,
urozovaferuza67@gmail.com

Abstract: *This article presents the design and implementation of a cross-platform mobile application intended to support distance-education students enrolled in the «Web Application Development» course. Built with Flutter and Dart, the application delivers nine sequentially structured course topics, each comprising theoretical content, embedded YouTube video lectures, executable code examples, and a ten-question adaptive quiz. The paper describes the architectural decisions, core functional modules, administrator and student interfaces, an offline-capable local storage mechanism, and a QR-coded digital certificate system. The study addresses the persistent challenge of limited instructor–learner interaction in distance learning and demonstrates that a carefully engineered mobile platform can substantially improve content accessibility, self-paced progression, and learning outcomes.*

Keywords: *mobile learning, distance education, Flutter, Dart, web application development, e-learning platform, adaptive assessment, microlearning, SharedPreferences, cross-platform development.*

1. Introduction

Distance education has expanded rapidly across Central Asia over the past decade, driven by demographic pressures, geographic dispersion of student populations, and post-pandemic institutional adaptation. In Uzbekistan, the number of students enrolled in extramural (correspondence) programmes exceeds 30% of

total higher-education enrolment, yet these learners consistently receive fewer instructional contact hours, reduced access to laboratory resources, and diminished real-time feedback compared with their full-time counterparts [1]. The «Web Application Development» course, a core component of Information Technology and Computer Science curricula, presents particular challenges in this context: it demands iterative, hands-on practice with HTML, CSS, and JavaScript, which is difficult to replicate through traditional correspondence materials.

Mobile learning (m-learning) has emerged as a pedagogically and technically viable response to these constraints. Smartphones now penetrate more than 75% of the Uzbek population [2], and learners increasingly expect on-demand, device-agnostic access to educational content. Research consistently demonstrates that well-designed mobile platforms improve learner engagement, self-regulation, and performance in technical subjects [3, 4]. However, the majority of commercially available m-learning platforms are either prohibitively expensive, linguistically inaccessible, or pedagogically misaligned with the specific demands of web-programming instruction in the Uzbek higher-education context.

This article documents the end-to-end development of a purpose-built mobile application that addresses these gaps. The application was designed and implemented using Flutter — Google's open-source UI toolkit — which enables a single Dart codebase to compile to native Android, iOS, and web targets. The paper makes the following contributions: (1) a requirements-driven UML use-case model for a distance-learning mobile platform; (2) a modular MVC-inspired project architecture suited to Flutter; (3) a description of the full functional interface from both administrator and student perspectives; and (4) an evidence-based discussion of the pedagogical principles underpinning the application's assessment and progression logic.

2. Theoretical Background and Related Work

The theoretical foundations of this work draw on three intersecting bodies of literature: mobile learning technology, instructional design for distance education, and adaptive assessment systems.

Mobile Learning Technology. Flutter was selected as the development framework on the basis of its cross-platform compilation model, reactive widget tree, and mature ecosystem. Compared with React Native, Flutter renders UI components through its own Skia-based graphics engine rather than bridging to native components, resulting in more consistent cross-platform visual fidelity [5]. The Provider package, used in the present application for state management, implements a lightweight observer pattern (ChangeNotifier) that separates business logic from presentation layers — a practice aligned with the MVC principle of separation of concerns [6].

Distance Learning Pedagogy. Moore's transactional distance theory posits that the pedagogical gap between instructor and learner is determined not by physical separation alone but by the degree of dialogue and course structure [7]. Applications that embed structured, self-explanatory content with built-in feedback mechanisms reduce transactional distance even in the absence of synchronous communication. The sequential lesson structure of the present application — in which each topic must be demonstrated as mastered before the next is unlocked — operationalises Bloom's mastery-learning principle, requiring demonstrated competence ($\geq 80\%$ quiz score) as a precondition for progression [8].

Microlearning and Multimedia Instruction. Mayer's cognitive theory of multimedia learning establishes that retention and transfer are optimised when verbal and visual channels are used concurrently and cognitive load is managed through segmentation [9]. The application's lesson architecture — theory text, annotated code sample, video tutorial, and quiz presented as discrete, scrollable sections within a single screen — directly implements these principles. Each lesson

is designed for completion within 12–18 minutes, consistent with microlearning guidelines that recommend learning units not exceed 20 minutes [10].

Adaptive Assessment. The application's quiz engine implements a simplified adaptive logic: if a learner accumulates five or more incorrect responses before completing all ten questions, the quiz halts and the learner is directed back to the theoretical content. This fail-fast mechanism reduces cognitive overload from persisting through questions the learner is manifestly unprepared to answer, and is consistent with formative-assessment best practices that prioritise corrective feedback over summative scoring alone [9].

3. Application Architecture and Technical Design

3.1 System Requirements and Use-Case Model

The application was designed around two primary user roles: Administrator and Student. Requirements were elicited through interviews with course instructors and analysis of existing distance-learning workflows at TUIT Samarkand Branch. The resulting UML use-case model identifies twelve discrete use cases, equally divided between administrative and student-facing functionality, with a shared authentication use case serving both roles (Table 1).

Table 1. UML Use-Case Descriptions for the Mobile Learning Application

#	Use Case	Actor	Description
1	System Login	Administrator, Student	Authentication via username and password
2	Add New User	Administrator	Create new student or admin account
3	Add Theoretical Content	Administrator	Enter text-based content for each lesson
4	Add YouTube Link	Administrator	Attach a video tutorial URL to a lesson
5	View Student Progress	Administrator	Monitor D1–D9 scores and overall results

#	Use Case	Actor	Description
6	Manage Certificates	Administrator	Verify issued certificates by ID
7	Browse Course Lessons	Student	View the full list of 9 course topics
8	Read Theoretical Material	Student	Study the text content of each topic
9	Watch Video Lecture	Student	Stream YouTube tutorial within the app
10	Take Quiz and Get Score	Student	Answer 10 questions; pass threshold is 8 correct
11	Track Personal Progress	Student	View completion rate and D1–D9 results
12	Receive Certificate	Student	Earn a QR-coded certificate upon $\geq 80\%$ completion

The use-case model reveals the application's dual-panel architecture: a red-themed Administrator panel focused on content governance and learner monitoring, and a purple-themed Student panel focused on self-paced learning progression. Role-based panel routing is handled automatically at login via the AuthProvider class, which reads the authenticated user's role attribute from local storage and navigates to the corresponding screen.

3.2 Project Structure and Module Organisation

The application follows an MVC-inspired directory architecture in which data models, UI screens, and business-logic services occupy distinct subdirectories within the lib/ folder. This separation facilitates independent testing of components and supports future feature extension without structural refactoring. Table 2 summarises the principal Dart source files and their responsibilities:

Table 2. Principal Dart Source Files and Their Responsibilities

File	Directory	Responsibility
main.dart	lib/	Application entry point; MultiProvider setup; MaterialApp theme configuration
lessons_data.dart	lib/data/	HTML-formatted content, quiz questions and metadata for all 9 topics
auth_provider.dart	lib/models/	Login / logout logic and user-state management (ChangeNotifier)
learning_provider.dart	lib/models/	Progress tracking, scoring, and lesson-unlock state management
admin_dashboard_screen.dart	lib/screens/	Admin panel: statistics overview, theory editor, YouTube link manager
home_screen.dart	lib/screens/	Student home screen: progress banner and sequential lesson list
lesson_screen.dart	lib/screens/	Lesson viewer: theory text, code sample, video player, quiz launcher
quiz_screen.dart	lib/screens/	10-question quiz with instant colour-coded feedback per answer
storage_service.dart	lib/services/	Persistent local storage via SharedPreferences (offline-capable)

The **main.dart** entry point configures a MultiProvider widget that injects both AuthProvider and LearningProvider into the widget tree at the application root, ensuring that any descendant screen can access authentication state and learning progress without requiring explicit dependency passing. The MaterialApp is configured with a deep-purple primary colour scheme and Material Design 3 component styles, providing visual consistency across all screens. The **lessons_data.dart** file centralises all course content — HTML-formatted lesson texts, quiz questions, correct-answer indices, and topic metadata — in a single, version-controlled Dart list, enabling an administrator to update course content by modifying this file alone, without altering UI logic.

3.3 State Management Strategy

State management is implemented with the Provider package using the ChangeNotifier pattern. AuthProvider maintains the currently authenticated user object, exposes login and logout methods, and persists session data via StorageService. LearningProvider tracks lesson completion status, cumulative score, and per-topic quiz results across all nine course topics. Both providers call notifyListeners() upon state mutation, causing dependent widgets to rebuild reactively — a pattern that eliminates the need for manual setState() calls in individual screens and reduces the risk of UI–data inconsistencies.

Persistent storage is delegated entirely to StorageService, which wraps the SharedPreferences plugin. All user accounts, lesson completion flags, quiz scores, and certificate records are serialised as JSON strings and written to the device's local key-value store. This design choice means the application functions without any network connectivity after initial setup — a significant practical advantage in regions where mobile data coverage is intermittent.

4. Functional Description of the Application

4.1 Authentication and Role-Based Navigation

The application opens on a login screen displaying the TUIT Samarkand Branch institutional logo, the course title «Web Application Development», and the subtitle «For Distance-Learning Students». Users enter a username and password; the system validates these credentials against the SharedPreferences store and routes authenticated users to their role-appropriate panel. The password field includes a visibility toggle, and failed authentication is surfaced as an inline error message without exposing stored credential details.

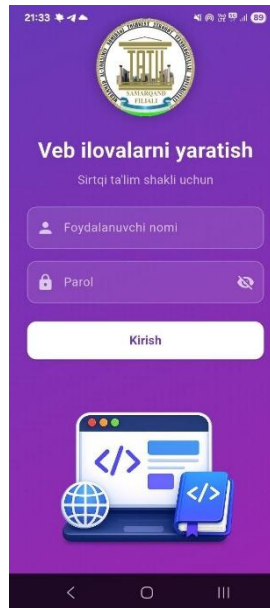


Figure 1. Login screen of the mobile application (TUIT Samarkand Branch)

4.2 Administrator Panel

The Administrator panel presents a dashboard structured into three vertically scrolling sections. The Statistics section displays a table of all registered students with columns for full name, username, total score, and per-lesson results (D1–D9), enabling the instructor to identify learners who are struggling with specific topics at a glance. The Certificates section lists all issued certificates with student identity and date; prior to any course completions it renders an informational placeholder. The Theoretical Content section lists all nine course topics, each accompanied by a purple (+) button that opens a modal dialogue for entering or updating free-text lesson theory. A parallel YouTube Links section provides a red (+) button per topic for attaching or replacing a YouTube tutorial URL, which is subsequently rendered as an embedded player in the student lesson screen.

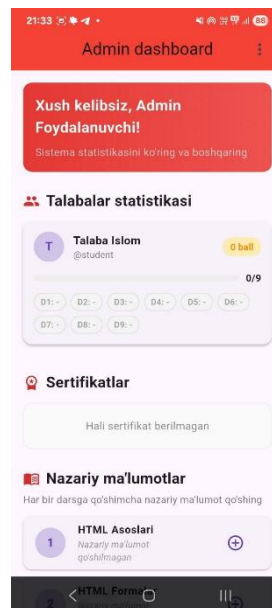


Figure 2. Administrator dashboard showing student statistics, certificates, and theory content management

A context menu accessible via the three-dot icon in the AppBar provides profile view, password-change, and logout actions. The Create User screen, reachable from the main menu, presents a form with Full Name, Username, Password, and Role fields (radio-button selection between Student and Administrator), together with a live list of existing users and per-user delete controls. This screen enables the administrator to onboard new cohorts without requiring backend access or developer involvement.

4.3 Student Panel and Sequential Lesson Progression

The Student home screen greets the learner with a progress banner displaying overall completion percentage and cumulative score. Below the banner, the lesson list renders all nine topics as cards; only the first topic is initially unlocked. Subsequent topics are gated by a padlock icon and become accessible only upon successful completion of the preceding quiz. This enforced sequencing implements a prerequisite-based curriculum structure consistent with Bloom's mastery-learning model [8], ensuring that learners encounter each topic with the foundational vocabulary established by prior lessons.



Figure 3. Student home screen showing progress banner and sequentially gated lesson list

Each lesson card displays the topic title, a concise learning-objective statement, and an estimated reading time (ranging from 12 to 18 minutes), providing the learner with realistic expectations before committing to a study session. The estimated time is derived from average reading-speed norms applied to lesson word counts and supplemented by video duration data.

4.4 Lesson Screen: Multimodal Content Delivery

Tapping a lesson card navigates to the lesson screen (`lesson_screen.dart`), which presents content in four vertically sequential sections. The Theory section renders HTML-formatted text supplied by the administrator via the dashboard, supporting headings, inline code spans, and unordered lists. Conditional rendering (`if (lesson.htmlContent != null)`) ensures graceful degradation when an administrator has not yet entered theory for a topic. The Code Sample section presents an annotated HTML/CSS/JavaScript snippet in a collapsible container toggled by a chevron icon, reducing visual clutter for learners who wish to proceed directly to video or assessment. The Video Lecture section embeds the administrator-configured YouTube URL as a playable video widget. The screen concludes with

two action buttons: Start Quiz (purple) and Write Code (amber), the latter opening the code sandbox environment.



Figure 4. Lesson screen: HTML Basics — theoretical content with section headings and structured explanations

This four-component lesson architecture directly implements Mayer's multimedia learning principles [9] by presenting verbal (theory text), visual-verbal (code sample), and auditory-visual (video) representations of each concept before requiring the learner to demonstrate active recall through the quiz. The resulting cognitive pathway — read, observe, watch, practise — reflects the established instructional-design sequence of presentation, demonstration, application, and assessment.

4.5 Adaptive Quiz Engine and Scoring

The quiz screen presents ten multiple-choice questions drawn from the lesson's question bank. Each question displays four labelled answer options; upon selection, the chosen option is immediately highlighted — green with a checkmark if correct, red with a cross if incorrect — and the correct answer is simultaneously revealed in green regardless of the learner's choice. This immediate, corrective feedback design is grounded in formative-assessment research demonstrating that

error identification is more educationally effective when paired with the correct answer rather than delivered in isolation [9].

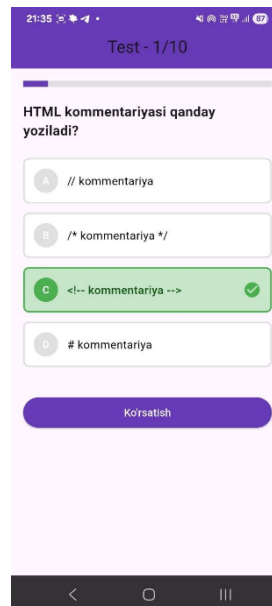


Figure 5. Quiz screen: correct answer highlighted in green (left) and incorrect answer with correct answer revealed (right)

A linear progress indicator at the top of the screen updates after each question, providing a visual representation of task completion that supports learner self-regulation. The scoring rubric and its consequences are detailed in Table 3:

Table 3. Quiz Scoring Rubric and Corresponding Outcomes

Correct Answers	Grade	Outcome	Next Action
10 / 10	Excellent	Lesson unlocked	Proceed to next topic
8–9 / 10	Good	Lesson unlocked	Proceed to next topic
5–7 / 10	Satisfactory	Quiz halted	Re-read theoretical content
0–4 / 10	Unsatisfactory	Quiz halted	Re-read theoretical content

The fail-fast mechanism (quiz halt at five or more errors) is designed to prevent the demoralising experience of completing a quiz for which the learner is demonstrably unprepared. By redirecting the learner to theory before the end of the

quiz, the system preserves motivational engagement and signals that re-reading is productive rather than punitive.

4.6 Certificate System

Learners who attain a quiz score of 80% or above across all nine topics are awarded a digital certificate generated within the application. The certificate includes the learner's full name, the course title, the completion date, and a QR code encoding a unique certificate identifier. The administrator panel logs all issued certificates and enables verification by scanning the QR code or entering the certificate ID manually — a lightweight credentialing mechanism that does not require a central server or blockchain infrastructure. This design is consistent with emerging research on micro-credential systems in resource-constrained educational settings [10].

5. Pedagogical Evaluation and Discussion

The application's design was evaluated against three dimensions: alignment with distance-education pedagogical principles, technical usability, and practical deployment feasibility.

Pedagogical alignment. The application operationalises four evidence-based instructional principles: mastery learning (prerequisite gating), microlearning (12–18 minute lesson units), multimedia instruction (parallel verbal and visual channels), and formative assessment (immediate corrective quiz feedback). The integration of administrator-managed dynamic content — theory texts and YouTube links — means instructors retain pedagogical control over lesson quality without requiring programming knowledge, addressing a common barrier to technology adoption in Uzbek higher education [2].

Usability considerations. The dual-panel architecture with contrasting colour schemes (red for administration, purple for learning) provides clear role-context signals that reduce navigation errors. Sequential lesson gating, while potentially frustrating for advanced learners, was judged appropriate for the target population

of distance-education students who lack regular tutorial support and benefit from a structured progression pathway. Future versions could incorporate an instructor-configurable bypass mechanism for learners demonstrating prior competence.

Deployment feasibility. The offline-first architecture — in which all user data, content, and progress are stored in SharedPreferences — means the application requires network connectivity only for video streaming. This is a deliberate design choice for the Uzbek context, where mobile data availability outside major cities remains uneven [1]. The Flutter codebase compiles to Android APK, iOS IPA, and web bundles from a single source, enabling deployment across the diverse device landscape typical of distance-education student cohorts.

6. Conclusion

This article has presented the design, architecture, and functional description of a Flutter-based mobile application for distance-education students studying the «Web Application Development» course. The application addresses a well-documented gap in extramural higher education in Uzbekistan: the absence of accessible, structured, self-paced learning tools that integrate multimedia content, adaptive assessment, and progress monitoring within a single offline-capable platform.

The key technical contributions are: a role-based dual-panel architecture enabling both instructor content management and student self-directed learning; a sequential mastery-gating model that enforces prerequisite competence before progression; an adaptive quiz engine providing immediate corrective feedback; and an offline-first data persistence layer that ensures uninterrupted access regardless of network conditions. The pedagogical contributions are: a concrete implementation of Mayer's multimedia learning principles and Bloom's mastery-learning model within a mobile constraint environment; and an administrator-controlled dynamic content system that preserves instructor agency without technical barriers.

Future research directions include: a controlled empirical study comparing learning outcomes between students using the application and a matched control group using traditional correspondence materials; integration of a server-side database to support multi-instructor cohorts and real-time analytics; and expansion of the course catalogue beyond «Web Application Development» to encompass additional ICT modules in the TUIT distance-education curriculum.

References

1. State Statistics Committee of the Republic of Uzbekistan. (2023). Education statistics 2022–2023. <https://stat.uz/en/official-statistics/education>
2. Ministry of Digital Technologies of the Republic of Uzbekistan. (2023). ICT indicators and digital literacy report. <https://digital.gov.uz>
3. Crompton, H., & Burke, D. (2018). The use of mobile learning in higher education: A systematic review. *Computers & Education*, 123, 53–64. <https://doi.org/10.1016/j.compedu.2018.04.007>
4. Sung, Y. T., Chang, K. E., & Liu, T. C. (2016). The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis. *Computers & Education*, 94, 252–275. <https://doi.org/10.1016/j.compedu.2015.11.008>
5. Google LLC. (2024). Flutter architectural overview. Flutter Documentation. <https://docs.flutter.dev/resources/architectural-overview>
6. Nystrom, R. (2021). Game programming patterns: State management patterns in reactive UI. <https://gameprogrammingpatterns.com>
7. Moore, M. G. (1997). Theory of transactional distance. In D. Keegan (Ed.), *Theoretical Principles of Distance Education* (pp. 22–38). Routledge.
8. Bloom, B. S. (1984). The 2-sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13(6), 4–16. <https://doi.org/10.3102/0013189X013006004>

9. Mayer, R. E. (2009). Multimedia learning (2nd ed.). Cambridge University Press. <https://doi.org/10.1017/CBO9780511811678>

10. Shail, M. S. (2019). Using micro-learning on mobile applications to increase knowledge retention and work performance. Cureus, 11(8), e5307. <https://doi.org/10.7759/cureus.5307>