INTERNATIONAL CONFERENCE PROCEEDINGS
UNIVERSITAS ISLAM BALITAR BLITAR
Empowering Education and Society through Digital Transformation
Volume 1 (2025)

Designing a Digital Al-Qur'an Application for Featuers Color Tajwid Using Personal Extreme Programming Method

Nugroho Nurwanda Zakaria[1]*, Zunita Wulansari S.Kom., M.T [2], Abdi Pandhu Kusuma S. Kom M.T [3]

Informatics Engineering Study Program, Faculty of Engineering and Informatics, Universitas Islam Balitar

Keywords:
Tajwid Berwarna, Al-Qur'an Digital, React Native, Personal Extreme Programming (PXP), Automated Testing, User Acceptance Testing (UAT)

***Correspondence Address:**
nugrohonurwandaz@gmail.com

Abstract: This study aims to design a digital Qur'an application with interactive color-coded *tajwid* features using the Personal Extreme Programming (PXP) method. The application is designed to enable users to read the Qur'an while visually learning *tajwid* rules through a color-based system. Development was conducted iteratively over five cycles within 34 working days using the React Native framework integrated with the Quran.com API. Evaluation employed two testing methods. Automation testing using Jest showed average unit test coverage of 77.34% (statements), 69.33% (branches), 81.97% (functions), and 77.70% (lines). The *translationCleaner.js* file achieved 100% coverage, while *CompactMusicPlayer.js* reached only 56.31%. Integration testing produced excellent results, with most files exceeding 90% coverage, though *TabNavigator.js* achieved only 55.31%. User Acceptance Testing (UAT) involving 24 respondents and 21 questions revealed that 20 items were rated "Excellent" (>80%). The highest score, 91.67%, was for *tajwid* material assistance in reading ability, while the lowest, 75%, concerned *tajwid* material quality. The study concludes that the PXP method is effective for developing small-scale educational applications through an iterative approach that allows quick responses to user needs, producing an application that meets expectations.

INTRODUCTION

The advancement of information and communication technology has brought transformative impacts on various aspects of life, including in Islamic religious education. Learning the Qur'an, as a fundamental element of this education, requires interactive and easily accessible methods to engage users from diverse backgrounds and levels of comprehension. A significant challenge is the delivery of **tajwid** knowledge, which is an important guide for reciting the Qur'an correctly and in accordance with established rules (Mahkfudz et al., 2021).

The science of **tajwid** is defined as the discipline that studies how to pronounce each letter in the Qur'an by giving it its proper rights and characteristics,

as well as by observing specific reading rules such as elongation (**mad**), pronunciation thickness and thinness (**tafkhim** and **tarqiq**), and the point of articulation (**makhraj**). Reading the Qur'an with correct tajwid is a fundamental aspect of Islamic religious education, aimed at preserving the authenticity of its meaning and the beauty of its recitation. However, challenges in learning tajwid often arise, especially for users who have limited access to interactive and effective learning methods (Oktarina, 2020).

Based on a survey conducted by the researchers on 48 respondents, the most dominant challenges were a lack of interactive guidance (31.3%), difficulty in understanding tajwid theory (27.1%), and time constraints for learning (39.6%). Although most respondents read the Qur'an every day (37.5%) or several times a week (37.5%), there remains an urgent need for a more modern and interactive learning approach. A significant finding showed that 75% of respondents considered a colored tajwid feature "very important" to help them understand Qur'an recitation, which indicates the great potential of a visual approach in tajwid learning.

The colored tajwid feature can provide clear and systematic visual guidance, especially for users with a low level of tajwid comprehension or those who need a practical reminder during recitation. Previous research has highlighted the importance of a visual approach in tajwid learning, where a coloring system can facilitate the real-time identification of reading rules without the need for in-depth text analysis (Lajnah Pentashihan Mushaf Al-Qur′an, 2011; Yani et al., 2021). The tajwid coloring system categorizes each reading rule with a specific color: red for the **mad** rule (long vowels), blue for the **ikhfa** rule, green for **ghunnah** (nasal sounds), yellow for **qalqalah** (vowel reflection), and gray for the **wasal** and **waqaf** rules.

A comparative analysis of popular digital Qur'an applications, namely MyQuran and Alquran Indonesia, revealed significant limitations in the implementation of interactive features. Although both applications provide a colored tajwid feature, they lack interactive features that would allow users to receive direct explanations of the meaning of the tajwid colors or a specific menu

detailing the coloring rules. This gap presents an opportunity to develop an application that not only displays colored tajwid but also includes interactive features to enhance learning effectiveness.

In the context of modern mobile application development, Personal Extreme Programming (PXP) emerges as a suitable methodology for small-scale software development with characteristics of high flexibility and rapid iteration. PXP is an adaptation of Extreme Programming optimized for individual development, offering advantages in iterative planning, continuous testing, continuous integration, and a simple design that focuses on current needs(Ulfi et al., 2020; Wibowo et al., 2019). Agile methodologies like PXP are highly relevant in the context of modern digital transformation because they allow developers to adapt quickly to changing user needs and expectations (Oladapo Adeboye Popoola et al., 2024)

React Native, a JavaScript framework for cross-platform mobile application development, offers an efficient solution for creating a digital Qur'an application with a colored tajwid feature. React Native's strength in accessing native components allows for the implementation of specific features such as marking tajwid with different colors according to the rules, as well as integration with various libraries and APIs to enrich application functionality (Kurniawan & Yulhendri, 2023; Setiawan, 2021). To ensure the quality and reliability of the application, automation testing will be implemented in each development iteration to guarantee system stability and early detection of potential errors. Automation testing, specifically unit testing and integration testing, has been proven effective in increasing testing efficiency and effectiveness by reducing manual involvement and accelerating the development cycle (Kumar & Mishra, 2016; Melia & Putra, 2023). Furthermore, User Acceptance Testing (UAT) will involve end-users to measure the application's satisfaction and usability in real-world conditions (Afrianto et al., 2021).

The **State of the Art** in digital Qur'an application development shows trends in the use of native Android technology with the Waterfall methodology (Kasoni et al., 2024; Yaqin, 2022), the **User-Centered Design** approach (Sari et al., 2021),

and the implementation of web technology with REST APIs (Sholeh et al., 2022). Previous research has also explored a system for the automatic recognition of tajwid rules using machine learning with a validation accuracy of 99 (M. Alagrami & M. Eljazzar, 2020), as well as the development of tajwid learning methods through the Tajwid Wheel approach (Syaifullah et al., 2022) .The novelty of this research lies in the integration of several innovative aspects that differentiate it from previous studies, including the use of React Native for cross-platform Qur'an application development; the implementation of a dynamically designed colored tajwid feature through the utilization of **fetch API** to retrieve verse data, tajwid rules, and audio in real time; the application of the PXP methodology which provides the advantage of rapid iteration and high flexibility; and the integration of **tilawah** audio that is synchronized with colored tajwid highlights to create a more interactive and immersive learning experience.

This research is expected to make a significant contribution to the development of Islamic religious education technology by providing an application solution that not only displays the Qur'an in a digital format but also facilitates tajwid learning through a systematic and interactive visual approach. This Android-based application with a colored tajwid feature offers users the flexibility to read and learn tajwid anytime and anywhere without relying on physical books or face-to-face guidance. The target users of the application include students who study tajwid as part of their educational curriculum or independently, as well as the general public with varying levels of tajwid understanding who want to improve their Qur'an recitation skills for daily worship. Based on a pre-survey conducted at Universitas Islam Balitar involving the general Muslim community, the majority of respondents showed high enthusiasm for the development of an innovative and easily accessible tajwid learning application.

Based on the background analysis and the identification of the research gap that has been presented, the research problems are: how to design and implement a digital Qur'an application with a colored tajwid feature using the Personal Extreme Programming (PXP) methodology and **React Native** technology, and how effective the application of automation testing and User Acceptance Testing (UAT) is in

ensuring the quality, stability, and user acceptance of the digital Qur'an application with a colored tajwid feature.

RESEARCH METHODS

This research was conducted at Universitas Islam Balitar (UNISBA) in Blitar, East Java, Indonesia. Data collection took place in the second week of January 2025, with application development continuing until June 2025. The study uses a **Research and Development (R&D)** approach, aiming to design and develop a digital Qur'an application with a colored **tajwid** feature using the **Personal Extreme Programming (PXP)** methodology. PXP was chosen for its suitability for individual application development projects, offering high flexibility in the design and implementation processes.

Data was collected using three primary methods: **comparative observation, questionnaires, and a literature review**. A comparative observation was performed on two popular digital Qur'an applications (**MyAlquran** and **Alquran Indonesia**) to analyze their existing colored tajwid features and identify gaps in user interaction. The analysis showed that while both applications provide colored tajwid features, they do not offer interactive explanations or dedicated menus for color references. A structured questionnaire was distributed to 48 respondents, including students from Universitas Islam Balitar and the general Muslim community, via Google Forms and WhatsApp. The survey assessed user demographics, reading habits, tajwid comprehension, and the perceived importance of the colored tajwid feature. The results indicated that 75% of respondents considered the feature "very important," and 31.3% identified a lack of interactive guidance as a major challenge in learning tajwid. The literature review involved 16 scientific articles from Google Scholar, IEEE, and Semantic Scholar to provide a theoretical foundation for the application's development and the implementation of the PXP methodology.

This research utilized **primary data** from observation and questionnaires, supplemented with **secondary data** from previous studies and the **Quran.com API** for accurate Qur'an content integration. Development tools included a Lenovo

ThinkPad L13 G2 with an Intel i3-1115G4 processor and 8GB of RAM, **Visual Studio Code**, the **React Native** framework, **JavaScript**, an Android Studio emulator, a Redmi 12 for physical device testing, Postman for API testing, Jest for unit and integration testing, and PlantUML with Mermaid.js for diagram creation. Application development followed the PXP methodology, which included requirements gathering, planning with **MoSCoW** priorities, design, implementation, automation testing, and **User Acceptance Testing (UAT)**. The needs analysis identified essential "Must Have" features such as Arabic Qur'an text display, a list of 114 surahs, verse numbering, Indonesian translation, and font size settings. "Should Have" features included **tilawah** audio, colored tajwid display, audio quality settings, and a dark/light mode. The system design used **UML diagrams** (use case, sequence, and activity) to model user interactions and system workflows. The implementation leveraged the **Quran.com API** to integrate data, with colored tajwid rendering based on the official guidelines from the Ministry of Religious Affairs of the Republic of Indonesia. The application architecture followed a modular principle using React Native components and JavaScript for cross-platform mobile development.

Testing procedures included automated **unit testing** and **integration testing** using Jest to ensure code quality and functionality. **User Acceptance Testing (UAT)** was performed using the **Black Box Testing** methodology, which validates functionality without examining the underlying code. UAT scenarios covered navigation, surah and verse selection, and the accuracy of tajwid coloring to ensure user requirements were met before the application's release. This research methodology ensures the systematic development of a digital Qur'an application that addresses the gaps in existing applications while providing enhanced interactive learning features for tajwid comprehension.

RESULTS AND DISCUSSION

System Implementation Results

The implementation of the Digital Qur'an application using the Personal Extreme Programming (PXP) methodology was successfully completed through

five structured development iterations. The development process began with comprehensive requirements gathering, followed by systematic planning, iterative development, testing, and an evaluation phase.

Requirements Analysis and Planning Results

The requirements analysis phase successfully identified critical system needs through various data sources. A **comparative analysis of existing applications** (MyQuran and Alquran Indonesia) revealed significant gaps in interactive **tajwid** explanations and dedicated color-code menus. A **pre-survey questionnaire** involving 48 respondents showed strong user demand, with **75%** considering the colored tajwid feature "very important" and **25%** rating it as "important" for understanding Qur'an recitation. The main challenges identified included a lack of interactive guidance (31.3%), difficulty in understanding tajwid theory (27.1%), and time constraints for learning (39.6%).

A **literature review** of academic papers from Google Scholar, IEEE, and Semantic Scholar provided the theoretical foundation for the development of the digital Qur'an application and the implementation of the **PXP** methodology. This comprehensive review established the research context and identified relevant prior work, contributing to the methodological rigor of the development process.

Application development followed a structured **five-iteration approach** using the **MoSCoW** prioritization technique. Table 1 presents a comprehensive planning structure with **user stories**, feature priorities, time estimates, and risk assessment.

The total estimated development time was **31 working days**, with an additional **10% tolerance** (**34 days total**) to accommodate technical challenges, debugging needs, design revisions, or minor delays. This structured approach ensured systematic development while maintaining the flexibility for iterative improvements.

Table 1. planning table with Moscow Priorityzation

Iteration	User Story	Feature Description	Priority	Estimation	Risk
Iteration 1 (Core Quranic Reading Functions – 5	US-01	Arabic text display	Must Have	1 day	Low

days)					
	US-02	Complete 114 surah list	Must Have	1 day	Low
	US-03	Verse numbering per line	Must Have	1 day	Low
	US-04	Indonesian translation	Must Have	1 day	Low
	US-05	Font size settings	Must Have	1 day	Low
Iteration 2 (Reading Experience Enhancement – 6 days)	US-06	Tajweed coloring	Should Have	4 days	High
	US-07	Tajweed law explanation	Should Have	1 day	Medium
	US-08	Surah search	Should Have	1 day	Low
Iteration 3 (Audio Features and Personalization – 9 days)	US-09	Audio recitation playback	Should Have	5 days	High
	US-10	Audio quality settings	Should Have	2 days	Medium
	US-11	Dark/light display mode	Should Have	2 days	Low
Iteration 4 (Learning Features and Customization – 5 days)	US-12	Basic tajweed learning materials	Could Have	3 days	Medium
	US-13	Arabic font customization	Could Have	2 days	Low
Iteration 5 (Application Testing – 6 days)	US-14	Automated testing (unit & integration)	Must Have	4 days	High
	US-15	User Acceptance Testing	Must Have	2 days	Medium

Iteration 1 established the application's foundation with core reading functionalities, including the display of Arabic text, a complete surah list, verse numbering, Indonesian translation, and font size adjustment. These "Must Have" features formed the essential basis for subsequent iterations.

Iteration 2 enhanced the reading experience through the implementation of colored **tajwid** (US-06), which represented the highest technical risk due to a complex color-coding algorithm based on the official guidelines of the Ministry of

Religious Affairs. The integration of **tajwid** rule explanations (US-07) and a surah search function (US-08) complemented the core coloring features.

Iteration 3 introduced audio capabilities with **tilawah** playback (US-09), which required integration with an external audio API, audio quality settings (US-10), and visual personalization through the implementation of dark/light mode (US-11). These "Should Have" features significantly improved user experience and accessibility.

Iteration 4 added educational components through basic **tajwid** learning materials (US-12) and Arabic font customization (US-13), which were classified as "Could Have" features that provide added value without compromising core functionality.

Iteration 5 focused on quality assurance through automated testing (US-14), including unit testing and integration testing, followed by **User Acceptance Testing** (US-15) to validate user satisfaction and application usability.

Implementation and Technical Architecture

The application interface implements a **dual-mode system** that supports light and dark themes to accommodate diverse user preferences and different usage contexts. Figures 1 and 2 display a comprehensive interface design, which consists of three main components: the surah list page, the **tajwid** learning material, and the application settings.

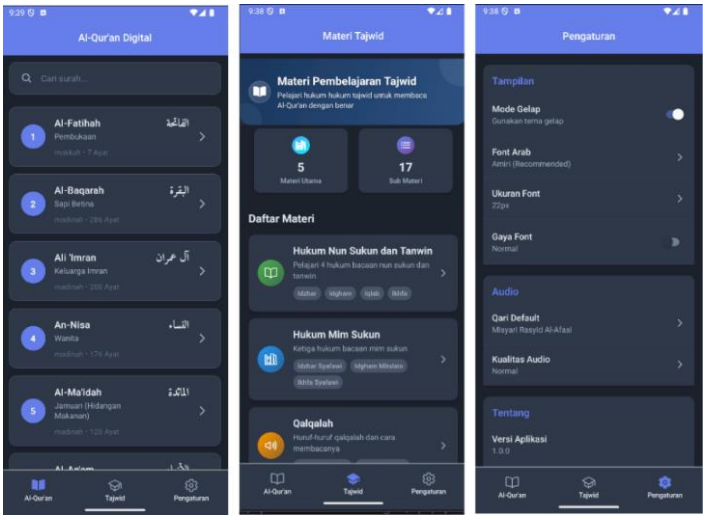

Figure 1. Digital Quran application design image dark mode

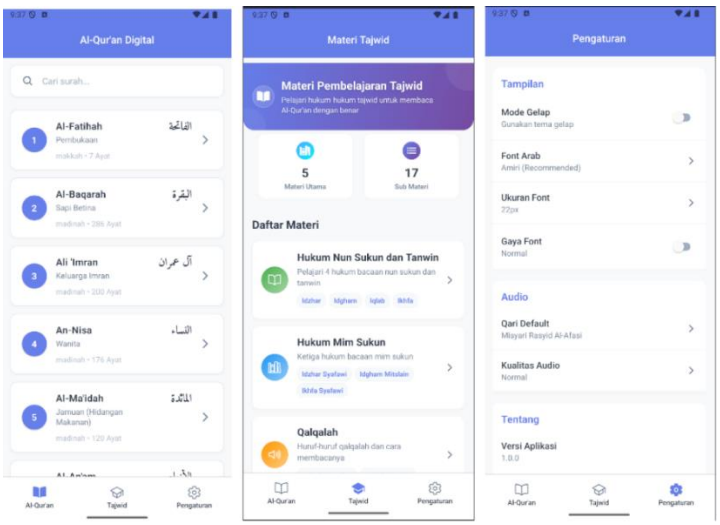

Figure 2. Digital Quran application design image light mode

Halaman daftar surah menyajikan seluruh 114 surah Al-Qur'an dengan nama Latin dan Arab yang dilengkapi fungsi pencarian efisien untuk memudahkan navigasi. Halaman materi tajwid mengorganisir konten pembelajaran secara sistematis dengan topik-topik seperti aturan Nun Sukun dan Tanwin, aturan Mim Sukun, dan Ghunah untuk mendukung pembelajaran tajwid interaktif. Halaman pengaturan menyediakan opsi personalisasi komprehensif melalui pemilihan mode gelap, penyesuaian ukuran font, kustomisasi jenis font, dan konfigurasi kualitas audio. Fleksibilitas ini memastikan pengalaman pengguna yang optimal di berbagai perangkat dan skenario penggunaan.

The design consistency between light and dark modes maintains a high-quality user experience without compromising text readability or navigational ease. The color schemes, typography, and element layouts have been calibrated to provide visual comfort in both modes, enabling a seamless transition. Clear and intuitive icons on the bottom navigation menu facilitate efficient page switching and support responsive interaction.

The application's architecture integrates various technologies, including the **React Native** framework, **JavaScript** programming language, and the **Quran.com** API, to ensure the accurate delivery of Qur'an content, including verse text, translations, and **tilawah** audio. The implementation of colored **tajwid** follows the official guidelines of the Indonesian Ministry of Religious Affairs to guarantee religious accuracy and educational validity.

The interface implementation successfully delivered light and dark modes across all main screens, including the surah list, tajwid material, and settings. The surah list interface displays all 114 surahs with a search function, Arabic and Latin text, and comprehensive information such as the type of revelation (**Makkiyah/Madaniyah**) and the number of verses. The surah details screen integrates Arabic text with colored **tajwid** markers, Indonesian translation, audio controls, and font size adjustment. The implementation of conditional **basmalah** logic for Surah At-Taubah demonstrates attention to the principles of Islamic scholarship. The **tajwid** learning module organizes five main topics with seventeen sub-materials, providing structured access to **tajwid** rules including the rules of **Nun Sukun** and **Tanwin**, **Mim Sukun**, and the principle of **Qalqalah**.

The modular architectural approach facilitates a **maintainable** code structure and enables efficient iterative development. The component-based design supports **reusable** interface elements and systematic testing procedures. The application's technical foundation supports ongoing development and future scalability requirements.

Hasil Pengujian dan Jaminan Kualitas

Unit Testing Performance: Automated testing using the Jest framework achieved comprehensive coverage across 10 files.

Table 2. unit test results metrics

No	File / Folder	% Statements	% Branches	% Functions	% Lines	Uncovered Lines
	All files	77.34	69.33	81.97	77.7	-
1	CompactMusicPlayer.js	56.31	67.15	33.33	57.89	…-127,133-152,158-177,183-188,202,349-357
2	SurahHeader.js	88.37	84.41	100	90.47	17-19,86
3	TajweedDisplay.js	84.82	78.26	88	84.76	111-114,120,150-151,231,241-251,256-257
4	VerseItem.js	75.86	59.77	66.66	80.39	13-14,68-80,93,99,102
5	AudioContext.js	65.77	50	84.84	66.15	…-423,434-442,454,468-470,483,489-496,509
6	FontContext.js	94.82	81.81	100	94.59	123-124,147-148,164-165
7	ThemeContext.js	94.44	100	100	94.44	29,39
8	quranAPI.js	72.28	54.28	84	71.06	…-158,194-195,243-244,322,394,413,451-471
9	TajweedColors.js	97.87	91.48	100	97.8	205-206
10	translationCleaner.js	100	100	100	100	-

Based on the code coverage test results, the following coverage levels were obtained: **77.34%** for statements, **69.33%** for branches, **81.97%** for functions, and **77.7%** for lines of code. These results indicate that the software testing implementation has reached an acceptable level, but it still requires improvement to meet optimal standards in **Object-Oriented JavaScript** application development.

An evaluation of the coverage distribution shows a significant variation between system components. Utility modules like translationCleaner.js achieved perfect coverage (**100%**), while core components like CompactMusicPlayer.js only reached **56.31%** statement coverage. This disparity indicates an imbalance in the testing strategy, where components with high complexity and critical responsibilities have not received adequate testing attention.

An in-depth analysis of the **69.33%** branch coverage reveals that approximately one-third of the program's conditional branches have not been comprehensively tested. In the context of **Object-Oriented JavaScript**, this is a serious concern, as the **OOP** paradigm relies on conditional logic for implementing polymorphism, encapsulation, and error handling. This condition could lead to undetected **runtime errors** and reduce the overall reliability of the system.

Components with low coverage levels, such as AudioContext.js (**65.77%**) and quranAPI.js (**72.28%**), require special attention given their crucial role as infrastructure layers in the application's architecture. In the **Object-Oriented JavaScript** paradigm, context providers and API abstraction layers form the foundation that determines a system's stability and maintainability. Uncovered lines in certain ranges within these components have the potential to cause **cascading failures** that are difficult to identify.

To achieve optimal industry standards, it is necessary to increase coverage to a minimum of **85%** for statements and **80%** for branches. The recommended improvement strategy includes the implementation of comprehensive **unit testing** for public methods, **integration testing** for inter-object interactions, and **end-to-end testing** for user workflows. Special focus should be given to testing interaction patterns, inheritance chains, and method chaining, which are fundamental characteristics of **Object-Oriented JavaScript** implementations.

A more rigorous testing implementation will not only enhance **quality assurance** but also support core software engineering principles such as **maintainability**, **scalability**, and **reliability**, which are the primary goals of **Object-Oriented JavaScript** application development.

Table 3. testing aspects in integration testing

No	Component	Testing Aspects	Number of Tests	Result
1	AppNavigator	Initialization, ThemeContext, StatusBar, NavigationContainer	8	PASS
2	TabNavigator	Tab rendering, styling, navigation to Quran/Tajweed/Settings, Android compatibility	13	PASS

3	SettingsStackNavigator	Rendering, header configuration, theme fallback, error handling, re-render	12	PASS
4	SurahStackNavigator	Navigation to surah details, parameter handling, adaptive theme	9	PASS
5	TajweedStackNavigator	Navigator structure, header configuration, gestures, dark mode	15	PASS
6	SettingsScreen	Main view, dark mode toggling, font/audio settings, error handling	17	PASS
7	SurahDetailScreen	Verse data loading, API fallback, audio playback, error handling	9	PASS
8	SurahListScreen	Surah loading, search feature, navigation, error handling, performance	22	PASS
9	TajweedDetailScreen	Lesson rendering, reading content, scroll/back interaction, testID	16	PASS
10	TajweedLessonsScreen	Rendering, theme, content validation, accessibility, error handling	25	PASS

Then below are the metrics from the integration test results.

Table 4. integration test results metrics

No	File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
1	AppNavigator.js	100	100	100	100	-
2	SettingsStackNavigator.js	100	85.71	100	100	19
3	SurahStackNavigator.js	95.08	20	50	95.08	55-57
4	TabNavigator.js	55.31	100	50	55.31	25-66
5	TajweedStackNavigator.js	95.65	20	100	100	20-32
6	SettingsScreen.js	91.59	58.06	77.41	91.59	...49, 151-153, 167-168, 170-172, 191-199, 201-203, 219-220, 222-231, 233-235, 250, 278
7	SurahDetailScreen.js	93.69	63.46	80	93.69	45-46, 65-66, 83-84, 117-119,

						132-143, 259
8	SurahListScreen.js	100	100	100	100	-
9	TajweedDetailScreen.js	100	86.95	100	100	305-309, 429
10	TajweedLessonsScreen.js	99.66	93.33	100	99.66	285-286

The results of the integration testing provide a comprehensive evaluation of the navigation flow and inter-screen interactions within the mobile application's architecture. Overall coverage indicates a varied level of integration testing completeness across different navigation components and screen modules, revealing both strengths and areas that require improvement at the system integration layer.

Navigation Layer Integration Assessment: The navigation infrastructure shows mixed performance in integration coverage. AppNavigator.js achieved perfect integration coverage (100% across all metrics), indicating that the root navigation logic and routing mechanisms were thoroughly tested for inter-component communication. However, TabNavigator.js showed a significant integration gap with only **55.31%** statement coverage. This indicates that the logic for tab switching, state persistence between tabs, and data sharing mechanisms across tabs were not adequately tested. Uncovered lines (25-66) likely contain critical navigation state management and essential tab lifecycle methods crucial for a seamless user experience.

Stack Navigation Integration Analysis: The stack navigator components revealed partial integration testing implementation. Both SurahStackNavigator.js and TajweedStackNavigator.js showed low branch coverage (20%) despite high statement coverage. This indicates that conditional navigation flows, deep linking scenarios, and error handling during navigation transitions were not comprehensively tested. This pattern suggests that while the main navigation paths function, alternative routes and edge cases in the navigation logic could fail at runtime, especially in scenarios involving backward navigation, passing parameters between screens, and restoring navigation state.

Screen-Level Integration Evaluation: Screen components generally showed strong integration coverage with a few notable exceptions. SurahListScreen.js achieved perfect coverage, indicating robust integration with data providers, navigation services, and UI state management. In contrast, SettingsScreen.js showed moderate integration coverage (91.59% statements, 58.06% branches), with uncovered lines scattered across functional areas including user preference synchronization, theme-switching integration, and cross-screen state propagation. The fragmented range of uncovered lines (151-153, 167-168, 191-199) points to incomplete testing of user interaction workflows and system integration points.

Critical Integration Gaps: The analysis reveals a systematic weakness in branch coverage within the navigation components, indicating inadequate testing of conditional integration scenarios such as authentication-based routing, permission-dependent screen access, and handling of error states during screen transitions. The low branch coverage in stack navigators (20%) represents a significant risk to integration reliability, as these components orchestrate complex inter-screen data flows and state management operations.

Implications for Integration Quality: The current integration testing coverage, while achieving high statement coverage on individual screens, demonstrates insufficient testing of cross-component interactions and system-wide integration scenarios. This pattern indicates that while individual components may function correctly in isolation, their collaborative behavior and integration points could exhibit unexpected failures in a production environment. The perfect coverage achieved by core screens like SurahListScreen.js and TajweedDetailScreen.js provides a solid foundation, but the weakness in the navigation layer could compromise the overall reliability of the system's integration.

To improve the effectiveness of integration testing, the focus should shift to comprehensive branch testing in navigation components, validating end-to-end user journeys, and systematically testing data flows between integrated components.

This approach will ensure a robust integration architecture and maintain consistency across all application layers and user interaction patterns.

User Acceptance Testing (UAT) represents a critical evaluation phase in application development to validate that the developed system meets end-user requirements and expectations. In this study, UAT was conducted through questionnaire distribution using Google Forms to 24 respondents who had utilized the digital Al-Qur'an application.

The testing instrument employed a five-point Likert scale to measure user satisfaction levels across various application aspects. The questionnaire was designed to evaluate seven primary dimensions: (1) navigation and usability, (2) search functionality, (3) Al-Qur'an reading experience, (4) personalization and settings, (5) audio recitation features per verse, (6) tajwid learning materials, and (7) overall application evaluation.

Table 5. likert scale

Category	Description	Score
SA	Strongly Agree	5
A	Agree	4
N	Neutral	3
D	Disagree	2
SD	Strongly Disagree	1

UAT results demonstrate positive response distribution toward the digital Al-Qur'an application. Complete respondent response data is presented in Table 2.

Table 6. Complete respondent response data results

No	Testing Aspect	Question	SA	A	N	D	SD
A	Navigation and Usability						
1	Navigation	Ease of understanding application navigation	11	10	3	0	0
2	Responsiveness	Application response during scrolling	8	11	5	0	0
B	Search Functionality						
3	Accessibility	Ease of finding	7	15	2	0	0

		search feature					
4	Accuracy	Search result relevance	7	15	1	1	0
C	Al-Qur'an Reading Experience						
5	Text Quality	Arabic text display quality	8	13	3	0	0
6	Typography	Font size appropriateness	9	11	4	0	0
7	Visual Aid	Tajwid color effectiveness	4	18	2	0	0
8	Implementation	Tajwid color implementation quality	10	10	4	0	0
D	Personalization & Settings						
9	Display Mode	Dark mode usage experience	11	12	1	0	0
10	Transition	Smooth mode switching	7	13	4	0	0
E	Audio Recitation Feature						
11	Accessibility	Ease of audio feature access	9	13	2	0	0
12	Audio Quality	Recitation audio quality	9	11	4	0	0
F	Tajwid Learning Materials						
13	Navigation	Ease of finding tajwid materials	9	13	2	0	0
14	Content	Material quality and completeness	2	14	8	0	0
15	Comprehension	Ease of understanding explanations	7	15	2	0	0
16	Effectiveness	Benefits for skill improvement	15	8	1	0	0
G	Overall Evaluation						
17	Performance	Overall application performance	7	16	1	0	0
18	Stability	Crash/force close frequency	15	5	4	0	0
19	Speed	Surah and feature loading speed	8	10	6	0	0
20	Functionality	Feature functions meet expectations	13	9	1	0	0
21	Satisfaction	Overall user satisfaction	12	10	2	0	0

To measure user acceptance level, ideal score calculation and assessment percentage were used with the formula:

Ideal Score = Σ Statement Categories × Σ Respondents

Ideal Score = 5 × 24 = 120

Assessment Percentage = (Total Score / Ideal Score) × 100%

Table 7. Result Percentage UAT

No	Aspek	Total Score	Presentace	Category
1	Navigation ease	104	86.67%	Excellent
2	Scroll responsiveness	99	82.50%	Excellent
3	Search feature access	101	84.17%	Excellent
4	Search accuracy	100	83.33%	Excellent
5	Arabic text quality	101	84.17%	Excellent
6	Font size	101	84.17%	Excellent
7	Tajwid color assistance	98	81.67%	Excellent
8	Tajwid implementation	102	85.00%	Excellent
9	Dark mode	106	88.33%	Excellent
10	Mode switching	99	82.50%	Excellent
11	Audio access	103	85.83%	Excellent
12	Audio quality	101	84.17%	Excellent
13	Tajwid material navigation	103	85.83%	Excellent
14	Material completeness	90	75.00%	Good
15	Tajwid comprehension	101	84.17%	Excellent
16	Learning effectiveness	110	91.67%	Excellent
17	Application performance	102	85.00%	Excellent
18	Application stability	107	89.17%	Excellent
19	Loading speed	98	81.67%	Excellent
20	Feature functionality	106	88.33%	Excellent
21	Overall satisfaction	110	91.67%	Excellent

UAT results demonstrate high acceptance levels toward the digital Al-Qur'an application with an average satisfaction percentage of 84.84%. Among 21 tested aspects, 20 aspects (95.24%) achieved "Excellent" category with percentages above 80%, while 1 aspect (4.76%) achieved "Good" category with 75%.

The highest-rated aspects were learning material effectiveness and overall user satisfaction (91.67%), indicating that the application successfully achieved its primary objective as an Al-Qur'an learning medium. The lowest-rated aspect was tajwid material completeness (75%), indicating an area requiring further development.

Overall, UAT results confirm that the digital Al-Qur'an application has met user acceptance criteria and is suitable for implementation in a production environment. The high satisfaction scores across multiple dimensions validate the application's design decisions and functional implementation.

Key findings from the UAT include:

1. **Navigation and Usability Excellence**: Users found the application interface intuitive and responsive, with navigation clarity scoring 86.67%.
2. **Functional Reliability**: Search functionality and core features performed according to user expectations, with most aspects scoring above 83%.
3. **Enhanced Reading Experience**: The combination of Arabic text quality, appropriate typography, and tajwid color coding significantly improved the Qur'an reading experience.
4. **Effective Learning Tools**: The tajwid learning materials proved highly effective for skill development, achieving the highest satisfaction score of 91.67%.
5. **Technical Performance**: Application stability and performance met user requirements, with minimal technical issues reported.

These results provide strong evidence for the application's readiness for deployment and its potential impact on digital Qur'an learning experiences.

Development Methodology and Application Significance Evaluation

The **PXP** methodology has proven effective for developing mobile applications of medium complexity. Its iterative approach allows for systematic feature development with a clear, priority-based implementation. Key success factors of the methodology include:

a) Structured **requirements gathering** with dual validation sources.

b) Effective **risk assessment and time estimation** using an ideal-day approach.
c) Successful **integration of continuous testing** throughout the development lifecycle.
d) Comprehensive **documentation** enabling efficient decision tracking.

Areas for methodological improvement include the need for additional **buffer** time in technically complex iterations, particularly for *tajwid* parsing and audio integration features.

Application Impact and Significance

The developed application successfully addresses gaps identified in existing Quran reading applications through innovative features, including interactive *tajwid* color-coding, comprehensive learning materials, and personalized user experience options. The integration of educational content with reading functionality creates a comprehensive platform for Quranic study and practice.

User feedback validation confirms the application's effectiveness in supporting proper Quranic pronunciation and understanding through visual *tajwid* indicators. A high satisfaction score for learning effectiveness (91.67%) demonstrates significant educational value for users seeking to improve their Quran reading skills.

The successful implementation showcases the viability of a modern mobile development approach for Islamic educational applications, providing a foundation for future enhancements and the development of similar projects. The comprehensive testing framework ensures application reliability and user satisfaction across various usage scenarios.

CONCLUSIONS AND RECOMMENDATIONS

Conclusion

Based on the research results and discussion presented in previous chapters, the following conclusions can be drawn:

a) The digital Al-Qur'an application was successfully designed and developed using the **Personal Extreme Programming (PXP) method**. This method

proved to be effective for small-scale development due to its iterative nature and flexibility toward changes. The application development was completed in 5 main iterations over a 34-day period (approximately 1 month). The core features, including colored *tajwid*, *tajwid* material menus, and *tilawah* audio, were successfully implemented to meet user needs.

b) The application was tested using two methods: **automation testing** and **User Acceptance Testing (UAT)**. For **automation testing**, the unit test results showed a high level of reliability. The average success rates were 77.34% for statements, 69.33% for branches, 81.97% for functions, and 77.70% for lines. This indicates that most of the program's functions and logic paths were well-tested, though some parts, particularly branch coverage, still need improvement. Integration testing yielded very satisfactory results, with most files achieving over 90% coverage, showing optimal integration between modules.

The **UAT** results showed that the digital Al-Qur'an application was rated very well by users. Out of 21 questions distributed to 24 respondents, most aspects tested received a percentage above 80% and were categorized as "Very Good." The highest score (91.67%) was for how much the *tajwid* material helped improve Al-Qur'an reading skills, while the lowest (75%) was for the quality and completeness of the *tajwid* material, which was still in the "Good" category. Overall, 20 out of 21 questions received a "Very Good" rating, demonstrating that the application met user expectations. In summary, this digital Al-Qur'an application successfully achieved its research objectives by providing an effective and interactive solution for learning *tajwid*.

Recommendations

Based on the research and implementation that has been carried out, the author provides the following recommendations for further development and research:

a) **Refinement of the Digital Al-Qur'an application**, as the results from automated unit and integration testing were still relatively low.

b) **Development of a backend system**, as this application is currently a pure front-end application using only *fetch* APIs.

c) **Further development of the application** by adding features such as reading practice with audio feedback and tiered learning modules to make it more educational and engaging.

d) The application is still considered a **prototype**. To be fully implemented, it requires more in-depth research, security testing, and performance optimization to ensure it can be used by a wide range of Android users and is sustainable.

REFERENCES

Afrianto, I., Heryandi, A., Finadhita, A., & Atin, S. (2021). *User Acceptance Test For Digital Signature Application In Academic Domain To Support The Covid-19 Work From Home Program*. *5*(36).

Kasoni, D., Liesnaningsih, L., & Afif, F. F. (2024). Perancangan Sistem Pembelajaran Al-Quran Berbasis Android Dengan Metode Extreme Programming. *JIKA (Jurnal Informatika)*, *8*(1), 89. https://doi.org/10.31000/jika.v8i1.10270

Kumar, D., & Mishra, K. K. (2016). The Impacts of Test Automation on Software's Cost, Quality and Time to Market. *Procedia Computer Science*, *79*, 8–15. https://doi.org/10.1016/j.procs.2016.03.003

Kurniawan, K., & Yulhendri, Y. (2023). Pemanfaatan Framework React Native dalam Perancangan Aplikasi Penjualan Merchandise. *NUCLEUS*, *4*(2), 84–97. https://doi.org/10.37010/nuc.v4i2.1369

Lajnah Pentashihan Mushaf Al-Qur′an. (2011). *Pedoman Tajwid Sistem Warna*. https://tashih.kemenag.go.id/uploads/1/2019-08/buku_pedoman_tajwid_sistem_warna.pdf

M. Alagrami, A., & M. Eljazzar, M. (2020). Smartajweed Automatic Recognition of Arabic Quranic Recitation Rules. *Computer Science & Information Technology (CS & IT)*, 145–152. https://doi.org/10.5121/csit.2020.101812

Mahkfudz, Nurpriatna, A., & Palah. (2021). Integrasi Teknologi Dalam Pembelajaran Ilmu Tajwid Untuk Meningkatkan Kemampuan Membaca Al Qur`an. *Rayah Al-Islam*, *5*(02), 779–791. https://doi.org/10.37274/rais.v5i02.714

Melia, S., & Putra, F. P. (2023). *Comparative Analysis of Automated Testing Tools on GUI WEB-Based Applications*.

Oktarina, M. (2020). Faedah Mempelajari dan Membaca Al-Quran dengan Tajwid. *Serambi Tarbawi*, *8*(2), 147–162. https://doi.org/10.32672/tarbawi.v8i2.5072

Oladapo Adeboye Popoola, Henry Ejiga Adama, Chukwuekem David Okeke, & Abiodun Emmanuel Akinoso. (2024). Conceptualizing Agile Development In Digital Transformations: Theoretical Foundations And Practical Applications. *Engineering Science & Technology Journal*, *5*(4), 1524–1541. https://doi.org/10.51594/estj.v5i4.1080

Sari, I. P., Purnama, I., & Ritonga, A. A. (2021). Implementasi API pada Aplikasi Al-Qur'an Berbasis Android dengan Metode UCD. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, *5*(2), 615. https://doi.org/10.30865/mib.v5i2.2913

Setiawan, R. (2021, October 19). *Apa Itu React Native? Apa Kelebihan dan Kekurangannya?* Dicoding Blog. https://www.dicoding.com/blog/apa-itu-react-native/

Sholeh, M., Ridhoni, I. W., & Basuki, U. J. (2022). Pengembangan Aplikasi Alquran Online Dengan Memanfaatkan Rest Api. *Device*, *12*(2), 1–9. https://doi.org/10.32699/device.v12i2.2861

Syaifullah, M., Jannah, M., Fradila, N., Ningrum, P. P., & Nasution, W. (2022). *The Implementation Of The Science Of Tajwd In Learning Of The Al-Qur'an With The Tajwid Wheel Method To Develop The Reading Of The Qur'an In*.

Ulfi, M., Marthasari, G. I., & Nuryasin, I. (2020). *Implementasi Metode Personal Extreme Programming dalam Pengembangan Sistem Manajemen Transaksi Perusahaan*. *2*(3).

Wibowo, M. I., Brata, A. H., & Brata, K. C. (2019). *Pengembangan Aplikasi Perangkat Bergerak Pengobatan Tuberculosis Berbasis Android Menggunakan Personal Extreme Programming (Studi Kasus: Puskesmas Polowijen)*.

Yani, A., Putra, H., Andika, A., Nisa, M. K., & Yunus, E. M. (2021). Studi Perbandingan Fitur-Fitur Aplikasi Al-Quran Digital Karya Greentech Apps Foundation dan Aplikasi Al-Quran Muslim Media untuk Mengetahui Perbedaan Kedua Fitur aplikasi. *Jurnal Riset Agama*, *1*(3), 132–156. https://doi.org/10.15575/jra.v1i3.15089

Yaqin, Moh. A. (2022). Aplikasi Go-Qur'an Berbasis Web dan android. *TRILOGI: Jurnal Ilmu Teknologi, Kesehatan, dan Humaniora*, *3*(1), 18–21. https://doi.org/10.33650/trilogi.v3i1.3644