Development Journey: A Reflective Journal on Mobile Application Design and Implementation

Paul Joseph Lawrance

Sahyajothi Arts and Science College, 1st mile, Kumily - 685509, Idukki District, Kerala, India
Email: *pauljosephlawrance[at]gmail.com*

Abstract: ***This study presents a reflective analysis of the design, implementation, and pedagogical outcomes of a Mobile Application Development course offered under the newly introduced four-year undergraduate honours framework in Kerala, India. Employing a mixed-methods approach over six months with eight undergraduate students, the study examined instructional practices, student preparedness, infrastructural constraints, and learning outcomes. A student-centered and experiential learning model was adopted, integrating blended instruction, flipped-classroom strategies, gamification-based formative assessment, AI-assisted coding tools, and structured laboratory practice using Android Studio and Kotlin. Initial infrastructural challenges, particularly limited hardware capacity, were addressed through resource optimization and technical intervention, enabling effective practical engagement. Student performance was evaluated through quizzes, class tests, application-based tasks, and oral examinations. Results indicate satisfactory to high levels of achievement, with consistent performance across cognitive and applied domains. Students demonstrated competence in GUI design, event handling, graphical rendering, GPS integration, file management, debugging, and collaborative development. The findings highlight the importance of aligning curriculum reform with pedagogical innovation, infrastructural preparedness, and continuous assessment strategies. The study contributes to ongoing discussions on experiential and technology-enhanced learning in undergraduate computer science education within emerging institutional contexts.***

Keywords: Mobile Application Development; Student-Centered Learning; Experiential Learning; Android Studio; Kotlin Programming; Gamification; AI-Assisted Learning; Undergraduate Honours Education; Blended Learning; Technical Education Reform.

1. Introduction

In the contemporary Information Age, knowledge production and the rapid circulation of information significantly influence social, economic, and technological transformations. Higher education institutions are therefore compelled to redesign curricula and pedagogical practices to equip learners with the competencies required for critical thinking, technological fluency, and informed decision-making. Within this context, undergraduate education in India has recently undergone structural reforms aimed at enhancing flexibility, interdisciplinarity, and research integration.

Sahyajyothi Arts and Science College, affiliated with Mahatma Gandhi University and certified under ISO 9001:2015 standards, offers several undergraduate honours programmes, including Bachelor of Arts in Communicative English, Bachelor of Social Work (Honours), Bachelor of Commerce (Honours), Bachelor of Business Administration (Honours), and Bachelor of Computer Science (Honours) (Sahyajyothi, 2024). Beginning in the 2024–2025 academic year, Mahatma Gandhi University implemented a four-year undergraduate honours framework in alignment with national higher education reforms. As announced by Higher Education Minister R. Bindu, the revised structure allows students to exit after three years with a degree certificate or complete a fourth year focused on research and internships to obtain an honours degree. The curriculum framework, developed through field assessment and contextual benchmarking against national and global trends, emphasizes active learning, disciplinary depth, and research orientation.

Despite curriculum reforms and the increasing integration of experiential learning in computer science education, limited empirical research has examined how newly implemented four-year honours frameworks influence classroom practices and student engagement in technical courses such as Mobile Application Development, particularly within emerging institutional contexts in Kerala. Against this institutional setting, the present study examines teaching and learning practices in the course *Mobile Application Development* offered to second-year Bachelor of Computer Science (Honours) students. The course introduces foundational concepts in mobile application development, including user interface design, data management, Android architecture, web services integration, and application deployment. University-recommended readings include *Head First Android Development: A Brain-Friendly Guide* (Griffiths & Griffiths, 2021) and *Android Studio 3.0 Development Essentials* (Smyth, 2017), which provide theoretical grounding and practical guidance in Android-based development. Given the growing relevance of mobile technologies in digital ecosystems- particularly in areas such as customer engagement, data analytics, and personalized digital services- the course plays a crucial role in developing industry-relevant competencies among undergraduate learners.

This study addresses the following research questions:
1) What are the existing pedagogical practices in teaching Mobile Application Development?
2) What is the level of students' prior technological preparedness?
3) How do institutional and infrastructural factors influence experiential learning outcomes?

The findings are intended to inform evidence-based pedagogical refinements, strengthen student learning outcomes, and guide the development of targeted academic support strategies. The course is designed around an integrated pedagogical framework that synthesizes empirical inquiry, inquiry-based learning, reflective practice, and experiential learning. Through structured exploration of

Android architecture and development tools, students engage in guided, hands-on activities to design, develop, and deploy functional mobile applications. This study employed a mixed-methods design and was conducted over six months with a cohort of eight students enrolled in the Mobile Application Development course of under graduate program.

2. Methodology

The findings of the preliminary study indicated the need to streamline the teaching and learning process through the adoption of a student-centered pedagogical approach. Classroom observations revealed that students experienced difficulty in recalling and connecting knowledge acquired in previous semesters, and their participation in discussions was limited. Although students demonstrated positive behavioural attributes—being respectful, attentive, and willing to learn—their engagement levels suggested the necessity for instructional redesign. Establishing an inclusive classroom environment extends beyond academic delivery; it involves fostering a supportive atmosphere in which learners feel valued and understood (Osmond, 2025). A student-centered approach was therefore adopted to encourage learners to actively construct knowledge, engage in reflective thinking, and take ownership of their learning process.

In addition to the prescribed course syllabus, both the instructor and students utilized university-recommended online learning resources to support instructional delivery. Given the digital nature of these materials, the use of smartphones in the classroom was permitted as a learning tool. Contemporary students are generally technologically adept and inclined to access information through online platforms (Sawarkar et al., 2019). Core theoretical components of the course- including Mobile Applications and Device Platforms, Comparative Analysis of Native and Hybrid Applications, the Mobile Application Development Lifecycle, and Mobile Application Front-End and Back-End Architecture- were delivered through blended instructional strategies integrating laptops and smartphones. To enhance engagement and promote meaningful classroom discourse, students were assigned preparatory readings prior to each session. This flipped-classroom element facilitated informed discussions and contributed to a more interactive and effective teaching–learning environment.

Upon completion of Unit 1.1 of the Mobile Application Development course, a formative assessment was conducted using the gamification-based platform Quizalize. The quiz consisted of 32 items, incorporating multiple-choice, true/false, and short-response formats, and was designed to evaluate students' conceptual understanding of the unit. The assessment results, which are presented in Table 1, were subsequently analysed to inform instructional adjustments and measure the effectiveness of the implemented pedagogical strategies.

Table 1: Quiz results of Unit 1.1

Serial No	Student name	Total correct answers out of 32 questions	Percentage
1	Student A	19	59.4
2	Student B	23	71.9
3	Student C	25	78.1
4	Student D	26	81.3
5	Student E	27	84.4

Table 1 presents the results of the Unit 1.1 quiz, which consisted of 32 questions. Although the class comprised eight students, only five students were present for the assessment. The results, therefore, reflect the performance of the students in attendance.

The scores ranged from 19 to 27 correct responses. The highest score was achieved by Student E, who obtained 27 out of 32 correct answers (84.4%), followed by Student D with 26 correct responses (81.3%). Student C scored 25 (78.1%), and Student B obtained 23 correct answers (71.9%). The lowest score was recorded by Student A, with 19 correct responses (59.4%).

Overall, the findings indicate moderate to high academic achievement among the participating students, with four out of five students scoring above 70%. The mean score for the group was 24 out of 32 (75.0%), suggesting satisfactory understanding of the Unit 1.1 concepts. However, the variation in individual performance highlights differences in conceptual mastery and suggests the need for differentiated instructional support to address learning gaps.

Effective teaching and learning of the Mobile Application Development course require adequate instructional resources, structured course materials, reliable computing infrastructure, and sustained student engagement. Given the practice-oriented nature of the subject, emphasis is placed on hands-on laboratory experience to ensure the development of technical competencies. To facilitate practical implementation, the institution must provide well-equipped computer laboratory facilities that meet the technical requirements of contemporary mobile application development tools.

In alignment with the course specifications, the university recommended the use of Android Studio as the primary Integrated Development Environment (IDE) for mobile application development (Android Studio, 2025). The minimum system requirements for installing Android Studio include a 64-bit operating system (Windows 8/10/11 or a modern Linux distribution), an x86_64 processor with virtualization support, 8 GB of RAM, 8 GB of available disk space, a minimum screen resolution of 1280 × 800, and at least an Intel Core i3 (2nd generation or newer) or an equivalent AMD processor.

An initial audit of the institutional computer laboratory revealed that the systems were configured with Windows 10 (64-bit) on x86_64 processors with virtualization support, 8 GB of disk space, and the required screen resolution; however, the machines were equipped with only 4 GB of RAM. Following the installation of Android Studio, it was observed that the systems were unable to efficiently execute even basic mobile application projects due to insufficient memory capacity.

To address this limitation, a structured technical workshop was conducted in the computer laboratory. The session aimed to enhance students' technical skills by engaging them in analysing system configurations and examining hardware components, particularly the installed memory modules. During the supervised disassembly of selected systems, students demonstrated their understanding of internal computer components and their functional principles. The inspection revealed that several non-operational systems contained functional 4 GB DDR4 RAM modules. These modules were repurposed to upgrade the operational systems in the laboratory.

Subsequent to the installation of an additional 4 GB of RAM, students verified the updated system configurations through the operating system settings, confirming a total memory capacity of 8 GB. The upgrade significantly improved system performance, enabling the successful execution of Android Studio and facilitating the development of students' first mobile applications.

In accordance with Unit 1.2 of the course syllabus, instruction subsequently covered core mobile application services and development concepts, including the complete application development lifecycle, UI/UX design principles, comparative approaches to native and hybrid development for iOS and Android platforms, Application Programming Interface (API) integration, and cloud-based backend services. Foundational topics- such as an introduction to the Android platform, installation and configuration of development tools, exploration of the IDE environment, debugging techniques, and application deployment—were systematically demonstrated during laboratory sessions to reinforce experiential learning outcomes.

A class test was administered to evaluate students' learning outcomes in Units 1.1 and 1.2 of the Mobile Application Development courses. The assessment comprised 20 marks and was designed to measure both conceptual understanding and applied knowledge. The results of the test are presented in Table 2.

Table 2: Class Test1 of Unit 1.1

Serial No	Student name	Total correct answers out of 20 marks	Percentage
1	Student A	13	65
2	Student B	15	75
3	Student C	18	90
4	Student D	18	90
5	Student E	19	95
6	Student F	13	65
7	Student G	16	80
8	Student H	13	65

The findings indicate satisfactory academic performance across the cohort. All eight students scored 65% or above, with marks ranging from 13 to 19 out of 20. Three students achieved 90% or higher, demonstrating strong mastery of the subject matter. Overall, the results suggest that students developed a sound understanding of the Android Studio interface and its core functionalities. Furthermore, their written responses reflected an appropriate comprehension of the stages involved in the mobile application development lifecycle, as prescribed in the course syllabus.

In alignment with the university's recommended course materials, students were provided with curated online resources to support foundational learning in Kotlin programming. However, during practical laboratory sessions, it was observed that executing even short Kotlin code snippets in Android Studio required considerable system processing time, particularly on systems with limited hardware capacity. This delay affected the pace of hands-on activities and reduced immediate feedback during coding exercises.

To mitigate this challenge, an alternative instructional strategy was implemented through the introduction of Kotlin Playground. Kotlin Playground is an online coding environment that enables users to write, compile, execute, and share Kotlin programs without requiring local installation of compilers or integrated development environments (Kodeco Inc, 2025). The platform was introduced to facilitate beginner-level programming practice, allowing students to experiment with variable declarations, input–output statements, functions, and conditional constructs.

Although the platform supported rapid code execution, students encountered difficulties when handling user input during runtime. Specifically, errors occurred when attempting to provide dynamic input values through the console interface. Despite attempting multiple debugging strategies, the issue persisted for several students, which limited the effectiveness of the platform for certain programming tasks.

Consequently, another online coding platform, Programiz, was introduced as a supplementary learning tool. Programiz is an interactive web-based platform that supports multiple programming languages, including Kotlin, and provides built-in compilers accessible via web and mobile interfaces (Programiz, n.d.). Through this platform, students practiced writing and executing Kotlin code snippets to validate program outputs. This approach enabled them to compare Kotlin syntax and structural elements with previously learned programming languages such as C and Python. The integration of Programiz enhanced students' confidence and improved their fluency in Kotlin programming concepts.

To further strengthen coding proficiency, the course adopted an experiential learning approach by transforming theoretical instruction into structured hands-on laboratory sessions. Although the institution provided reliable internet connectivity- an essential requirement for mobile application development- the limited availability of computer systems meeting the minimum specifications for Android Studio posed logistical constraints. To ensure equitable access to resources, students were divided into two groups, and practical sessions were conducted in a staggered format. This arrangement optimized laboratory utilization while maintaining instructional effectiveness.

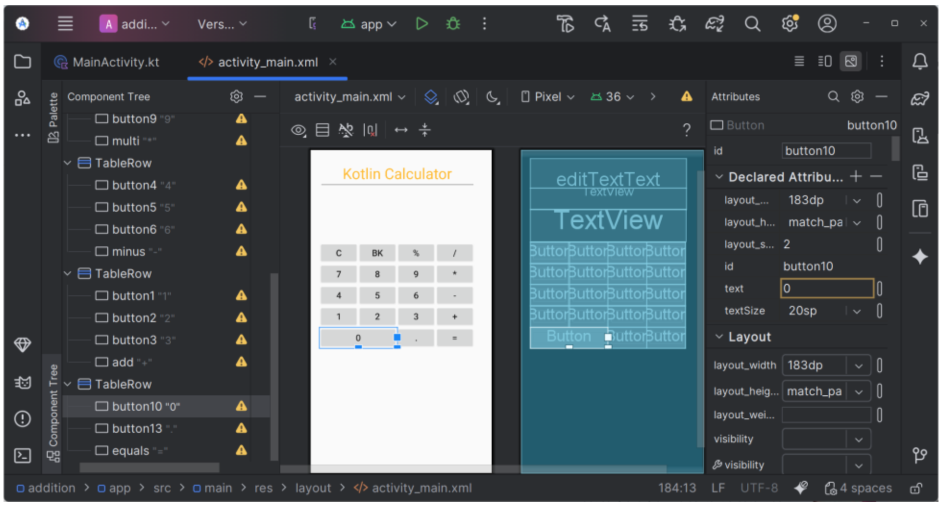
In groups, students developed mobile apps as per the course's practical questions. Out of ten questions, randomly select five questions for the article as shown below. Every question was explained and guided. With the help of AI tools, students developed mobile applications. After the completion of each app, a 5-minute oral exam was conducted for each group. The results are shown in Table 3.

1) Develop an application that uses Graphical User Interface (GUI) components, fonts, and colours.

Students demonstrated strong familiarity with Android Studio and its associated tools, developed through structured theoretical instruction and extensive practical sessions. Observations indicated that students frequently used AI-based tools to generate Kotlin and XML code, which they integrated into Android Studio with minimal difficulty and were able to resolve the resulting errors effectively. To assess their coding competencies, targeted questions on GUIs were posed to evaluate their understanding of code structure, purpose, and functionality, and students responded satisfactorily. Furthermore, students were informed in advance that conceptual questions would follow the completion of the application during the practical session. This approach encouraged active engagement and facilitated a deeper understanding of the coding process.

2) Develop a native calculator application that uses Layout Managers and event listeners.

The students designed a native calculator application using a linear layout that incorporates TextView, TableLayout, and Button components. Within the TableLayout, the buttons were arranged systematically to create an organized, user-friendly interface. The design demonstrates effective use of the Palette window, Component Tree window, and Attributes window, as illustrated in the screenshot below. The component tree reflects the structure of the calculator in the design view and automatically generates the corresponding XML code. Furthermore, the design can be efficiently enhanced by adjusting properties through the Attributes window or by directly modifying the XML layout file.



Component Tree
- button9 "9"
- multi "*"
- TableRow
  - button4 "4"
  - button5 "5"
  - button6 "6"
  - minus "-"
- TableRow
  - button1 "1"
  - button2 "2"
  - button3 "3"
  - add "+"
- TableRow
  - button10 "0"
  - button13 "."
  - equals "="

Attributes — Button button10
id: button10
layout_...: 183dp
layout_h...: match_pa
layout_s...: 2
id: button10
text: 0
textSize: 20sp
layout_width: 183dp
layout_heig...: match_pa
layout_wei...:
visibility:
visibility:

Figure 1: Screenshot of the development stage of the lculator application

Students can create a Kotlin file in Android Studio as part of a new project by following a sequence of standard setup steps. The generated Kotlin file includes a default "Hello World" program, which serves as a foundation for further application development through incremental modifications. Students demonstrated proficiency in basic Kotlin programming concepts, such as variables, data types, operators, and conditional statements, due to their prior exposure to Python programming in the previous semester. This prior knowledge supported their ability to recall fundamental programming constructs and adapt to Kotlin syntax and semantics.

However, challenges were observed in understanding Kotlin functions and lambda expressions, particularly in event handling, which is essential for developing a native calculator application. To address this, event handling was demonstrated in a smart classroom environment using a simple example: adding two numbers and displaying the result when the equal button is pressed. The implementation employed functions and lambda expressions to perform the calculation. This instructional approach enabled students to comprehend event handling mechanisms, lambda expressions, and control flow in Kotlin programming, thereby allowing them to complete the calculator application.

3) Write an application that draws basic graphical primitives on the screen.

This task required students to configure the graphical interface using a RelativeLayout, enabling them to position user interface components in relation to one another. Students utilized built-in Android graphics classes by importing the android.graphics package, including Canvas, Color, Paint, and Bitmap, to construct graphical primitives such as lines, circles, and rectangles. Drawing operations were implemented using Kotlin functions, including canvas.drawLine(), canvas.drawCircle(), and canvas.drawRect() to render geometric shapes on the screen.

Beyond the basic primitives, students extended their implementations to create more complex shapes—such as triangles, hexagons, and rhombuses—by strategically combining multiple canvas.drawLine() function calls. This exercise facilitated a deeper understanding of graphical primitives and enhanced students' ability to effectively programmatically render geometric forms in screen space.

4) Develop a native application that uses GPS location information.

Students demonstrated an understanding of the fundamental concepts underlying the Global Positioning System (GPS), which determines geographic location in terms of latitude and longitude using satellite signals and network-based services (High Accuracy mode). This exercise enabled students to explore the core principles of location tracking in Android applications by accessing fine and coarse location data through Google Play Services and the Fused Location Provider API.

To ensure proper functionality, the required location permissions were declared in the AndroidManifest.xml file. The user interface was designed to include a button for initiating location retrieval and two text fields for displaying the corresponding latitude and longitude values. During implementation, GPS services were enabled on the device, while emulator-based testing involved manually configuring geographic coordinates. Upon execution, the application successfully retrieved and displayed the current latitude and longitude values, thereby reinforcing students' practical understanding of mobile-based location services.

5) Implement an application that writes data to the SD card (Secure Digital Card).

This question required students to recall and apply their prior knowledge of file-handling concepts learned in Python programming during the previous semester. Specifically, they examined the functionality for saving text data to external storage locations, such as device storage, SD cards, or public directories. To implement this functionality in a mobile application context, students designed a user interface using an XML layout file. The interface included an input text field for data entry, a submission button to trigger the save operation, and a text view to display status messages. Upon a button click event, a listener function was invoked to retrieve the user-entered text and pass it to a dedicated function, saveToExternalStorage(data: String).

Within this function, an external storage directory was created (if not already present), followed by the creation of a text file. A FileOutputStream object was instantiated using the constructor call FileOutputStream(file). This object was then used to invoke the write() method, which writes the provided string data—converted into a byte array—into the file. Finally, the close() method was called to properly release system resources and ensure data integrity.

It was observed that students understood the fundamentals of software architecture, coding principles, and software design techniques. They also gained exposure to Gradle (the build system), external libraries, and components. In the aspect of the user interface, students learnt to design responsive interfaces by working with XML layouts and worked on adaptive layouts for multiple screen sizes. Students effectively used various layouts and modern user interface components, which reflects their accessibility understanding and awareness of user-centered design by improving their visual thinking skills.

In terms of testing and quality assurance, students gained exposure to the user interface testing, handled errors using debugging tools, and used an emulator to view a virtual screen and connected to a mobile phone for real-device testing. It enabled students to build a quality mindset and reliability awareness. As students worked in groups, they developed teamwork skills to achieve the goal.

In alignment with the stated course outcomes, a structured set of oral questions was designed and administered to assess students' competencies across multiple cognitive and affective domains, including remembering, understanding, applying, analyzing, evaluating, creating, skill development, interest, and appreciation. The results of this assessment are presented in Table 3.

Table 3: Oral exam marks based on the practical work

Serial No	Student	Oral Exam Out of 20 marks
1	Student A	17
2	Student B	16
3	Student C	17
4	Student D	18
5	Student E	18
6	Student F	16
7	Student G	17
8	Student H	15

Table 3 presents the oral examination scores (out of 20 marks) for eight students (A–H). The marks range from 15 to 18, indicating generally strong performance across the cohort.

The highest score (18 marks) was achieved by Students D and E, demonstrating the strongest oral performance. A score of 17 marks was obtained by Students A, C, and G, representing a high level of competence. Students B and F scored 16 marks, reflecting satisfactory performance, while Student H obtained the lowest score (15 marks), though still within a good performance range. The mean score of the group is 16.75 marks, with a range of 3 marks (15–18), suggesting low variability in performance. The clustering of scores between 16 and 18 indicates consistent achievement among students, with no extreme deviations.

Overall, the results suggest that the majority of students performed well in the oral examination component based on the practical, demonstrating a strong and relatively uniform level of understanding and presentation skills.

3. Conclusion

This reflective study examined the pedagogical processes, infrastructural challenges, and student learning outcomes associated with the Mobile Application Development course offered under the newly implemented four-year undergraduate honours framework. The findings demonstrate that the integration of student-centered learning, blended instructional strategies, gamification-based formative assessment, and experiential laboratory practice contributed positively to student engagement and competency development.

Despite initial infrastructural constraints- particularly limited hardware capacity- the adaptive strategies implemented, including laboratory resource optimization, hardware upgrades, and the incorporation of alternative online coding platforms, ensured continuity and effectiveness in practical instruction. The introduction of platforms such as Kotlin Playground and Programiz supplemented Android Studio–based development and supported incremental skill acquisition in Kotlin programming.

Assessment results from quizzes, class tests, practical applications, and oral examinations consistently indicate satisfactory to high levels of academic achievement across the cohort. The relatively narrow range of scores and strong clustering around higher performance bands suggest conceptual clarity, technical competence, and consistent learning outcomes. Moreover, students demonstrated the ability to apply theoretical knowledge to real-world application development tasks, including GUI design, event handling, graphical rendering, GPS integration, and external storage management.

Beyond technical proficiency, the course fostered collaborative learning, reflective practice, problem-solving skills, and adaptability- competencies aligned with the objectives of the revised undergraduate honours curriculum. The integration of AI-assisted coding tools further reflects contemporary industry practices and highlights the evolving nature of programming pedagogy.

Overall, this study underscores the importance of aligning curriculum reform with pedagogical innovation, infrastructural preparedness, and continuous assessment strategies. Future research may expand the sample size, incorporate longitudinal tracking of skill development, and explore the impact of AI-supported programming environments on independent coding proficiency. The findings contribute to the growing discourse on experiential learning in computer science education within emerging institutional contexts and offer practical insights for strengthening mobile application development pedagogy at the undergraduate level.

References

[1] *Android Studio*. (2025, September 18). Retrieved from Developers: https://developer.android.com/studio/install

[2] Griffiths, D., & Griffiths, D. (2021, November). *Head First Android Development, 3rd Edition*. Retrieved from oreilly: https://www.oreilly.com/library/view/head-first-android/9781492076513/

[3] Kerala State Curriculum Committee for Higher Education. (2023). *KERALA STATE HIGHER EDUCATION CURRICULUM FRAMEWORK FOR UNDERGRADUATE PROGRAMMES*. Retrieved from Kerala State Curriculum Committee for Higher Education: https://www.old.kshec.kerala.gov.in/images/pdf/Curriculum_Framework_Final_-_03102023.pdf#:~:text=New%20Curriculum%20Framework%20shall%20give%20great%20importance,of%20credits%20for%20all%20universities%20to%20adopt.

[4] Kodeco Inc. (2025, December 19). *Kotlin Playground: Getting Started*. Retrieved from Kodeco - The new raywenderlich.com: https://www.kodeco.com/16929465-kotlin-playground-getting-started

[5] Osmond, P. (2025, November 17). *Top Strategies for Managing Behavior in Inclusive Classrooms*. Retrieved from LinkedIn: https://www.linkedin.com/feed/update/urn:li:activity:7396088019142721538/?originTrackingId=ilQb%2BH%2Fl6o7wf5J0vFpovg%3D%3D

[6] Programiz. (n.d.). *Learn programming for free*. Retrieved from Programiz: https://www.programiz.com/

[7] Sahyajyothi. (2024). *Who we are*. Retrieved from Sahyajyothi: https://www.sahyajyothi.ac.in/

[8] Samsukha, A. (2024). *emizentech*. Retrieved 4 11, 2025, from Why is mobile application development is important in 2024?: https://emizentech.com/blog/why-is-mobile-app-development-important.html

[9] Sawarkar, G., Desai, P. R., & Sawarkar, P. (2019). Effectiveness of Mobile App as a Teaching and Learning Tool. *Journal of Indian System of Medicine, 7*(2), 104-111. https://doi.org/10.4103/JISM.JISM_36_19

[10] Smyth, N. (2017). *Android Studio 3.0 Development Essentials Android 8th Edition.* Payload Media, Inc. All Rights Reserved. https://doi.org/http://repo.darmajaya.ac.id/5623/1/Android%20Studio%203.0%20Development%20Essentials%20%20-%20Android%208%20Edition%20%28%20PDFDrive%20%29.pdf

Author Profile

Dr. Paul Joseph Lawrance, Assistant Professor, Sahyajothi Arts and Science College, 1st mile, Kumily - 685509, Idukki District, Kerala, India
Email: pauljosephlawrance[at]gmail.com