

## **DESIGN AND EVALUATION OF AN OFFLINE ANDROID-BASED ACADEMIC SCHEDULING SYSTEM WITH AUTOMATED CONFLICT DETECTION AND REMINDER MECHANISM**

**\*Adi Alfaisal, Nur Fadilah<sup>1</sup>, Awalul Syawal<sup>2</sup>, Nur Fitrianiingsih<sup>3</sup>**

Information Technology Education, STKIP Taman Siswa Bima, Indonesia <sup>1,2,3</sup>

[adialfaisal28@gmail.com](mailto:adialfaisal28@gmail.com), [ombob999999999@gmail.com](mailto:ombob999999999@gmail.com), [nurfitrianiingsih984@gmail.com](mailto:nurfitrianiingsih984@gmail.com)

*\*Corresponding Author*

---

### **ABSTRACT**

Students with multiple academic activities often have difficulty organizing class schedules, which results in missed sessions, incorrect room information, and overlapping course entries. Existing student scheduling applications mainly provide schedule recording and reminders, and studies that combine automatic conflict validation with offline local storage remain limited. This study aims to design and evaluate an offline Android-based academic scheduling system with an automated schedule conflict detection mechanism and a class reminder mechanism. The system was developed using the Waterfall model, which consists of requirement analysis, system design, implementation, testing, and evaluation. It uses a three-layer architecture, Create, Read, Update, and Delete (CRUD) operations on a persistent SQLite database, a conflict validation query that compares the day and time range of a new entry with stored schedules, and reminders scheduled through Android's AlarmManager. Black-box functional testing with four scenarios showed that all scenarios passed: valid schedules were saved, invalid time input was rejected, overlapping schedules were blocked with a warning, and reminder alarms were registered in the Android system to trigger a local notification 15 minutes before class. These results indicate that the proposed system prevents scheduling input errors without requiring an internet connection. The system provides a practical basis for student time management, and future work should include user acceptance testing, cloud synchronization, and schedule export.

**Keywords:** Android Scheduling System, Offline Mobile Application, Schedule Conflict Detection, Automated Reminder Notification, SQLite Database.

### **1. Introduction**

The digitalization of education has changed how students access information and manage their academic activities (Harini et al., 2024; Mohamed Hashim et al., 2022; Timotheou et al., 2023). Among these activities, managing class schedules is a routine task that directly affects attendance and discipline. Students with several courses often find it difficult to keep track of class times, remember room locations, and avoid entering overlapping schedules, which can lead to missed or clashing sessions.

Similar problems are reflected in the literature: schedule clashes are treated as hard constraints in university course timetabling (Ge & Chen, 2022), and conventional reservation practices are known to cause double booking in other domains, which motivated the use of conflict-prevention mechanisms in a web-based booking system (Abdillah et al., 2026).

Several mobile applications have been proposed to support students' schedules and tasks. Android-based reminders for lecture schedules and assignments (Irawan & Utami, 2023), task reminders built with the Agile method (Dewi et al., 2024), and SQLite-based academic activity managers (Arsyanda et al., 2024) mainly provide recording, searching, and reminding, while a class scheduling application has also been developed with Extreme Programming (Wiratama & Santoso, 2023). Based on the information reported in these studies, the explicit combination of automatic overlap validation at the time of data entry, fully offline local storage, and class-time reminders in one student-oriented system is not their main focus. This is the research gap addressed by the present study.

This study aims to design and evaluate an offline Android-based academic scheduling system equipped with data management (CRUD) capabilities, instant keyword-based schedule searching on the local database, automated reminders, and schedule overlap prevention. The utilization of smartphones is considered highly efficient due to students' high mobility. The contributions of this study are: (1) a three-layer system architecture that separates the user interface, business logic, and local data storage; (2) a schedule conflict detection mechanism based on a same-day time-range overlap query that is executed before a record is saved; (3) a reminder mechanism based on Android's `AlarmManager` that issues a local notification 15 minutes before class; and (4) a functional evaluation of the system using black-box testing.

2. Literature Review

Related Work

Previous studies have developed mobile-based applications for student schedules and tasks. Irawan and Utami built an Android application that reminds students of lecture schedules and assignments (Irawan & Utami, 2023). Remindly is an Android task reminder developed with the Agile method that offers task addition, automatic reminders, and grouping of tasks by status, and it was tested with black-box and white-box methods (Dewi et al., 2024). StudySync stores academic tasks and notes in SQLite and offers reminders and a search feature, and it was developed in three Agile sprints (Arsyanda et al., 2024). A class scheduling mobile application for a private campus has also been developed with the Extreme Programming model (Wiratama & Santoso, 2023).

At the institutional level, course scheduling has been studied as a constraint problem. An ant colony approach, for example, aims to produce course schedules in which teacher, class, and time do not conflict (Ge & Chen, 2022). Web-based academic information systems manage schedules together with student, lecturer, attendance, and grade data (Halimi et al., 2026; Himmah & Laouar, 2025; Makkaraka et al., 2024; Prasetyo et al., 2024), and a web-based booking system uses double check locking to prevent double booking (Abdillah et al., 2026). These systems generally rely on server infrastructure and network access. Table 1 summarizes how the related studies differ from the present study.

Table 1. Comparison of related studies and the position of the present study

| Study                      | Platform / method           | Main focus                                                    | Difference from this study                                                            |
|----------------------------|-----------------------------|---------------------------------------------------------------|---------------------------------------------------------------------------------------|
| (Irawan & Utami, 2023)     | Android application         | Reminders for lecture schedules and assignments               | Reminder-centered; this study adds overlap validation before saving                   |
| (Dewi et al., 2024)        | Android; Agile              | Task reminders with login, task addition, and status grouping | Task-oriented; this study targets class schedules with time-range validation          |
| (Arsyanda et al., 2024)    | Mobile; SQLite; Agile       | Task management, notes, reminders, and search                 | General academic activity manager; this study focuses on class-time overlap detection |
| (Wiratama & Santoso, 2023) | Mobile; Extreme Programming | Class scheduling for a private campus                         | Applies XP; this study details the conflict rule and reminder mechanism               |

| Study                   | Platform / method    | Main focus                                                              | Difference from this study                                                                       |
|-------------------------|----------------------|-------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| (Ge & Chen, 2022)       | Ant colony algorithm | Course timetable generation under hard constraints                      | Institutional timetable generation; this study validates personal schedule entries on the device |
| (Halimi et al., 2026)   | Web; Agile           | Academic information system including schedules, attendance, and grades | Web-based and institution-wide; this study is a stand-alone offline mobile system                |
| (Abdillah et al., 2026) | Web; Agile           | Booking with double check locking to avoid double booking               | Different domain; confirms the value of clash prevention, applied here to class schedules        |

*Note: the comparison is based on the information reported in the abstract of each study.*

Based on this review, the present study is positioned as a stand-alone, offline mobile system that validates schedule overlaps at the moment of data entry and reminds students before class, rather than a general task manager or a server-based institutional scheduler.

### Theoretical Foundation

This section presents the theoretical foundation of the main components used in mobile system development:

- **Android & Android Studio:** Android is an open-source mobile operating system based on Linux developed by Google (Google, 2024a; Horton, 2019). Android was selected as the implementation platform because it is open source, is available on devices from many manufacturers and price ranges, and can be developed with free tools on common desktop operating systems, which suits an educational application aimed at students. In comparison, iOS development requires Apple's toolchain and hardware, and a web-based alternative needs network access to a server to work consistently. The development utilizes Android Studio as the official IDE, providing an intelligent editor and testing emulators (Android Studio Team, 2024; Diansyah & Syafrinal, 2025).
- **SQLite Database:** An embedded, relational local database management system integrated by default in Android for persistent offline data storage without requiring an external server (Arsyanda et al., 2024; SQLite Consortium, 2024).
- **RecyclerView:** An efficient user interface (UI) component that implements the *ViewHolder* pattern to load large-scale data with minimal memory consumption (Google, 2024a).
- **AlarmManager & Notifications:** An Android system service used to schedule background operations in the future. It triggers notifications for class reminders even when the application is inactive (Google, 2024a; Murphy, 2017).
- **Material Design:** A design language system by Google aimed at delivering modern, clean, and user-friendly visual elements (Google, 2024b).

### 3. Research Methods

This research adopts the Waterfall development model, which organizes development into sequential phases: requirement analysis, system design, implementation, testing, and evaluation (Saravanos & Curinga, 2023). The Waterfall model was chosen because the functional scope (CRUD, conflict detection, calendar view, and reminders) was defined before implementation, the system is a stand-alone offline application without server integration or external stakeholders, and the sequential documentation supports replication of the study. Iterative approaches such as Agile and Extreme Programming are better suited to changing requirements and continuous user

feedback (Halimi et al., 2026; Naufal & Furqan, 2026; Wandri et al., 2025; Zorzetti et al., 2022), which were not the focus of this study.

### System Requirements Analysis

The main functional requirements are detailed to ensure the system encompasses CRUD schedule operations, automatic detection of overlapping class times, weekly calendar visualization, keyword-based schedule searching on the local database, and reminder alarm scheduling. Non-functional requirements demand high responsiveness (under 2 seconds) and compatibility with a minimum of API 21 (Android 5.0).

### System Architecture

The system is designed using a multi-layered architecture that separates the user interface, business logic, and local data storage (Meier, 2018). Figure 1 shows the components of each layer and the data flow between them.

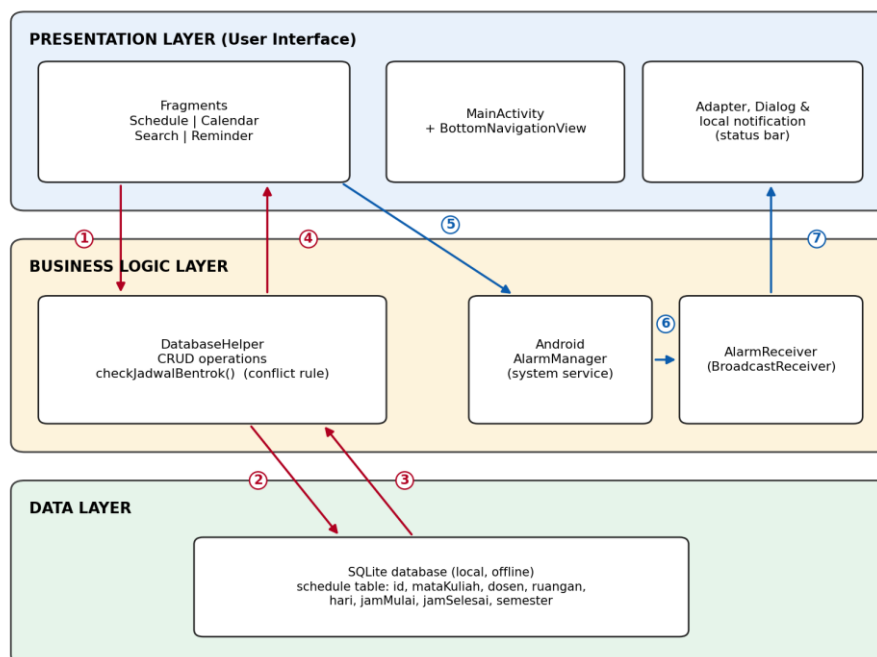


Figure 1. System architecture and data flow of the proposed scheduling system

1. *Presentation Layer*: Handles UI interactions through Activity, Fragment, Adapter, and Dialog.
2. *Business Logic Layer*: Processes data through DatabaseHelper to validate time conflict rules, and schedules reminders through AlarmManager and AlarmReceiver.
3. *Data Layer*: Utilizes SQLite for physical data persistence.

The data flow in Figure 1 is as follows. (1) The user enters or edits a schedule in a Fragment or Dialog, which passes the data to DatabaseHelper. (2) DatabaseHelper runs the conflict validation query on the SQLite database. (3) SQLite returns the matching rows as a Cursor. (4) DatabaseHelper returns the validation result to the Fragment, which either shows a conflict warning or saves the schedule. (5) When a schedule is selected and confirmed for a reminder, the Fragment registers an alarm with AlarmManager. (6) At the scheduled time AlarmManager triggers AlarmReceiver. (7) AlarmReceiver displays a local notification to the user 15 minutes before class.

### Schedule Conflict Detection Flow

The core logic for preventing overlapping schedules is implemented through relational validation of the class day and operational hours using the checkJadwalBentrok method. Each schedule is represented by the Jadwal entity shown in Figure 2, which stores the course name, lecturer, room, day, start time, end time, and semester. When a user attempts to save or update a schedule, the system queries the SQLite database for records on the same day and evaluates the time conditions described below.

```

public Jadwal(int id, String mataKuliah, String dosen, String ruangan,
              String hari, String jamMulai, String jamSelesai, String semester) {
    this.id = id;
    this.mataKuliah = mataKuliah;
    this.dosen = dosen;
    this.ruangan = ruangan;
    this.hari = hari;
    this.jamMulai = jamMulai;
    this.jamSelesai = jamSelesai;
    this.semester = semester;
}

```

Figure 2. The Jadwal entity used by the checkJadwalBentrok method

The checkJadwalBentrok function receives parameters for the class day, start time, and end time. The system evaluates two main overlapping conditions on the exact same day:

1. First, it detects if an existing class starts before the new input time and ends after the new class begins (the new schedule falls within an existing time slot).
2. Second, it detects if an existing class starts during the period between the new input's start time and end time (the existing schedule overlaps with the new input).

If the cursor object (Cursor cursor) finds one or more rows matching these conditional criteria (cursor.getCount() > 0), the method returns true, triggering a user interface warning that a conflict exists and preventing the database from executing the insertion. Otherwise, it returns false, allowing the transaction to proceed safely.

#### Testing Method

The system was evaluated with black-box testing, which checks whether the output of each function matches the expected output without inspecting the source code (Putri et al., 2024). Four scenarios were defined to cover the main functions: saving a valid schedule, rejecting an invalid time input, detecting a schedule conflict, and registering a reminder alarm. Each scenario was recorded with its data input, expected output, and result (Pass or Fail).

## 4. Results and Discussions

### Results

The application was successfully built using the Java programming language within a modular structure. MainActivity controls the navigation through a BottomNavigationView to direct users to four main fragments: Schedule, Calendar, Search, and Reminder. The DatabaseHelper component executes CRUD functions and validates input conflicts at the time the user submits a schedule. Lastly, AlarmReceiver acts as a broadcast receiver that triggers a local notification 15 minutes before the class begins.

### Functional Testing Results

Functional testing was conducted using black-box testing scenarios to ensure the validity of the system logic. The results are presented in Table 2.

Table 2. System functional testing results

| No | Test scenario          | Data input                            | Expected output                        | Result |
|----|------------------------|---------------------------------------|----------------------------------------|--------|
| 1  | Add a valid schedule   | All fields filled, slot is empty      | Schedule is saved to the list          | Pass   |
| 2  | Invalid time input     | Start time > End time (08:00 - 07:00) | Displays an invalid time error message | Pass   |
| 3  | Schedule conflict test | Input an already occupied day & time  | Displays a schedule conflict warning   | Pass   |
| 4  | Reminder alarm setup   | Schedule selected & confirmed         | Alarm registered in the Android system | Pass   |

## Discussion

The results indicate that the conflict rule implemented in checkJadwalBentrok behaves as specified in the tested scenarios. An entry with an invalid time range was rejected (scenario 2), and an entry that occupied an already used day and time slot triggered a conflict warning and was not inserted (scenario 3). Because the rule queries the stored records of the same day before insertion, the validation is applied at the moment of data entry, so overlapping entries do not reach the database. This is conceptually related to the treatment of time clashes as hard constraints in course timetabling (Ge & Chen, 2022) and to the prevention of double booking in web-based booking (Abdillah et al., 2026), but here it is applied on the device to the schedule of a single student and requires no server.

Storing schedules in SQLite allows the application to read and write data without an internet connection, which is consistent with the reasons for choosing SQLite in a comparable student activity manager (Arsyanda et al., 2024). Compared with web-based academic systems that depend on server infrastructure (Halimi et al., 2026; Himmah & Laouar, 2025; Makkaraka et al., 2024), the proposed system gives up institution-wide data sharing in exchange for availability and simplicity for individual students.

Scenario 4 confirms that the reminder alarm is registered in the Android system when a schedule is selected and confirmed, and the AlarmReceiver component is implemented to issue a local notification 15 minutes before class. Reminder applications for students have been reported to support time management (Dewi et al., 2024; Irawan & Utami, 2023), although the evidence on behavior is mixed: in a smartphone-based study, reminders raised the likelihood of studying after they were received, whereas the reminder group studied on fewer days overall than the control group (Nobbe et al., 2024). The effect of the reminders in this system on actual class attendance therefore needs to be measured rather than assumed.

This study has several limitations. First, testing was limited to four black-box scenarios; boundary cases (for example, a class that ends exactly when another begins), many simultaneous schedules, and device-specific alarm behavior were not reported. Second, the non-functional requirement of a response time under two seconds was defined in the requirement analysis but was not measured in the reported tests. Third, no user acceptance testing with students was conducted, whereas comparable studies combined black-box testing with user acceptance testing (Halimi et al., 2026; Naufal & Furqan, 2026). Finally, the system is a stand-alone offline application, so schedules cannot be shared or synchronized between devices.

## 5. Conclusion

This study designed and evaluated an offline Android-based academic scheduling system that combines CRUD operations on a local SQLite database, a schedule conflict detection rule, and reminders scheduled through AlarmManager. In the black-box functional tests, all four scenarios passed: valid schedules were saved, invalid time input was rejected, overlapping schedules were blocked with a warning, and reminder alarms were registered in the Android system, which shows that conflict validation and local storage work without an internet connection. The system offers students a practical tool for reducing scheduling input errors and supporting time management, and it provides a basis for further development. Future work should add user acceptance testing with students, measure response time and boundary cases, and extend the system with cloud synchronization and schedule export to PDF or Excel formats.

## References

- Abdillah, A., Nadiya, & Yaqin, M. A. (2026). Sistem pemesanan lapangan futsal berbasis web dengan otomatisasi pembayaran dan notifikasi real-time. *CODEX: Journal of Software Engineering*, 1(2), 48–54. <https://doi.org/10.33650/codex.v1i2.15367>
- Android Studio Team. (2024). *Android Studio user guide*. <https://developer.android.com/studio/intro>
- Arsyanda, Fakhri, M. M., Syukur, P. A., Jannah, D. M., Jannah, E. N., & Rauf, A. (2024). StudySync mobile application design for student academic activity management based on

- SQLite database. *Journal of Embedded Systems, Security and Intelligent Systems*, 5(3), 223–232. <https://doi.org/10.59562/jessi.v5i3.5042>
- Dewi, A. S., Syach, Z. A. A. M., & Nurpadila, A. (2024). Remindly: Android-based task reminder app with Agile methods to improve student study time efficiency. *Journal of Digital Technology and Computer Science*, 2(1), 70–99. <https://doi.org/10.61220/digitech.v2i1.20241>
- Diansyah, H., & Syafrinal, S. (2025). Design and development of a mobile application using Android Studio and Flutter. *Journal Mobile Technologies (JMS)*, 3(2), 69–76. <https://doi.org/10.59431/jms.v3i2.646>
- Ge, R., & Chen, J. (2022). Analysis of college course scheduling problem based on ant colony algorithm. *Computational Intelligence and Neuroscience*, 2022, 7918323. <https://doi.org/10.1155/2022/7918323>
- Google. (2024a). *Android developer documentation*. <https://developer.android.com/docs>
- Google. (2024b). *Material design guidelines*. <https://m3.material.io/>
- Halimi, A., Maulidiansyah, & Widiarti, D. (2026). Web-based academic information system development using Agile: A higher education case study. *CODEX: Journal of Software Engineering*, 1(2), 69–78. <https://doi.org/10.33650/codex.v1i2.15592>
- Harini, H., Ripki, A. J. H., Sulistianingsih, S., Herlina, H., & Putri, A. (2024). Digital transformation: The utilization of information and communication technology to enhance educational management efficiency in the modern era. *Jurnal Minfo Polgan*, 13(2), 1668–1674. <https://doi.org/10.33395/jmp.v13i2.14195>
- Himmah, F., & Laouar, M. S. (2025). Development of a web-based student registration system using Laravel framework. *JOKI: Journal of Computing and Informatics*, 2(2), 28–35. <https://doi.org/10.65678/joki.v2i2.218>
- Horton, J. (2019). *Android programming for beginners*. Packt Publishing.
- Irawan, A. D., & Utami, W. S. (2023). Aplikasi reminder jadwal kuliah dan tugas mahasiswa berbasis Android. *Jurnal Komputer dan Informatika*, 5(2), 288–300.
- Makkaraka, A. M. R. B., Iskandar, A., & Yang, W. (2024). Design of web-based student academic information system. *Ceddi Journal of Education*, 3(2), 9–15. <https://doi.org/10.56134/cje.v3i2.102>
- Meier, R. (2018). *Professional Android* (4th ed.). Wrox Press.
- Mohamed Hashim, M. A., Tlemsani, I., & Matthews, R. (2022). Higher education strategy in digital transformation. *Education and Information Technologies*, 27(3), 3171–3195.
- Murphy, M. L. (2017). *The busy coder's guide to Android development*. CommonsWare.
- Naufal, M., & Furqan, M. (2026). A mobile-based alumni relationship management system using Extreme Programming: Enhancing digital engagement in Islamic boarding school education. *CODEX: Journal of Software Engineering*, 1(1), 39–47. <https://doi.org/10.33650/codex.v1i01.14456>
- Nobbe, L., Breitwieser, J., Biedermann, D., & Brod, G. (2024). Smartphone-based study reminders can be a double-edged sword. *npj Science of Learning*. <https://doi.org/10.1038/s41539-024-00253-7>
- Prasetyo, D., Utami, A., & Laksana, T. G. (2024). Website based academic information system design using Extreme Programming method. *Journal of Informatics Information System Software Engineering and Applications (INISTA)*, 6(2), 134–143. <https://doi.org/10.20895/inista.v6i2.1214>
- Putri, S. J., Putri, D. G. P., & Putra, W. H. N. (2024). Analisis komparasi pada teknik black box testing (studi kasus: Website Lars). *Journal of Internet and Software Engineering*, 5(1), 23–28. <https://doi.org/10.22146/jise.v5i1.9446>
- Saravanos, A., & Curinga, M. X. (2023). Simulating the software development lifecycle: The Waterfall model. *Applied System Innovation*, 6(6), 108. <https://doi.org/10.3390/asi6060108>

- SQLite Consortium. (2024). *SQLite documentation*. <https://www.sqlite.org/docs.html>
- Timotheou, S., Miliou, O., Dimitriadis, Y., Sobrino, S. V., Giannoutsou, N., Cachia, R., Monés, A. M., & Ioannou, A. (2023). Impacts of digital technologies on education and factors influencing schools' digital capacity and transformation: A literature review. *Education and Information Technologies*, 28(6). <https://doi.org/10.1007/s10639-022-11431-8>
- Wandri, R., Fadhilah, M. R., Setiawan, P. R., & Fadhillah, M. (2025). Agile Scrum as a development approach: A case study of web-based school information system design. *Sistemasi: Jurnal Sistem Informasi*, 14(4), 1722–1735.
- Wiratama, J., & Santoso, H. (2023). Developing a class scheduling mobile application for private campus in Tangerang with the Extreme Programming (XP) model. *G-Tech: Jurnal Teknologi Terapan*, 7(2), 484–493. <https://doi.org/10.33379/gtech.v7i2.2288>
- Zorzetti, M., Signoretti, I., Salerno, L., Marczak, S., & Bastos, R. (2022). Improving agile software development using user-centered design and lean startup. *Information and Software Technology*, 141, 106718.