

Spec-Driven Development of a Role-Based Attendance Request System with Dynamic QR Verification

1st Marija Apostoloska-Kondoska

Faculty of Information and Communication Technologies –
Bitola
University “St. Kliment Ohridski” - Bitola
Bitola, Macedonia
marija.kondoska@uklo.edu.mk

2nd Ilija Jolevski

Faculty of Information and Communication Technologies –
Bitola
University “St. Kliment Ohridski” - Bitola
Bitola, Macedonia
ilija.jolevski@uklo.edu.mk

Abstract— This paper presents a web-based system for attendance tracking and management in a higher education institution, featuring three roles (student, professor, admin) and three role-based views. Beyond the application itself, the paper highlights the practical value of Spec-Driven Development (SDD) today: when a specification is complete and of high quality, it becomes an actionable contract that enables effective AI-assisted programming (via LLM-based assistants and agentic workflows) while preserving control through verification against the spec. In this work, the system is produced through a spec-driven workflow using the spec-kit toolkit as the primary development artifact, with implementation and tests generated via an AI assistant based on the written specification (i.e., without manual coding of application logic). We outline a process that (1) plans and documents the specification with an explicit scope, role model, use cases, constraints, and acceptance criteria; (2) uses the specification as input for AI/agent-assisted implementation and test generation; and (3) supports verification through traceability from requirements to code and tests. The proposed solution uses a dynamic QR code for real-time verification, workflow support for attendance and excused-absence requests, notifications, and Excel export. The system is implemented using Next.js (frontend), Node.js (backend), and SQLite, and the paper discusses practical implication and limitation of applying a spec-driven verification-oriented workflow in an academic attendance management context. This work is presented as a case study demonstrating the practical application of specification-driven development combined with AI-assisted implementation in an academic student attendance system.

Keywords—*Spec-Driven Development (SDD), AI-assisted programming, LLM agents, web-based attendance system, role-based access control, dynamic QR code.*

I. INTRODUCTION

The development of information technology has created opportunities for educational institutions to integrate digital solutions into daily operations, including web-based student attendance systems [1]. Student attendance in higher education affects academic performance [2], yet many institutions still rely on instructors to manually record attendance [3]. This is time-consuming, error-prone, and particularly challenging in large classes [3]. Automated attendance systems can improve accuracy, transparency, and processing efficiency by reducing manual intervention and enabling real-time data handling.

Manual attendance tracking also carries risks such as lost records and limited auditability [3]. In practice, collecting handwritten signatures can distract from teaching and may be difficult to manage reliably in large cohorts [3]. Since missed classes are associated with poorer performance, robust attendance monitoring remains important for supporting student outcomes [2].

In parallel, recent advances in artificial intelligence (AI), especially large language models (LLMs), have enabled tools that can generate and modify source code from natural-language prompts [4]. Examples include OpenAI’s GPT-based assistants and GitHub Copilot, which is integrated into development environments to support code completion and generation. In this paper, we refer to these LLM-powered tools as AI assistants [5].

With the rapid development of mobile and web technologies, several digital solutions have been proposed for attendance recording, including RFID, NFC, and QR codes [6]. QR-code systems are particularly attractive due to their low cost, ease of deployment, and compatibility with mobile devices and web platforms [7]. In such systems, students confirm attendance by scanning a code generated by the instructor, which speeds up processing and reduces administrative effort [8]. Dynamically generated or time-limited QR codes further reduce misuse by adding time validation and centralized verification [9].

Beyond the technology choice, the development methodology strongly influences correctness and maintainability [10]. Specification-driven development emphasizes defining and validating requirements before implementation [11]. In today’s tooling landscape, a complete and well-structured specification can also serve as a high-fidelity input for AI assistants and agentic workflows, enabling faster implementation while maintaining control through verification against the specification [11].

In this study, the specification (created with spec-kit) is considered as the primary artifact that guides the development and testing with the help of artificial intelligence. The structured elements of the specification (scenarios, constraints, and acceptance criteria) serve as machine-processable inputs, while traceability allows the connection of requirements to implementation and tests.

The paper presents a web-based attendance system with role-based access control, dynamic QR verification, and request workflows, including a threat model and baseline evaluation. The aim is to demonstrate that the specification-based approach enables a structured, reliable, and auditable system for academic environments.

This study is a case study that explores the practical feasibility of implementing specification-driven development in combination with AI-assisted programming. The aim of the study is to provide an in-depth engineering perspective on how structured specifications can guide the implementation and verification of AI-generated software in academia.

A. Research Questions

To clearly define the research contribution of this study, the following research questions are formulated:

RQ1: Does specification-driven development with artificial intelligence (SDD+AI) reduce implementation time compared to the basic development workflow under identical requirements?

RQ2: Does SDD+AI change the state of traceability from requirement to testing?

RQ3: What types of defects occur in AI-generated code and how are they detected during verification?

RQ4: Does specification-driven AI-assisted development reduce the number of development iterations (PR/prompt cycles) compared to a baseline workflow?

RQ5: Does specification-driven AI-assisted development support correct implementation of security-critical constraints, such as QR token validity and RBAC enforcement?

II. RELATED WORK

Automated attendance management systems have received significant attention as educational institutions transition from manual record-keeping to scalable digital solutions [6]. Early approaches employed technologies such as RFID and NFC, which improved accuracy and reduced administrative overhead but typically required dedicated hardware infrastructure and added maintenance costs [12]. With the widespread adoption of smartphones, QR-code-based attendance systems have emerged as a flexible and low-cost alternative, enabling students to confirm presence through mobile scanning while supporting real-time data synchronization [8]. Prior work also suggests that dynamic or time-limited QR tokens can reduce misuse and improve reliability compared to static codes [9].

In parallel, secure access management remains a core aspect of academic information systems. Role-Based Access Control (RBAC) is widely used to enforce separation of privileges among students, instructors, and administrative personnel, simplifying governance and improving maintainability [13]. From a software engineering perspective, specification-driven methodologies emphasize defining requirements and system interfaces before implementation, which supports traceability and verification in web-based applications.

III. SPECIFICATION-DRIVEN DEVELOPMENT METHODOLOGY

This study applies a specification-driven development (SDD) methodology to guide system design and implementation. The specification is treated as a development artifact that evolves through processes and is refined

throughout development to maintain alignment between requirements and implementation.

The methodology consists of four phases: Constitution, Specify, Plan, and Tasks, ensuring a structured transition from requirements to implementation.

A. Constitution

The constitution phase defines fundamental principles such as architectural consistency and structural separation of components to guide AI-assisted code generation.

B. Specify

The Specify phase defines system functionality, constraints, and acceptance criteria independently of implementation details. Specification artifacts serve both as design documentation and as input for AI-assisted code generation.

- *Specification Structuring Toolkit Integration*

The specification toolkit was used to organize functional scenarios, constraints, and acceptance criteria into modular specification segments. These structured artifacts improved accuracy and provided high-quality input for AI-assisted development.

C. Plan

Planning, scoping and defining the QR token rotation time, database modeling strategy using institutional identifiers and workflow decomposition aligned with specification modules, so that planning is not a process where we have a static treatment of the plan with a predefined timeframe, but rather as a continuous and adaptable process, which is revised in accordance with the refinement and expansion of the specification segments.

D. Tasks

The Tasks phase translates specification elements into UI components, API endpoints, and database entities using AI-assisted development. GitHub Copilot was used with structured specification context to generate implementation components, with human validation ensuring compliance with requirements. These phases demonstrate how SDD bridges system design and AI-assisted implementation, with specification artifacts serving as both process guides and machine-interpretable inputs.

“ Fig. 1” show the Spec-kit workflow.



Fig. 1. Spec-Kit Workflow

E. Specification artifact example and traceability matrix

In order to illustrate the proposed workflow, this section presents a compact example of a specification artifact and an accompanying traceability matrix. The intention is to demonstrate how a high-quality and verified specification can function as an enforceable contract for AI-assisted implementation, while simultaneously preserving verification and engineering control mechanisms.

Artifact ID: ATT-SCAN-01**Scenario: Student attendance registration via dynamic QR code**

TABLE I. MODULAR SPECIFICATION DERIVED USING STRUCTURED SPECIFICATION METHODOLOGY

Category	Description
Actors	Student, Attendance Service
Trigger	Student scans QR token during active session
Preconditions	Student authenticated; session active; student assigned to course group
Functional Behaviour	System validates token time window, verifies group membership, records attendance
Constraints	Token validity interval ≤ 3 seconds; one valid scan per session
Non-Functional Requirements	Response latency $< 1s$; secure token handling
Acceptance Criteria	Valid scan stored; invalid scan rejected; event logged
Output	Attendance record; notification event

TABLE II. REPRESENTATIVE TRACEABILITY RELATIONSHIPS

Requirement ID	Specification Artifact	Implementation Component	Test Scenario
FR4	ATT-SCAN-01	QR Validation Service	Token expiry test
FR4	ATT-SCAN-01	Attendance API Endpoint	Valid scan workflow
FR2	AUTH-RBAC-02	Middleware Authorization	Unauthorized access test
FR5	REQ-HANDLE-03	Request Processing Module	Approval flow test
NFR1	SEC-01	Password Hashing Module	Security validation
NFR4	PERF-02	QR Validation Routine	Latency measurement

F. Example Specification Artifact

The following artifact represents a modular specification block derived using the structured specification tools shown in Modular Specification Derived Using Structured Specification Methodology I. It captures the intent of the system in a format that is both machine-interpretable and human-understandable, thus providing an appropriate basis for guiding implementation.

G. Traceability Matrix

Traceability mapping ensures alignment between requirements, implementation components, and validation activities. Representative Traceability Relationships II demonstrates representative traceability relationships. This mapping demonstrates how specification artifacts serve as a central linkage point between conceptual requirements and system realization. In the context of AI-assisted development, traceability provides a mechanism for verifying that generated components remain aligned with intended system behaviour.

Collectively, the artifact and traceability matrix illustrate the methodological contribution of this study, structured specification segments not only enable implementation, but also enable systematic validation and reproducibility within AI-supported software engineering workflows.

H. QR attendance threat model and mitigation analysis

QR-based attendance systems are exposed to a set of common threats that can undermine integrity (false attendance), availability (service disruption), and privacy. The following threat model shown in Threat Model and Mitigation Strategies for QR-Based Attendance ValidationIII summarizes the main attack scenarios considered in this work and the corresponding mitigations implemented in the proposed design.

Mitigation Definitions

- M1 — Time-bounded rotation: Tokens rotate every 2–3 seconds and expire outside validity windows.
- M2 — Server-side protection: Tokens are generated, signed, and validated server-side to prevent tampering.
- M3 — Context binding: Tokens are bound to session and course/group identifiers.
- M4 — RBAC enforcement: Session lifecycle restricted to professors; scans accepted only during ACTIVE sessions.
- M5 — Rate limiting: Request throttling and anomaly logging at validation endpoints.
- M6 — Data minimization: Tokens exclude student identifiers; PII remains server-protected.

Residual risks and limitations

While token rotation and server-side validation reduce replay and forging, proxy attendance (T2) cannot be fully eliminated without stronger liveness or presence signals (e.g., geolocation, Bluetooth proximity or device binding). These are considered in the Future work section as optional validation mechanisms, as they may increase deployment complexity and raise additional privacy concerns.

IV. SYSTEM SPECIFICATION AND ARCHITECTURE

The student attendance management system support three roles – student, professor and administrator, with role-specific interfaces, permissions and views. Functional requirements include authentications, course management, QR code-based attendance registration, request processing, notifications and reporting. Non-functional requirements address security, usability, maintainability, performance, and portability. Constraints ensure single active sessions per group, valid token intervals, sign-in-based participation, timestamped request and enforced access privileges. Formal models, API contract and database constraints represent the foundations of system specification.

TABLE III. THREAT MODEL AND MITIGATION STRATEGIES FOR QR-BASED ATTENDANCE VALIDATION

Threat ID	Threat Description	Potential Impact	Mitigation(s) in Proposed System
T1	Token replay or reuse outside intended time window	Unauthorized attendance registration	M1: Time-bounded rotating tokens rejected outside validity window
T2	Token sharing enabling proxy attendance	False presence records	M1: Rapid token rotation; M4: RBAC session control limiting scan context
T3	Forged token submission	Fabricated attendance entries	M2: Server-side generation, validation, and signing of tokens
T4	Session confusion between groups/sessions	Incorrect attendance attribution	M3: Context binding to session/group; M4: RBAC session control
T5	Denial of service via excessive scans	Service degradation	M5: Rate limiting and abuse detection
T6	Privacy leakage from tokens or logs	Exposure of personal data	M6: Data minimization and role-restricted access

The proposed student attendance management system follows a layered, role-based web architecture implemented with:

A. Presentation layer (Next.js)

Provides separate role-based views and navigation for students, professors, and administrators. After authentication, the UI renders features according to the user's role (RBAC-controlled). The student view focuses on scanning the QR code to record course attendance and requests for excused absence; the professor view supports session control, QR code generation, and approval/rejection of incoming requests; the admin view supports master data management (courses, semesters, staff, users, attendance requests, exemption requests, notifications, settings, etc.)

B. Application Layer (Node.js server logic)

- Authentication and password-change flows (including mandatory change on first login)
- Authorization checks (RBAC middleware/guards)
- Attendance session management (start/stop, active-session checks)
- Dynamic QR token generation and validation (rotation every 2–3 seconds)
- Request processing (attendance requests and exemption requests)
- Notification creation and delivery
- Excel import/export reports

C. Data Layer (SQLite)

SQLite persists all entities and relationships, including users, students (identified by student ID number), courses and groups, group membership, staff assignments, attendance sessions, QR code scans, various types of requests, and notifications. Constraints (UNIQUE/CHECK/FK) and indexes enforce core invariants (e.g., at most one active session per group, unique membership per group).

We chose SQLite as a lightweight data storage solution suitable for prototyping and controlled academic environments. While it reduces the complexity of the system, it is not intended for large-scale use. In real-world deployments, the system would migrate to a scalable database solution (e.g., PostgreSQL or cloud-based systems) without affecting the underlying specification-driven methodology.

“Fig. 2” show the architecture of student attendance management system.

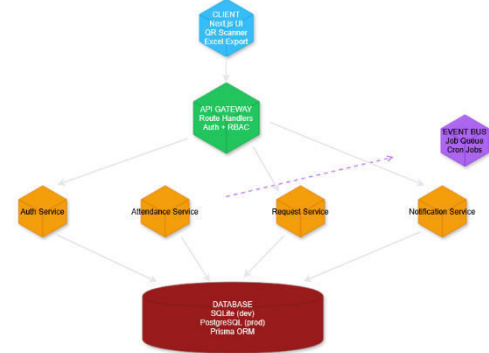


Fig. 2. Attendance management system architecture

V. DATA MODEL AND IMPLEMENTATION

The database schema defines core entities including users, students, staff, courses, groups, sessions, tokens, attendance records, requests, and notifications. Relationships and constraints ensure data consistency and enforce system invariants.

“Fig. 3” show the database schema of student attendance management system

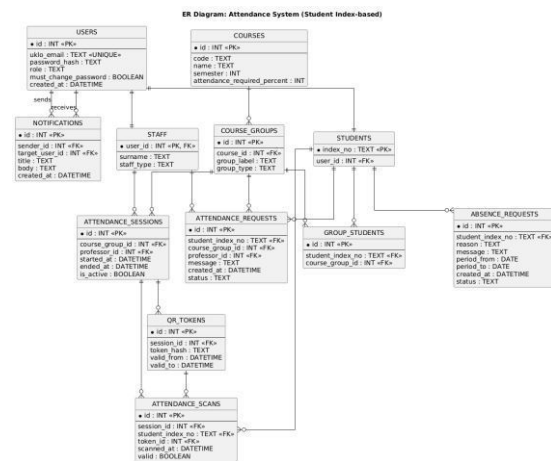


Fig. 3. Database schema of student attendance management system

The implementation aligns system components with the defined specification to maintain traceability between requirements, design, and system behavior. Authentication is performed using institutional email credentials with securely stored password hashes, while authorization is enforced through role-based access control (RBAC).

“Fig. 4” show Attendance management system - Login page

Fig. 4. Attendance management system - Login page

The system implements a session-based attendance workflow. Professors initiate attendance sessions, triggering the generation of time-limited QR tokens that rotate every 2–3 seconds. Students scan the QR code to register attendance, after which the record is validated and stored.

“Fig. 5” show Attendance management system – Professor page.

Fig. 5. Attendance management system -Professor page

The attendance registration endpoint (POST /api/attendance/scan) validates the QR token expiration, checks the active session and user membership, and returns either a successful attendance record or a rejection with a validation reason.

This ensures that specification constraints, such as token validity and role-based access, are enforced at the application level.

The system also supports workflows for attendance correction and exemption requests, which are processed through approval mechanisms. Notifications are generated based on user actions and system events. Reporting functionality enables generation and export of attendance data.

VI. EVALUATION AND DISCUSSION

To make a good assessment of the functional soundness of the system and the methodological impact of specification-driven AI-assisted development, a light comparative evaluation was conducted.

The evaluation compares the observed SDD+AI workflow used in this project with a reconstructed baseline workflow with manual coding, which represents a conventional implementation without the assistance of AI, but maintaining the same requirements and acceptance criteria. The basic workflow represents conventional development driven by the same specification, identical

requirements and criteria, but without the use of AI tools for code generation. Evaluation metrics were recorded during the implementation and verification phases in order to obtain real-world comparison results between the two workflows.

In order to present an objective comparison between SDD+AI and the baseline development workflow, the analyses are performed according to the following evaluation metrics:

A. Results and interpretation

From the data shown in **Error! Reference source not found.IV**, the SDD+AI workflow required less implementation time, fewer iterations, and fewer acceptance defects compared to the baseline, while also achieving significantly higher traceability coverage. These observations show that the artifacts from the structured specification improved the efficiency and effectiveness of development and the alignment between requirements, implementation, and testing. These results indicate that a well-structured specification provided high-quality semantic context for AI assistants, improving the accuracy of the initially generated code and reducing correction cycles. Also an important segment here is verification, which is necessary, especially for security such as QR token validation and RBAC implementation.

TABLE IV. IMPLEMENTATION-TIME COMPARISON BETWEEN BASELINE WORKFLOW AND SDD+AI

Metric	Baseline	SSD+AI
Total implementation time (person-hours)	52	34
Iterations (PR/prompt cycles)	18	11
Failed test runs before first full pass	9	4
Defects in acceptance testing	7	3
Traceability coverage (%)	61	89

B. Defect observations.

The defects discovered in the AI-generated components belonged to:

- errors in boundary conditions (QR token expiration window),
- lack of authorization guards,
- incorrect handling of database constraints.

All defects were detected through specification-derived tests, confirming that the specification functioned as an effective verification contract.

C. Security validation

Testing was performed for abuse of the generated QR code, such as attempted replays, sending expired tokens, and unauthorized session scanning. All tested scenarios were correctly rejected by the implemented validation logic.

The comparative results in Table IV address RQ1 and RQ4 by indicating a reduced implementation effort and a smaller number of development iterations in the SDD+ AI workflow compared to the baseline method.

The significantly higher level of traceability coverage observed in SDD+AI addresses RQ2, while RQ3 addresses the identified types of defects and their detection through tests derived from the specification.

Finally, the successful validation of QR token constraints and RBAC enforcement through abuse-oriented tests provides relevant evidence for RQ5, indicating that specification-driven development with AI support can enable the correct implementation of security-critical logic.

D. Limitations

The evaluation was conducted based on a single academic implementation scenario and should be interpreted as an engineering case study rather than a controlled experiment. Larger scale studies are needed to generalize quantitative improvements.

Although the focus of this study is on the development methodology, code quality and maintainability remain important factors. The generated code followed the structural constraints defined in the specification, which improved consistency across modules. However, the long-term maintainability, readability, and extensibility of AI-generated code have not been systematically assessed and represent an important direction for future work.

E. Scope of Testing

The testing activities in this study focused on functional correctness, integration validation, and security-related scenarios derived from the specification and threat model. As such, no extensive performance and security testing was conducted.

VII. CONCLUSION

This paper presented the design and development of a web-based student attendance management system built upon a specification-driven development approach. The system integrates role-based access control (student-professor-admin), QR code-based dynamic attendance verification, request processing workflows, and creating report tailored for academic environments.

The implemented architecture, which combines Next.js, Node.js, and SQLite, proved to be suitable for constructing a lightweight yet functional system capable of supporting multi-functional interaction and recording student attendance in the course in real time. The evaluation results showed that the system meets the defined functional requirements and preserves data integrity in the tested scenarios. Additionally, dynamic QR code generation reduces administrative effort, replacing mechanical attendance recording with automatic creation of student attendance reports per specific course.

It is important to note that the paper, in addition to the technical implementation, also highlights the practical value of specification-driven development as a mechanism that enables programming assisted by artificial intelligence. When requirements, constraints, and acceptance criteria are clearly and precisely formulated, AI assistants can be used more efficiently and consistently, aiming for rapid system development based on implementation accuracy and compliance.

Overall, the proposed system demonstrates both the feasibility of modern web technologies in addressing

academic administrative needs and the methodological advantages of specification-focused development over ad-hoc “vibe programming”.

VIII. FUTURE WORK

Future work should focus on improving system scalability and deployment, including migration to distributed databases and cloud-based environments. Additionally, further research is needed to systematically evaluate AI-assisted development using metrics such as development time, iterations, defect density, and traceability, as well as to compare different AI development frameworks (e.g., GitHub Copilot vs. Claude Code) in terms of efficiency and code quality.

REFERENCES

- [1] Ziky, Z., Yusma, Y., & Rizal, C. (2025). Design of New Student Registration System Using Web-Based Waterfall Method. *SAINTEKBU*, 17(01), 36-48.
- [2] Credé, M., Roch, S. G., & Kieszczyńska, U. M. (2010). Class Attendance in College: A Meta-Analytic Review of the Relationship of Class Attendance With Grades and Student Characteristics: A Meta-Analytic Review of the Relationship of Class Attendance With Grades and Student Characteristics. *Review of Educational Research*, 80(2), 272-295. <https://doi.org/10.3102/0034654310362998> (Original work published 2010)
- [3] Kassim, M., Mazlan, H., Zaini, N., & Salleh, M. K. (2012, July). Web-based student attendance system using RFID technology. In *2012 IEEE Control and System Graduate Research Colloquium* (pp. 213-218). IEEE.
- [4] Chen, M. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- [5] Klemmer, J. H., Horstmann, S. A., Patnaik, N., Ludden, C., Burton Jr, C., Powers, C., ... & Fahl, S. (2024, December). Using ai assistants in software development: A qualitative study on security practices and concerns. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (pp. 2726-2740).
- [6] RASHID, A. M. (2023). SMART CAMPUS: A REVIEW ON SMART ATTENDANCE SYSTEMS AS AN EFFICIENT APPROACH. *Journal of Engineering & Technological Advances*, 8(2), 16-24.
- [7] Talip, B. A., & Zulkifli, M. Z. (2018). Mobile attendance system using QR codes technology. *Journal of Computing Technologies and Creative Content*, 3(1), 1-3.
- [8] Liew, K. J., & Tan, T. H. (2021, September). QR code-based student attendance system. In *2021 2nd Asia Conference on Computers and Communications (ACCC)* (pp. 10-14). IEEE.
- [9] Gupta, S., Bhagat, H., Uphale, R., Patel, S., & Mankar, D. (2025). QR Code Based Attendance System. *International Journal for Research in Applied Science and Engineering Technology*, 13(4), 2576-2582.
- [10] Tian, F., Wang, T., Liang, P., Wang, C., Khan, A. A., & Babar, M. A. (2021). The impact of traceability on software maintenance and evolution: A mapping study. *Journal of Software: Evolution and Process*, 33(10), e2374.
- [11] What is Spec-Driven Development? [Online] <https://developer.microsoft.com/blog/spec-driven-development-spec-kit>, Accessed:25.12.2025
- [12] Ishaq, K., & Bibi, S. (2023). IoT based smart attendance system using RFID: A systematic literature review. *arXiv preprint arXiv:2308.02591*.
- [13] Sandhu, R. S. (1998). Role-based access control. In *Advances in computers* (Vol. 46, pp. 237-286). Elsevier.
- [14] Create the spec [Online], <https://github.com/github/spec-kit>, Accessed:25.12.2025