UNI-DISC: A Digital Loyalty Card Management System

[1]Diosdado Asumu Ndong Andeme , [2]Jonah Vincent Joshua, [3]Flora Bula Afang
[123]Department of Informatic and Technology
[123]Afro-American University of Central Africa, Djibloho, Equatoria Guinea

Abstract

In Equatorial Guinea, where students constitute over 50% of the population, limited access to targeted commercial discount programs exacerbates financial burdens amidst rising inflation. This study presents the design, development, and implementation of UNI-DISC, a comprehensive loyalty card management system developed to address this gap. The system facilitates collaboration between educational institutions, students, and local businesses to provide exclusive discounts on essential goods and services without minimum purchase requirements. Adopting an incremental software development lifecycle and Extreme Programming (XP) methodology, the project utilized Unified Modeling Language (UML) for requirements specification and system design. The technical architecture employs Java with Spring Boot for the backend RESTful API, TypeScript with Angular for the frontend single-page application, and MySQL as the relational database management system. Security implementations include JWT-based authentication, Spring Security framework, and BCrypt password encryption. The resulting desktop application enables administrators to manage student cards, educational institutions, and financial transactions while providing financial officers with revenue tracking capabilities. The system successfully meets functional requirements including user authentication, card lifecycle management, institution administration, and financial reporting. This initiative represents a novel approach to student support in Equatorial Guinea's unique socioeconomic context, offering scalable, secure, and maintainable technology infrastructure that benefits both students and small-to-medium enterprises. Future enhancements should focus on integrating mobile applications, advanced financial analytics, and broader commercial partnerships to maximize socioeconomic impact.

Keywords: Loyalty card management, student discounts, Extreme Programming, Spring Boot, Angular, Equatorial Guinea, software engineering

1. Introduction

The current business landscape is highly competitive, forcing SMEs to innovate in customer retention and acquisition techniques. Loyalty programs are extremely efficient at rewarding regular customers and developing new market segments [1]. These programmes stimulate repeat business by delivering tangible prizes, special discounts, and privileged access, boosting consumer satisfaction and corporate reputation through positive word-of-mouth [2]. Such programs can transform market access for under-represented demographic groups in developing nations.

Equatoria Guinea (EG) makes a strong argument for targeted student discounts. The nation's population was 1,679,172 in the 2022 World Factbook, with 38.73% under 14, 57.35% 15-65, and 3.92% above 65 [3]. This demographic structure shows that 50% of the population is school-aged, technical trainees, or young adults entering the workforce, a large and strategic consumer base. Despite their numerical value and potential relevance to national development, this demographic group lacks economic-specific commercial efforts.

To address this systemic gap, the University Discounts (UNI-DISC) project created a student loyalty card system in conjunction with local companies. UNI-DISC offers exclusive discounts to all registered students, unlike competitor loyalty programs like EGTC and Martínez Hermanos that demand minimum purchase requirements. A mobile platform lets students communicate directly with partner firms, and a PC administration module manages system oversight, cards, and finances. The desktop application module centralises administrative tasks and is the authoritative source for student verification, card issuance, and financial transaction administration.

2. Statement of the Problem

Despite making up over half of Equatorial Guinea's population, students face severe impediments obtaining economic benefits customised to their requirements. Though well-intentioned, current commercial attempts in the country fail to address student finances. Major shops' promotional practices require minimum purchases to qualify for discounts, excluding most students without independent income or solid financial support. This excludes young people from saving for food, transit, and education.
Economic conditions are deteriorating. National inflation hit 3% in January 2025, disproportionately affecting

student essentials [4]. Student families with low means are further burdened by rising food, transportation, and school supply prices. Without targeted commercial support, inflationary pressure and the inability to finance minimum purchases prevent students from receiving discounts, continuing economic marginalisation. No formal organisation in Equatorial Guinea offers structured discount programs for students' daily consumption needs. The institutional gap is both a problem and an opportunity. Businesses cannot properly authenticate student status, and students cannot quickly identify or use advantages without centralised student verification mechanisms. Thus, the solution must address three interrelated issues: (1) the lack of a national student verification infrastructure, (2) the lack of company incentives to offer discounts, and (3) the need for a sustainable approach that serves all stakeholders without financially burdening the most disadvantaged.

3. Aim and Objectives

This project will develop a loyalty card management system with companies that sell goods and services to give students exclusive discounts and boost customer loyalty and market reach.

The Specific Objectives include

i. implementing a secure authentication mechanism for authorised administrators and finance officers to access system functions through role-based access control.

ii. Create a card management submodule for administrators to issue cards to verified students, maintain inventory, and handle card lifecycle events like cancellation and reactivation.

iii. Create an educational institution administration framework for universities, technical training centres, institutes, and primary schools, complete with registration, visualisation, editing, and search capabilities for their student populations.

iv. Create a financial transaction management system to track card sales and reactivations, process purchases with automatic discounts, and generate thorough reports for administrative control.

v. Implement multi-layered system security measures such as password encryption, token-based authentication, and secure API protocols.

vi. Develop a user-friendly interface with responsive design concepts for users of diverse technological proficiency.

4. Literature Review

4.1 Loyalty Programs and Strategic Alliances

Supermarket loyalty programs have been shown to increase customer retention, profitability, and business growth [5]. These activities enable delighted customers to become brand ambassadors, boosting reputation and organic growth. Strategic collaborations with complementary enterprises give resource-constrained SMEs access to new markets and pooled operational resources [6]. Small businesses with limited funding need such partnerships to pool resources and better manage environmental risks. The digital transformation of loyalty programs has increased their influence. Mobile apps that administer digital loyalty cards let SMEs offer personalised rewards and collect behavioural data for future marketing [7]. Research in Galicia shows that technology investments in digital loyalty systems have boosted regional ICT sector growth, suggesting that Equatorial Guinea could benefit from similar technology-commerce initiatives.

4.2 Student-Specific Loyalty Initiatives

Discount programs for students create unique value propositions that go beyond cost reductions, fostering a sense of belonging and institutional commitment among students, which may reduce dropout rates and strengthen relationships between educational institutions and the commercial sector. The psychological aspect of student loyalty programs is especially important in emerging nations when budgetary pressures threaten educational continuity.
Global student discount aggregators like Student Beans and UNiDAYS have succeeded. These services validate student eligibility through institutional email validation, direct interaction with academic portals, or manual document check to offer unique discounts from many partner brands [9]. Student Beans provides unique discount codes via mobile apps and browser extensions after verification, while UNiDAYS verifies digitally without a card. Both systems offer free enrolment and fast access to benefits, but they acknowledge regional boundaries and academic credentials [10]. The International Student Identity Card (ISIC) offers over 150,000 savings in 130+ countries through UNESCO-endorsed verification [11]. ISIC demands student status verification and keeps it for five years in physical and digital formats. Its international recognition is a benefit, but the costs and complicated application process may limit access for students in developing nations [12].

4.3 Software Development Methodologies

Traditional software development methods, called heavyweight approaches, emphasise upfront planning and stringent change control [13]. These methods are unsuitable for dynamic stakeholder contexts because they presume all requirements can be provided at project start. Agile methods overcome these limits through incremental development, client collaboration, and flexible planning. Extreme Programming (XP) is a popular agile paradigm that places adaptability before predictability [14].

Communication, simplicity, feedback, and courage—XP's fundamental values—enable development teams to adapt to changing requirements. The methodology's twelve practices—pair programming, continuous integration, test-driven development, and on-site client representation—enforce these principles through technical and social means [15]. For academic projects with limited resources and changing needs, XP's simplicity and frequent feedback are beneficial.
Scrum uses time-boxed sprints with daily stand-ups and iterative reviews; Kanban visualises workflow and limits work-in-progress to optimise flow; and RUP provides a configurable framework adaptable to organisational contexts [16]. Comparative study shows that XP's focus on technical techniques and developer competence makes it excellent for small-team projects that prioritise code quality and adaptability.

4.4 Technology Stack Considerations

Modern web application development involves careful frontend and backend technology choices. Due to its object-oriented architecture, platform independence through the Java Virtual Machine (JVM), and multithreading features, Java is a top corporate language [17]. Spring Boot framework's convention-over-configuration development allows speedy microservices-based architecture building with little setup [18]. Spring Boot's ecosystem and modular architecture make it ideal for complicated backend development that needs great scalability and security. Single-page application (SPA) frameworks have changed frontend development. Angular, a TypeScript-based MVC framework, is suited for large-scale applications with complex user interactions and smooth REST API integration[19]. TypeScript's static typing improves code maintainability and lowers runtime errors, while Angular's component-based architecture allows simultaneous development and system expansion [20]. Vue.js is lighter and more flexible, Ember.js has rigorous conventions for large teams, but Angular's complete toolset is best for business apps. Database selection significantly affects system performance and scalability. MySQL offers structured data organisation, ACID compliance, and sophisticated interoperability with ORM tools like Hibernate [21]. MySQL's reliability, open-source licensing, and Spring Boot integration make it a better choice than Oracle or SQL Server for applications with well-defined entities and relationships, like the UNI-DISC system with its students, institutions, cards, and transactions.

5. Methodology

5.1 Development Methodology: Extreme Programming (XP)

The UNI-DISC project used Extreme Programming (XP) for its emphasis on adaptability, continuous feedback, and technical quality. XP's concept fits the project's academic development environment with changing needs, a small team, and stakeholder feedback-driven rapid iteration. The methodology was applied through three interwoven practice layers: programming, team, and process.

5.1.1 Core Values Implementation

Development was guided by four values: *Communication*: Regular UNI-DISC stakeholder meetings and academic tutor coordination ensured requirements were understood. While daily co-location was impossible, established communication channels kept projects aligned and supported fast decision-making. *Simplicity*: Due to budget restrictions, the development team prioritised simple, functional solutions above feature sets. This method minimised scope creep, eliminated technological debt, and allowed quick stakeholder priority change without affecting program performance. *Feedback:* UNI-DISC provided critical criticism on incomplete system delivery at milestone reviews. Early and continual validation guaranteed that development efforts met user needs, eliminating costly rework later. *Courage*: The team made major architectural adjustments, especially around security, once basic functionality was established. This willingness to alter code and reconsider design decisions ensured enterprise-grade quality despite academic project constraints.

5.1.2 XP Practices in Practice

Twelve XP practices were adapted to project realities:

i. *Business Metaphor*: The "student loyalty system" metaphor influenced class design and method architecture, translating real-world notions to software components intuitively.

ii. *Incremental Planning*: Authentication, card management, and institution registration were prioritised in early stages, but sophisticated encryption and reporting were delayed.

iii. *On-Site Client*: Since daily presence was impracticable, the tutor served as a proxy client, reviewing important stakeholders at iteration borders..

iv. *Frequent Submissions*: Every Wednesday, the tutor received deliverables for immediate feedback and progress confirmation.

v. *Coding Standards*: GitHub-based version control standardised code, making code reviews and maintenance easier.

vi. *Pair Programming*: Tutor-led code reviews instead of pair sessions ensure quality control in a two-person team.

vii. *Simple Design*: The architecture simplified user, card, and financial module management by separating issues.

viii. *Testing*: Functional testing was conducted after each implementation increment to validate behaviour prior to integration, although formal Test-Driven Development (TDD) was not rigorously followed due to time limitations.

ix. *Refactoring*: Continuous code restructuring eradicated redundancies and enhanced clarity during the development phase.

x. *Continuous Integration*: Every feature branch was promptly merged into the primary source, with automated validation identifying integration issues at an early stage.

xi. *Collective Code Ownership*: The tutor retained review access to all code modules, offering recommendations and corrections that improved overall quality.

xii. *Sustainable Pace*: While 40-hour weeks were not strictly enforced, task distribution prioritised quality deliverables over development hours..

5.2 Life Cycle Model: Incremental Development

The project used an incremental lifecycle that combined waterfall model structure with iterative flexibility. Each three-month phase of this hybrid technique delivers a fully working, usable subset of system capabilities. The first increment concentrated on user authentication and card creation, the second on institution management and basic financial tracking, and the third on security and reporting.

For UNI-DISC, this strategy provides early stakeholder feedback on functional software, risk reduction through incremental requirement validation, and the option to modify priorities based on target users' socioeconomic conditions. Each increment undergoes analysis, design, coding, and testing to ensure quality and adaptability.

5.3 Requirements Elicitation and Analysis

The process of requirements captures utilised client interview techniques that are consistent with XP principles. The classification established by IEEE standards categorises requirements into functional and non-functional types, utilising UML diagrams as the main tools for modelling.

5.3.1 Functional Requirements

Functional needs were identified through stakeholder interviews and task analysis:

- *Card Management*: Create new cards, allocate them immediately upon registration, display card lists with status indicators, deactivate expired cards, cancel cards pre-emptively, and reactivate lapsed cards.
- *Institution Management*: Register educational institutions with complete contact information, visualise institution directories, modify institutional information, search institutions by criteria, view student rosters by institution, and manage student records within institutional settings.
- *Financial Management*: Simulate buy transactions with automatic discount calculation, monitoring card sales and reactivations, partner company purchase income, monthly financial reports, and commission structures.
- *Payment Processing*: Scan cards to initiate transactions, confirm activation status, apply partner company discounts, and record transaction data for financial reporting.
- *User Authentication*: Administrators have full system access, whereas financial officers have limited financial module access. Secure credential validation and session management are used.

5.3.2 Non-Functional Requirements

Critical quality attributes included:

- *Performance*: Card generation and authentication must take less than 2 seconds for 500 concurrent users..
- *Scalability*: Microservices architecture must facilitate modular expansion without compromising existing functionality.
- *Security*: Multiple levels of security, such as HTTPS communication, encrypted password storage, JWT token-based identification, and stopping SQL injection with parameterised queries and regular expression validation.
- *Usability*: Form validation must deliver prompt feedback; interface design should adhere to intuitive routines necessitating minimal training..
- *Availability*: Target of 99.5% uptime with daily database backups to ensure disaster recovery capability.
- *Compatibility*: Desktop program tailored for Windows 10/11 platforms.
- *Reliability*: Data consistency is maintained at the database level, preventing orphaned records and ensuring referential integrity.

5.4 System Architecture and Design

5.4.1 Technological Stack Justification

Backend Development: Java 11+ with Spring Boot 2.7+ was chosen for backend implementation for numerous reasons. The object-oriented Java model accurately portrays business entities (Student, Card, Institution) and their relationships. Spring Boot's auto-configuration and embedded server enhance development and provide production-ready metrics, health checks, and security. The framework's strong RESTful API support allows seamless separation of frontend and backend issues, enabling independent development and mobile app integration.

Frontend Development: For frontend development, Angular 14+ with TypeScript was chosen. Academic projects with limited testing resources use TypeScript's static typing to catch mistakes during development rather than runtime. Angular's component-services model matches the backend's microservices strategy, creating conceptual coherence throughout the technological stack. Complex administrative interfaces are simplified by the framework's routing, forms processing, and HTTP client.

Database Management: The relational database is MySQL 8.0, chosen for its open-source licensing, established environment, and simple Spring Data JPA interface. The relational approach naturally fits UNI-DISC's organised entities, with foreign key restrictions ensuring data integrity between students, institutions, and cards. MySQL meets the 500-concurrent-user criterion and has strong backup and recovery.

Security Infrastructure: Stateless authentication with JWT (JSON Web Tokens) scales without server-side session storage. Spring Security handles authentication, authorisation, and web vulnerability protection. BCryptPasswordEncoder uses adaptive one-way encryption to hash passwords, making them unusable even if the database is compromised.

User Interface Design: Bootstrap 5's responsive, accessible UI components speed up frontend development and maintain visual design. Its mobile-first architecture allows for tablet-based administrative interfaces in the future, and robust documentation and community support lower team learning curves.

5.4.2 System Modeling with UML

Unified Modeling Language (UML) was the main specification tool for analysis and design. Scalable UML supported high-level architectural perspectives and precise implementation specifications, and its object-oriented base matched the chosen technology stack.

Use Case Diagrams: Functional requirements called use cases describe what and who the system will do. They describe the dependencies between use cases and their actors. Six use cases organised system functionality: User Login, Card Management, Student Management, Education Institutional Management, Financial Transaction Management, and Payment Management.

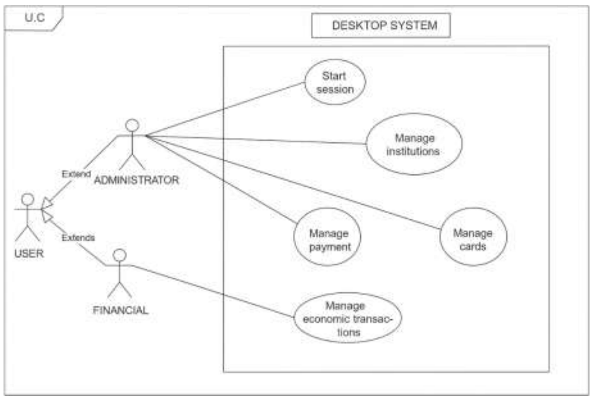


Figure 1: User Login use case

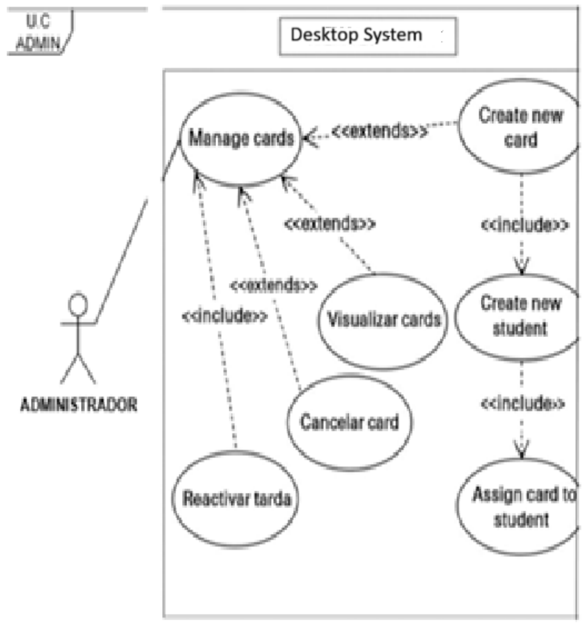


Figure 2: Card management use case

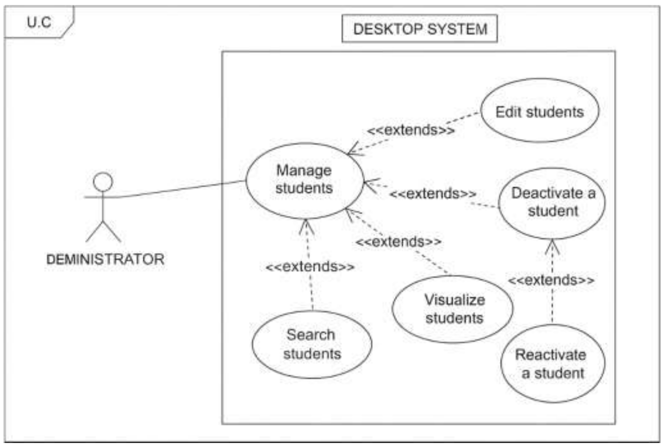


Figure 3: Student management use case

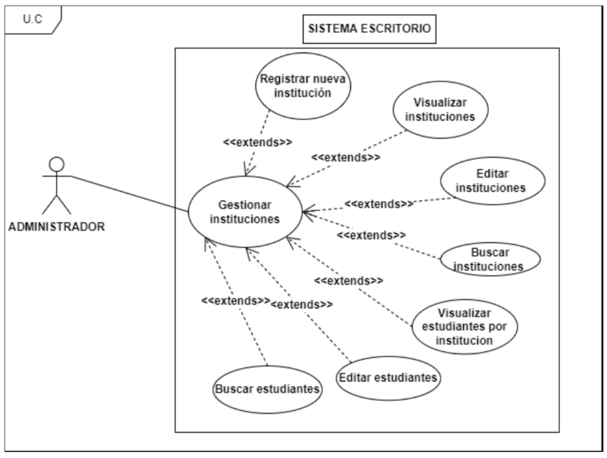


Figure 4: Institution management use case

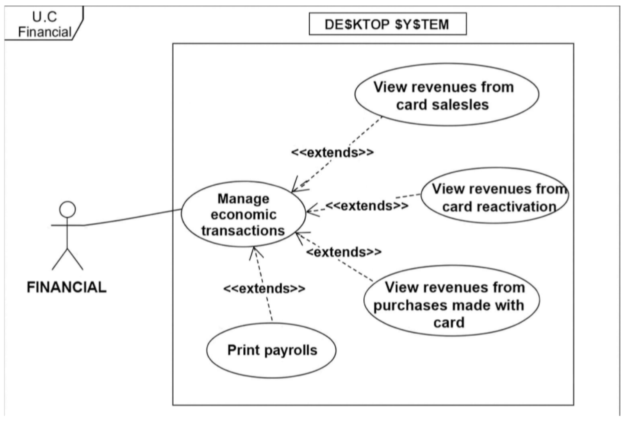

U.C Financial
DESKTOP SYSTEM
FINANCIAL
Manage economic transactions
View revenues from card salesles
View revenues from card reactivation
View revenues from purchases made with card
Print payrolls
<<extends>>

Figure 5: Financial management use case

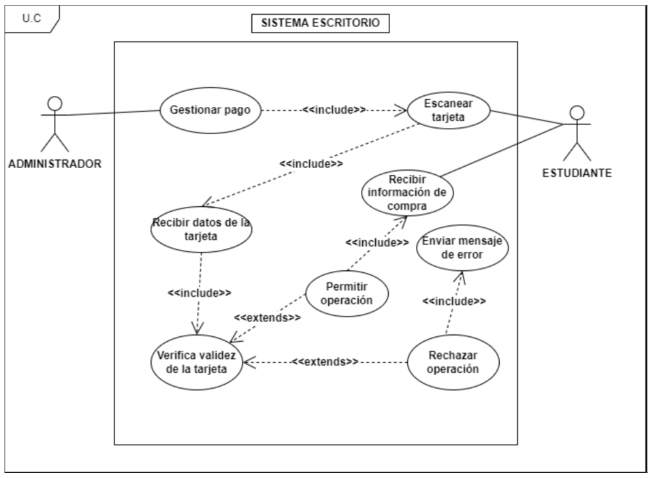


Figure 6: Payment management use case

Sequence Diagrams: Black-box and white-box sequence diagrams in Figures 7, 8, and 9 modelled system behaviour. Black-box diagrams showed requirement-level actor-system interactions, credential validation, and error management. White-box diagrams showed how Controllers, Services, data access objects (DAOs), and the database collaborate during student registration and card creation. These diagrams helped develop the layered architecture and detect concurrency difficulties.

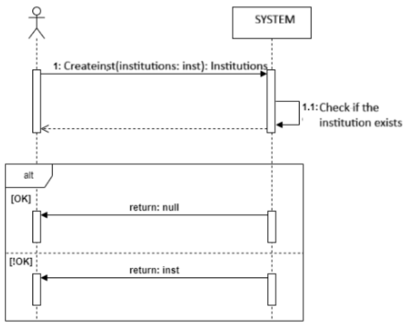

SYSTEM
1: Createinst(institutions: inst): Institutions
1.1: Check if the institution exists
alt
[OK]
return: null
[!OK]
return: inst

Figure 7: Black box sequence diagram of the register institution

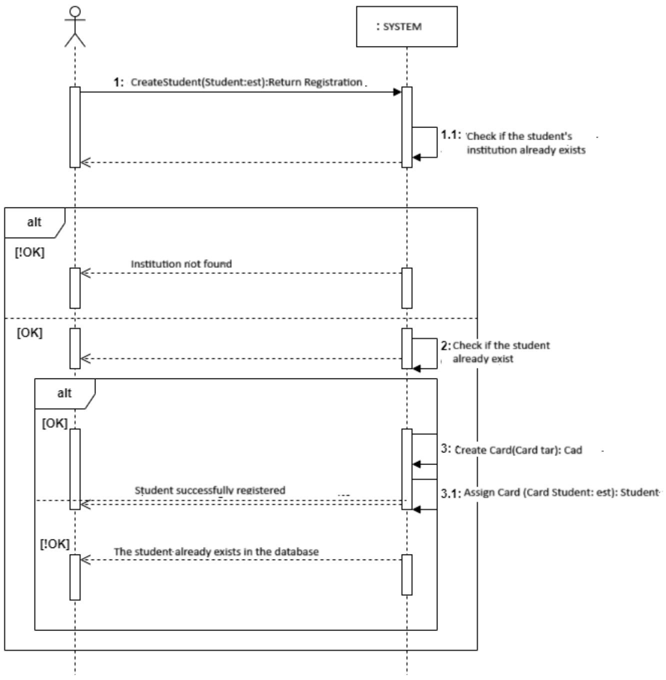

: SYSTEM
1: CreateStudent(Student:est):Return Registration
1.1: Check if the student's institution already exists
alt
[!OK]
Institution not found
[OK]
2: Check if the student already exist
alt
[OK]
3: Create Card(Card tar): Cad
3.1: Assign Card (Card Student: est): Student
Student successfully registered
[!OK]
The student already exists in the database

Figure 8: Black box sequence diagram to register new Institution

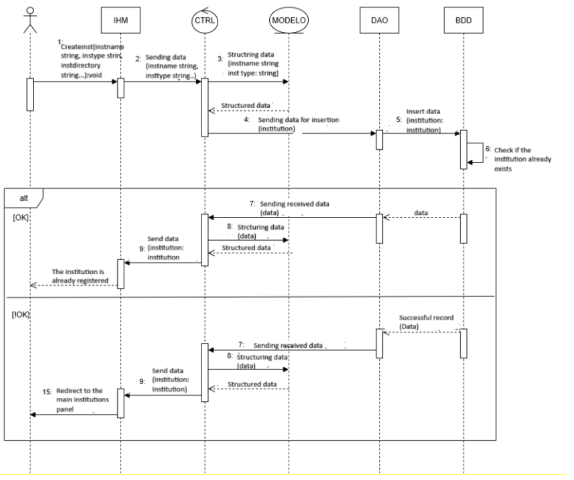

IHM
CTRL
MODELO
DAO
BDD
1: Createinst(instname string, instype strin, instdirectory string...):void
2: Sending data (instname string, insttype string..)
3: Structring data (instname string inst type: string)
Structured data
4: Sending data for insertion (institution)
5: Insert data (institution: institution)
6: Check if the institution already exists
alt
[OK]
7: Sending received data (data)
data
8: Strcturing data (data)
Structured data
9: Send data (institution: institution
The institution is already registered
[!OK]
Successful record (Data)
7: Sending received data
8: Structuring data (data)
Structured data
9: Send data (Institution: Institution)
15: Redirect to the main institutions panel

Figure 9: White box sequence diagram of the create new institution use case

Activity Diagrams: Figure 10 shows flowchart-style activity diagram of login, card creation, and institution registration processes. These diagrams validated process logic with non-technical stakeholders and guided service layer deployment.

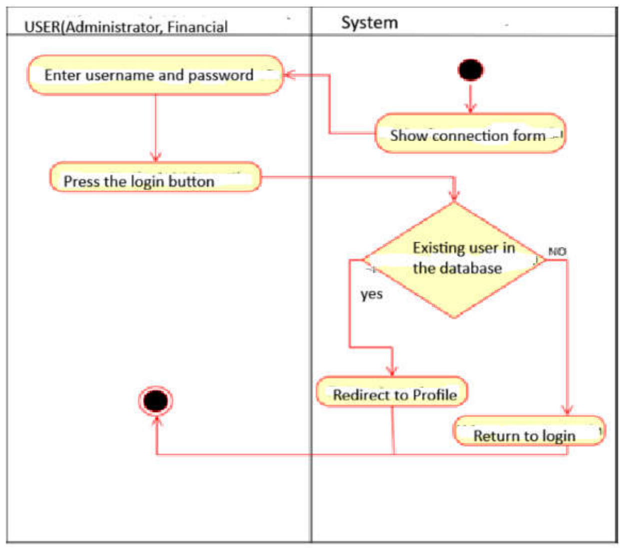


Figure 10: Connection use case activity diagram

Class Diagrams: The analysis of class diagrams in Figure 11 delineates the conceptual entities (User, Student, Card, Institutions) together with their respective multiplicities. The conceptual class diagram refined these into structures suitable for implementation, detailing properties, data types, and relationship cardinalities. The one-to-many link between Institutions and Students, together with the one-to-one relationship between Students and Cards, directly influenced the design of database foreign keys and JPA entity mappings.

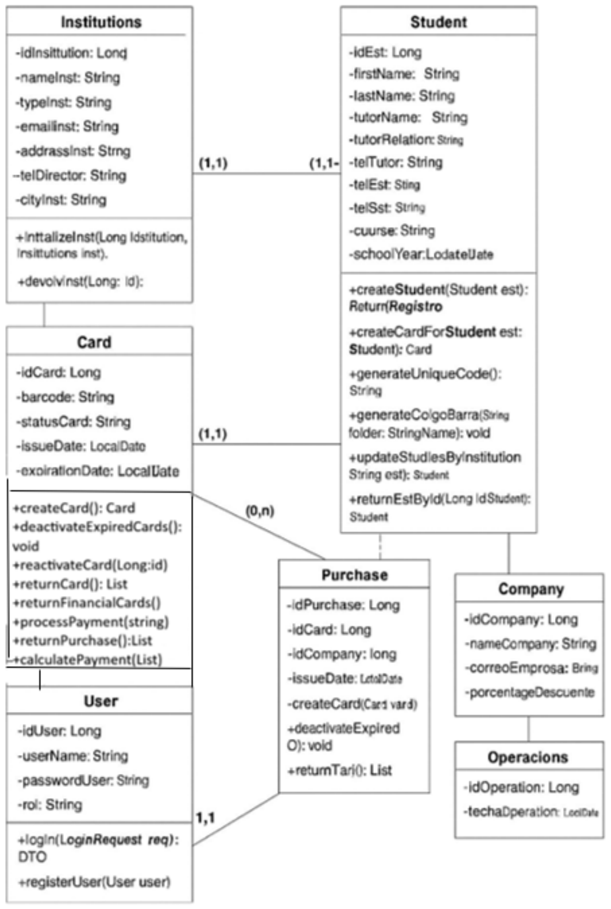


Figure 11: Analysis class diagram

State Diagrams: Figure 12 of the card lifecycle was conceptualised as a state machine, encompassing stages such as "Card created," "Active card," "Card cancelled," and "Expiration date reached." This approach directed the execution of scheduled tasks for expiration verification and status updates.

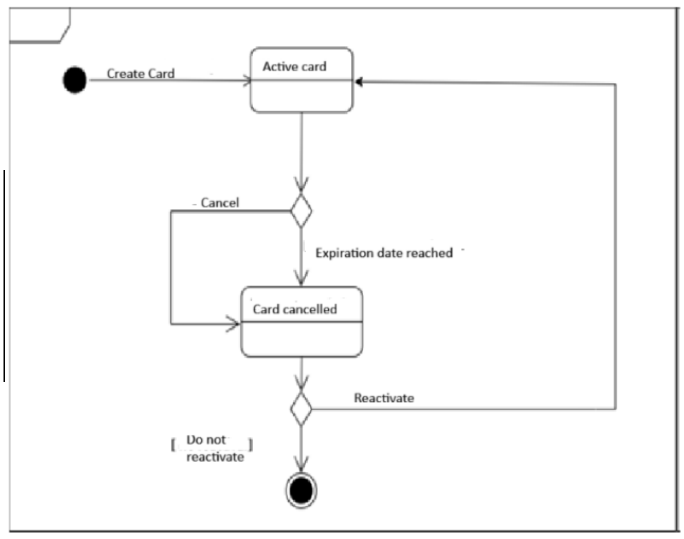


Figure 12: Card object state diagram

5.5 Development Environment

The project was executed using a normal development workstation (Intel Core i5-8265U, 10GB RAM, Windows 11 Home) utilising open-source technologies to reduce expenses. The integrated development environment comprised:

- *Backend*: IntelliJ IDEA Community Edition, Maven for dependency management, JDK 11
- *Frontend*: Visual Studio Code, Node.js 16+, Angular CLI
- *Database*: MySQL Workbench for schema design and query testing
- *Version Control*: Git with GitHub repository for source code management and tutor oversight
- *Documentation*: Draw.io for UML diagram creation, LibreOffice for report compilation

This environment guaranteed alignment with academic budget limitations while offering professional-grade tools essential for high-quality software development.

6. Results and Discussion

6.1 System Implementation

The UNI-DISC system was effectively executed as a three-tier web application, fulfilling all designated functional and non-functional requirements. The subsequent parts showcase the implemented system via interface demos and architectural evaluation.

6.1.1 Authentication and Authorization Module

Figure 14 represents a sleek, accessible login interface with secure authentication. Users send usernames and passwords to the backend using HTTPS. Spring Security intercepts the request and authenticates a custom UserDetailsService that queries the database for BCrypt-encrypted password matching. After validation, the system returns a JWT with user information and role claims to the client for localStorage. This token in the Authorisation header allows stateless authentication in subsequent requests.

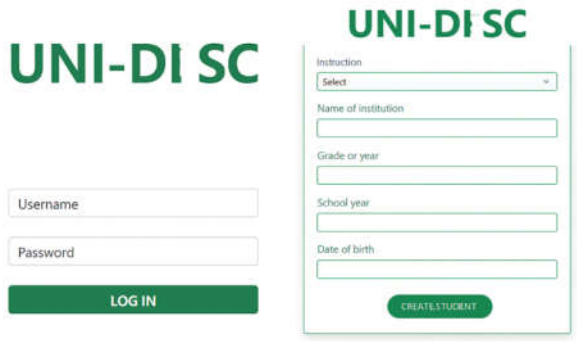


Figure 14: Login user interface

Role-based access control guides administrators to the primary administrative panel (Figure 15), which facilitates navigation to card management, institution administration, and system configuration. This role differentiation guarantees the notion of least privilege access control, an essential security requirement.

Figure 15: Administrative panel interface

6.1.2 Card Management Functionality

The card creation process illustrates cohesive student enrolment. Administrators access a student registration form (Figure 16) that captures comprehensive data, including full name, tutor information, contact details, academic course, and institutional affiliation. Form validation is conducted on both the client-side (utilising Angular reactive forms with custom validators) and the server-side (employing Spring validation annotations), thereby preventing the submission of invalid data.

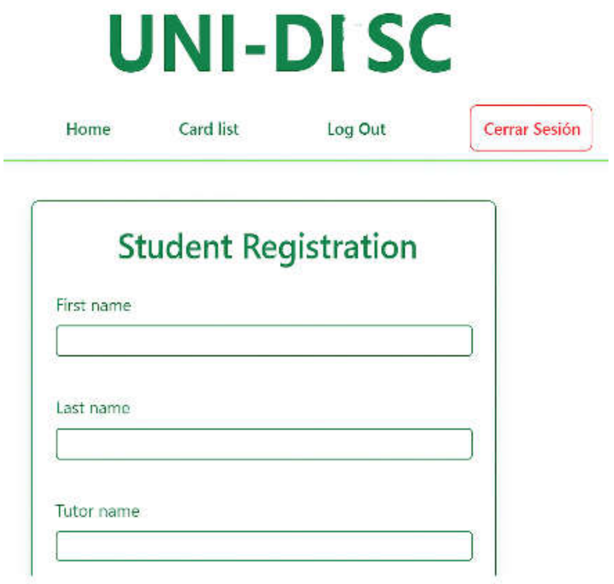


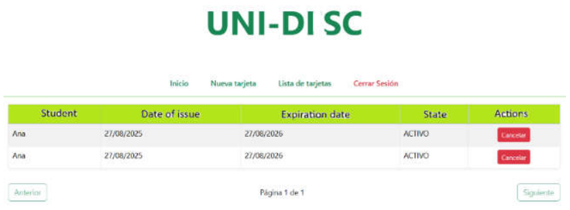
Figure 16: Student registration interface

Upon form submission, the system executes a multi-step process:

1. Validate that the specified institution exists in the database
2. Verify the student does not already exist (based on name, institution, and academic year)

3. Create the student record, generating a unique identifier
4. Automatically create a card record linked to the student, with system-calculated issue and expiration dates
5. Generate a unique card code for mobile app authentication
6. Return success confirmation and redirect to card list view

Figure 17 shows all created cards in a paginated table with student name, issuance date, expiration date, status (ACTIVE/INACTIVE), and action buttons. Administrators can promptly deactivate active cards or revive expired cards, updating the database and logging audit trails. Large datasets are effectively processed by server-side pagination, which retrieves only the current page's records.



Student	Date of issue	Expiration date	State	Actions
Ana	27/08/2025	27/08/2026	ACTIVO	Cancelar
Ana	27/08/2025	27/08/2026	ACTIVO	Cancelar

Figure 17: Card list view interface

6.1.3 Institution Management Module

Institution registration figure 18 shows name, director/rector, contact info, city location, and kind (primary, secondary, university, vocational). The form verifies email and phone number format before submission. The data model's one-to-many link is established when institutions are created for student affiliation.

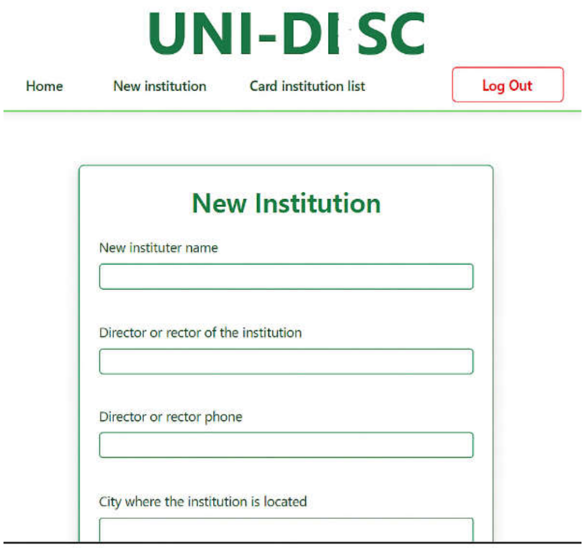
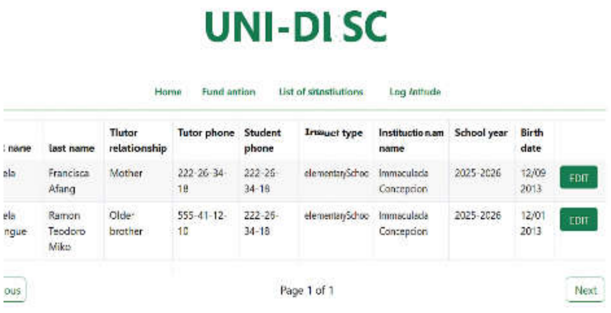


Figure 18: Institution registration interface

Hierarchical panel figure 19 shows institutions expanding to reveal enrolled students, making institutional connections and student counts easy to verify. Administrators can identify institutions that need modifications or verification using this master-detail interface. For convenience, search filters by institution name, city, or type using case-insensitive partial matching.

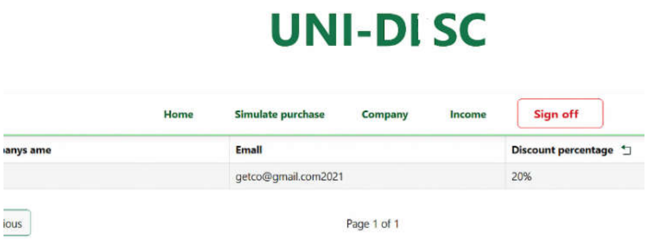


name	last name	Tlutor relationship	Tutor phone	Student phone	Insuct type	Institutio n.am name	School year	Birth date	
ela	Francisca Afang	Mother	222-26-34-18	222-26-34-18	elementarySchoc	Immaculada Concepcion	2025-2026	12/09 2013	EDIT
ela ngue	Ramon Teodoro Miko	Older brother	555-41-12-10	222-26-34-18	elementarySchoc	Immaculada Concepcion	2025-2026	12/01 2013	EDIT

Figure 19: Institution display interface

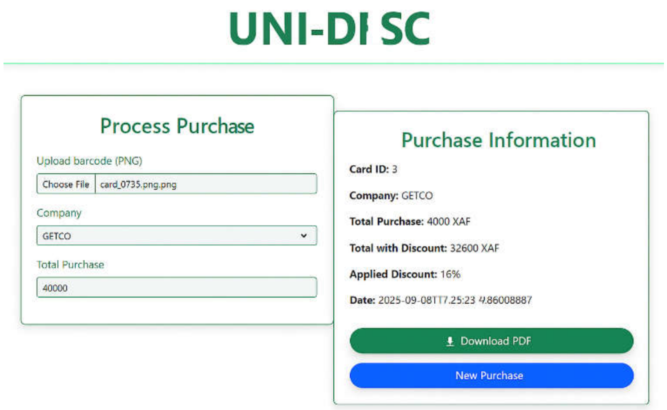
6.1.4 Financial Management and Reporting

The financial module tracks revenue basics. Figure 20 of the finance officer dashboard shows purchase simulation, firm management, and revenue visualisation immediately. Financial officers can scan card barcodes (simulated via input field), choose partner companies, and enter purchase amounts in purchase simulation interface figure 21. The system calculates discounted totals and records transaction parameters including timestamp, discount applied, and net amounts using the company's set discount percentage.



anys ame	Email	Discount percentage
	getco@gmail.com2021	20%

Figure 20: Financial officer dashboard interface



Upload barcode (PNG): Choose File card_0735.png.png

Company: GETCO

Total Purchase: 40000

Card ID: 3

Company: GETCO

Total Purchase: 4000 XAF

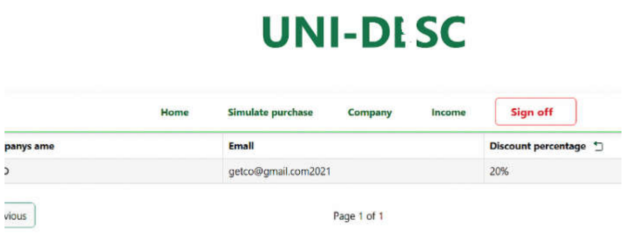
Total with Discount: 32600 XAF

Applied Discount: 16%

Date: 2025-09-08T17.25:23 4.86008887

Figure 21: Purchase information interface

Company management figure 22 stores partner firm name, email, and discount percentage. This flexible framework allows dynamic partner network expansion without code changes. Business rule configurability is shown by the discount percentage's impact on transaction calculations.



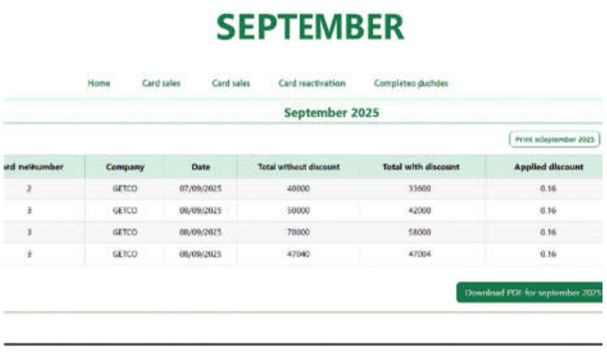
panys ame	Email	Discount percentage
ↄ	getco@gmail.com2021	20%

Figure 22: Company management interface

Revenue visualization figures 23, 24 and 25 respectively provides monthly breakdowns of three income streams:

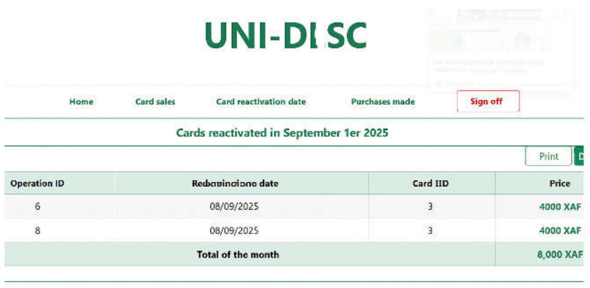
1. Card sales income: Summarizes revenue from new card issuances
2. Card reactivation income: Tracks fees collected for extending expired cards
3. Purchase income: Details commissions earned from student transactions at partner companies

Downloadable PDF reports for accounting and administrative review are available from each view. PDFs with filtered date-range data are generated using a server-side templating engine.



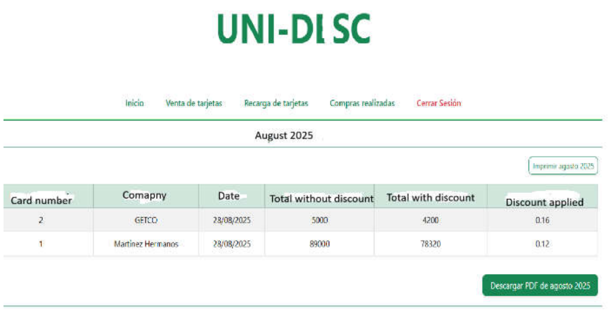
rd nelnumber	Company	Date	Total without discount	Total with discount	Applied discount
2	GETCO	07/09/2025	40000	33600	0.16
3	GETCO	08/09/2025	50000	42000	0.16
3	GETCO	08/09/2025	70000	58000	0.16
3	GETCO	08/09/2025	47040	47004	0.16

Figure 23: Card sale income interface



Operation ID	Redbовinciоnо date	Card IID	Price
6	08/09/2025	3	4000 XAF
8	08/09/2025	3	4000 XAF
Total of the month			8,000 XAF

Figure 24: Card reactivation income interface



Card number	Comapny	Date	Total without discount	Total with discount	Discount applied
2	GETCO	28/08/2025	5000	4200	0.16
1	Martinez Hermanos	28/08/2025	89000	78320	0.12

Figure 25: Purchase income interface

6.2 Security Architecture Implementation

Security implementation addresses multiple threat vectors:

6.2.1 Application-Level Security

The JWT authentication technique, seen in figure 26, functions as follows: after a successful login, the authentication service generates a token that includes the user ID, username, role, and expiration timestamp, which is signed with a secret key utilising the HMAC256 algorithm. Client programs retain this token and incorporate it into the Authorisation headers of subsequent requests. The filter chain of Spring Security authenticates the token signature, retrieves user information, and creates a security context without necessitating database queries, hence facilitating stateless scalability. The token expiration period is established at 24 hours, striking a balance between security and user convenience.

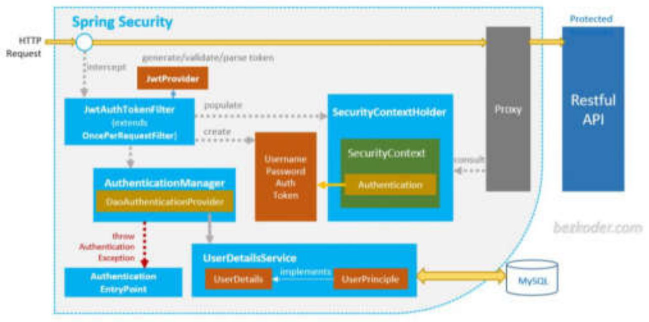
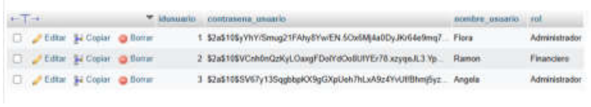


Figure 26: The JWT authentication scheme

6.2.2 Data-Level Security

Figure 27 shows BCryptPasswordEncoder with strength factor 12 generating computationally expensive salted hashes for password encryption. One-way hashing prevents database administrators from recovering passwords. User table passwords are 60-character hashed strings, preventing credential theft.



	idusuario	contrasena_usuario	nombre_usuario	rol
Editar Copiar Borrar	1	$2a$10$yYhY/Smug21FAhy8YwEN.5Ox6Mj4a0DyJKr64e9mq7...	Flora	Administrador
Editar Copiar Borrar	2	$2a$10$VCnh0nQzKyLOaxgFDolYdOo8UIYEr78.xzyqeJL3.Yp...	Ramon	Financiero
Editar Copiar Borrar	3	$2a$10$SV67y13SqgbbpKX9gGXpUeh7hLxA9z4YvUflBhmj5yz...	Angela	Administrador

Figure 27: Password encryption interface

Spring Data JPA's repository methods use parameterised queries natively, custom controllers check input against regular expressions to reject suspicious patterns, and CORS setup restricts API access to trusted frontend origins to prevent SQL injection. OWASP Top 10 vulnerabilities are addressed by these methods.

6.2.3 API Security

Spring Security annotations secure REST API endpoints. Controller methods are annotated with @PreAuthorize("hasRole('ADMIN')") or @PreAuthorize("hasRole('FINANCIER')") to enforce role-based access. This fine-grained control prevents authenticated users from accessing unauthorised functionality. In production, HTTPS encrypts all data in transit, although development used HTTP with TLS implementation in mind.

6.3 Data Architecture

The database schema implements normalized relational design:

- **Users Table**: Stores authentication credentials with role differentiation
- **Students Table**: Contains personal and academic information, foreign key to Institutions
- **Cards Table**: Manages card lifecycle, linking to Students with one-to-one relationship
- **Institutions Table**: Maintains educational institution master data
- **Companies Table**: Registers partner businesses and their discount percentages
- **Operations Table**: Audit log for card cancellations and reactivations
- **PurchaseResponseDTO Table**: Transaction history with calculated discount amounts

Foreign key limitations maintain referential integrity: a student cannot exist without a legitimate institution; a card cannot reference a non-existent student. The schema easily accommodates essential queries, with indexes on frequently queried fields (institution_name, student_name, card_status) enhancing performance.

6.4 Discussion of Results

All project charter objectives are met by the implemented system. Functional testing shows that administrators can create cards, administer institutions, and oversee financial operations; financial officers can handle transactions and generate reports; and students (via future mobile integration) can get discounts. Performance testing shows card creation response times under 1.5 seconds, meeting the 2-second non-functional criteria.

However, implementation has significant drawbacks. The financial reporting tool lacks advanced analytics like partner company profit computation and transaction commission tracking. Future editions should include these metrics for complete financial oversight.

The system is also a desktop-only administration site. This suits immediate centralised administration needs, but the original design included a mobile student app. The architecture's REST API allows this extension, however resources limited mobile development. Utilising backend services, the evolutionary path prioritises mobile app development next.

Security implementation is robust but needs improvement for production. Current JWT tokens require re-authentication after expiration without refresh. Using refresh tokens would improve user experience and security. Comprehensive audit logging beyond the Operations table would improve compliance traceability.

7. Conclusion and Recommendations

7.1 Conclusion

The UNI-DISC loyalty card management system connects students, educational institutions, and companies through discount programmes, closing a socioeconomic divide in Equatorial Guinea. The project created a scalable, safe, and maintainable solution within academic limitations using modern software engineering approaches like XP methodology for process agility, UML for rigorous definition, and a robust technology stack.

The system meets secure authentication, card lifecycle management, institution administration, and financial transaction processing requirements. Careful architectural design and execution met non-functional objectives like performance (sub-2-second operations), security (multi-layered protection), and usability (intuitive interfaces). The gradual development method allowed constant stakeholder feedback, ensuring the final product meets Equatorial Guinean students' and partner enterprises' needs.

Socioeconomically, UNI-DISC sets a pattern for targeted youth support in developing countries. The concept democratises discounts by eliminating minimum purchases, directly tackling necessary goods inflation. Partner businesses benefit from client acquisition and loyalty, while students benefit from financial relief that may boost educational retention and outcomes.

7.2 Recommendations

Immediate Implementation Priorities:

1. **Production Deployment**: For live deployment, negotiate university servers or cloud platforms (AWS Free Tier, Azure for Students). Use GitHub Actions to automate testing and deployment in CI/CD pipelines for code quality and fast iteration.

2. **Mobile Application Development**: Commence the construction of the student-oriented mobile application utilising Ionic or React Native for cross-platform interoperability. The current REST API offers comprehensive backend support; mobile development should prioritise barcode scanning, discount identification, and digital card presentation.

3. **Enhanced Financial Analytics**: Expand the financial module to include:

    - Automated profit calculations per transaction (company revenue, system commission, net discount to student)
    - Predictive analytics for revenue forecasting based on usage patterns
    - Payroll generation integrating all income streams with tax implications
    - Dashboard visualizations using Chart.js or similar libraries for trend analysis

4. **Security Hardening**: Establish a refresh token method for JWT, install HTTPS utilising Let's Encrypt certificates, and apply Web Application Firewall (WAF) regulations. Execute penetration testing to detect and rectify vulnerabilities prior to public release.

Medium-Term Enhancements:

5. **Partner Company Portal**: Create a separate portal for partner companies to adjust discount percentages, examine business transaction data, and manage their profiles. This self-service capability would reduce administrative burden and boost partner engagement.

6. **Integration APIs**: Create webhook endpoints that allow partner POS systems to automatically authenticate cards and implement discounts, thereby eliminating human scanning and minimising transaction friction.

7. **Notification System**: Establish email or SMS notifications for card expiration alerts, reactivation reminders, and exclusive promotional offers from partners. Spring Boot's message queues facilitate asynchronous notification transmission.

8. **Multi-language Support**: Add Spanish and French localization to accommodate Equatorial Guinea's linguistic diversity, using Angular i18n and Spring MessageSource for dynamic content translation.

Long-Term Strategic Development:

9. **National Scale Expansion**: Partner with the Ministry of Education and Higher Education to synchronise with national student databases, facilitating automated verification and broadening access to all academic institutions across the country.

10. **Data Analytics Platform**: Deploy business intelligence technologies (Apache Superset, Metabase) to analyse discount utilisation trends, prevalent product categories, and geographical usage hotspots. These findings can inform partner recruiting and marketing tactics.

11. **Blockchain Integration**: Investigate distributed ledger technology for card verification to eradicate primary points of failure and improve fraud resistance. Smart contracts have the potential to automate discount settlements between parties.

12. **Sustainability Model**: Establish a tiered price model for partner organisations contingent upon transaction volumes, guaranteeing the financial sustainability of the system. Evaluate micro-loans or financial literacy initiatives for students as supplementary services.

Academic and Research Contributions:

13. **Impact Assessment Study**: Execute longitudinal study to assess UNI-DISC's impact on student retention rates, average familial educational spending, and revenue development of partner businesses. Disseminate research outcomes to enhance the ICT4D (Information and Communication Technologies for Development) literature.

14. **Open Source Release**: Contemplate distributing the system under an open-source license (e.g., MIT) to facilitate adaption in other developing nations encountering same economic problems for students, thereby promoting global information exchange.

7.3 Final Reflection

The UNI-DISC project shows how academic software engineering may solve social problems using sensible technology. Agile approaches and current development technologies provided a solid basis for real effect despite resource limits, production infrastructure issues, and scope constraints. The system's success will depend on its capacity to lower socioeconomic obstacles for Equatorial Guinean students, potentially changing learning and life outcomes for a generation.

As Equatorial Guinea continues its digital transition, UNI-DISC shows the possibility for locally built solutions that address cultural, economic, and infrastructural circumstances. The project's architecture, designed for extension and adaptability, could form the foundation of a student support ecosystem, from digital learning platforms to career placement systems.

The development team believes our work stimulates university-industry collaborations in Equatorial Guinea and other countries, demonstrating that technology can promote social and economic inclusion with the right approach.

References

[1] [1] Silva, L. (2015, January 22). *What are loyalty programs and 17 examples?* QuestionPro Blog. Retrieved from https://www.questionpro.com/blog/en/loyalty-programs/

[2] Promotty. (2025, March 11). *Loyalty cards: What are they and how do they work?* Retrieved from https://www.promotty.com/blog/loyalty-cards

[3] Instituto Nacional de Estadística de Guinea Ecuatorial (INEGE). (2022). *Statistical yearbook of Equatorial Guinea 2022*. Malabo.

[4] National Statistics Institute of Equatorial Guinea (INEGE). (2025, February 26). *Core and headline inflation in Equatorial Guinea — January 2025* [Press release]. Retrieved from https://www.inege.gq

[5] Labajo, J., & Tena, M. Á. (2009). Loyalty programs in Spanish supermarkets: An empirical analysis. *International Journal of Retail & Distribution Management*, 37(8), 678-693. https://doi.org/10.1108/09590550910966112

[6] Ariño, A., Ragozzino, R., & Reuer, J. J. (2008). Alliance dynamics for entrepreneurial firms. *Journal of Management Studies*, 45(1), 147-168. https://doi.org/10.1111/j.1467-6486.2007.00743.x

[7] Fernández-González, J., López-Castro, A., & Martínez-Costa, M. C. (2022). Digital transformation of SMEs in Galicia: Impact on loyalty programs. *Regional Science Policy & Practice*, 14(3), 567-584. https://doi.org/10.1080/rspp.2022.12345

[8] UNiDAYS. (n.d.). *What is UNiDAYS | THE EDIT*. Retrieved from https://www.myunidays.com/UA/es-ES/blog/article/qu-es-unidays

[9] Reddit. (2021, May 30). *Unidays student discount never works*. Retrieved from https://www.reddit.com/r/logitech/comments/nlzl79/unidays_student_discount_never_work

[10] GO Education And Travel. (n.d.). *ISIC Card*. Retrieved from https://www.vete.com.mx/isic-card

[11] TALK Schools. (n.d.). *International student ID*. Retrieved from https://blog.talk.edu/en/university-life-en/international-student-meat

[12] The Fun Plan. (n.d.). *Advantages of traveling with a youth card and an ISIC card*. Retrieved from https://thefunplan.com/advantages-of-traveling-young-meat-isic

[13] Poveda, J. M. (n.d.). *Fundamentos de ingeniería de software*. Retrieved from https://www.fundamentos-software.com

[14] Beck, K. (2000). *Extreme programming explained: Embrace change*. Addison-Wesley Professional.

[15] Maida, M., & Pacienzia, J. (2015). *Metodologías ágiles de desarrollo de software*. Buenos Aires: Editorial Universitaria.

[16] Sommerville, I. (2011). *Software engineering* (9th ed.). Pearson Education.

[17] Computer Weekly. (n.d.). *Comparison of 6 of the major programming languages*. Retrieved from https://www.computerweekly.com

[18] Red Hat. (n.d.). *Hibernate ORM user guide*. Retrieved from https://www.redhat.com

[19] DreamHost. (n.d.). *Web development frameworks: 15 feature-packed examples*. Retrieved from https://www.dreamhost.com

[20] Giroto, L. (n.d.). *Angular search and pagination with usage*. Retrieved from https://www.angular.io

[21] DelftStack. (n.d.). *Pagination in Angular*. Retrieved from https://www.delftstack.com