**INTERNATIONAL BURCH UNIVERSITY**

FACULTY OF ENGINEERING, NATURAL AND MEDICAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY



**SOFTWARE ASSISTING TEACHING STAFF IN TESTING PROCEDURES USING RFID TECHNOLOGY**

UNDERGRADUATE PROJECT

ELDIN GUZIN

MSc Nedim Bandžović

SARAJEVO

June, 2024

SOFTWARE ASSISTING TEACHING STAFF IN TESTING PROCEDURES USING RFID TECHNOLOGY

ELDIN GUZIN

Report Submitted in Fulfillment of Requirement for the

Undergraduate Project

INTERNATIONAL BURCH UNIVERSITY

2024/2025

**APPROVAL PAGE**

| **Student name and surname:** | *Eldin Guzin* |
| --- | --- |
| **Faculty:** | Faculty of Engineering, Natural and Medical Sciences |
| **Department:** | Information Technologies |
| **Project Title:** | *Software Assisting Teaching Staff In Testing Procedures Using RFID Technology* |
| **Date of Defense:** | *July 14, 2025* |

I certify that this final work satisfies all the requirements as an undergraduate project for the bachelor's degree in information technologies.

|  |  |
| --- | --- |
|  | Assoc. Prof. Dr. Dino KEČO  **Head of Department** |

This is to certify that I have read this final work and that, in my opinion, it is fully adequate, in scope and quality, as an undergraduate project for the bachelor's degree in information technologies.

|  |  |
| --- | --- |
|  | (name, with academic title)  **Mentor** |

**Examining Committee Members**

|  | **Title / Name and Surname** | **Affiliation** | **Signature** |
| --- | --- | --- | --- |
| 1. | Assist. Prof. Dr. Dželila Mehanović | head |  |
| 2. | MSc Nedim Bandžović | member |  |
| 3. | MSc Adnan Miljković | member |  |

This final work has been approved as being written in compliance with the formatting rules laid down by the Department of Information Technologies.

**SOFTWARE ASSISTING TEACHING STAFF IN TESTING PROCEDURES USING RFID TECHNOLOGY**

**ABSTRACT**

Manual attendance taking during academic examinations and lectures tends to be a time-consuming and error-prone process, especially when it comes to a large amount of inputs at the same time. This senior design project addresses the need for a more efficient method of monitoring student attendance during lectures and exams. To tackle this issue, this project proposes a software-assisted system that uses RFID (Radio Frequency Identification) technology integrated with an Arduino microcontroller. Each student already has their unique student identification card that has an integrated RFID chip. With the given software, it can be scanned upon entering the examination room or a classroom. The scanned data is immediately transmitted to a Ruby on Rails based web application that logs attendance records in real time. This system supports secure authentication, timestamped logs and intuitive administrative interface for educators to monitor attendance activity, enhances transparency in the testing process, strengthens exam policy enforcement and ensures that attendance data is accessible in a digital format.. The project combines low-cost hardware components such as RFID readers and Arduino boards, with robust web development practices. Arduino hardware acts as the physical interface for RFID scanning, while the backend web application performs data processing, storage and visualization.

**LITERATURE REVIEW**

In this section we present literature regarding the issue on the topic. [1] The authors built the system that integrates hardware for real-time student identification and software development using Microsoft Visual Studio in a three-tier B/S architecture, offering efficient data management and ease of maintenance. By integrating an improved dynamic frame time slot ALOHA algorithm and an iterative Taylor series expansion positioning method, she system achieves enhanced robustness and reduces the average positioning error below 10, outperforming traditional approaches in indoor RFID environments. [2] This research highlights the inefficiencies of traditional attendance systems and presents an RFID-based solution that automates attendance tracking using RFID tags and readers integrated with a secure real-time database. The proposed system demonstrates improved accuracy and efficiency validated through prototype testing in academic settings. [3] This paper explores the integration of RFID technology in attendance focusing on the use of Arduino microcontrollers and RFID readers. The system was tested for real-time data synchronization, user interface functionality and performance metrics. [4] This study presents RFID-based lab inventory and attendance system that uses NodeMCU, MySQL and PHP for real-time data handling. It incorporates machine learning to analyse experiment usage with linear regression achieving the best performance (MSE = 0.14). Key innovations include multi-tag detection. [5] Key studies in this literature emphasize enhancements like virtual reference tags for broader academic utility, and the potential for hybrid systems to overcome individual limitations. Future improvements focus on biometric integration, wireless communication and online database support. [6] This study proposes an RFID and GSM-based attendance system that automates student check-ins via RFID cards, with real-time SMS notifications to parents. The system improves accuracy, minimizes manual errors and offers special benefits for elderly and disabled users by emphasizing comfort and ease of monitoring. [7] The study presents an RFID-based attendance and security system that automates student tracking using Arduino and ATMEGA328P, significantly reducing manual errors and saving time. The system logs arrival/departure times, improves security by restricting access and features real-time updates LCD feedback and data storage via EEPROM – demonstrating high accuracy and ease of use in pilot tests. [8] The reviewed literature highlights efficient smart attendance systems using Arduino-based hardware with RFID, IR and biometric sensors. These systems automate attendance recording, reduce fraud and enable real-time data transmission to web servers or databases for reporting.

Key innovations include combining multiple sensors to improve accuracy and prevent manipulation. [9] This study presents a smart attendance system integrating RFID and IoT to automate and enhance student attendance tracking, replacing manual roll calls. Using UHF RFID tags, Raspberry Pi and Microsoft Azure, the system offers real-time monitoring, SMS alerts and proxy prevention. [10] This study proposes an RFID-based attendance system integrated with IoT and machine learning for efficient student monitoring. It highlights the use of NodeMCU ESP8266 and RC522 modules to automate attendance logging and enable time-series forecasting using Facebook’s Prophet toolkit to predict attendance trends. [11] This study presents real-time RFID-based attendance using Arduino, LCD, SD card and RTC modules to automate and accurately record employee attendance. The system differentiates on-time and late entries via visual (LED) and audio (buzzer) alerts, significantly improving over manual tracking methods. [12] The study presents an RFID-based attendance system that automates employee tracking and office operations using secure, personalized RFID cards. It integrates fingerprint and Pin authentication, encrypted data storage and supports real-time monitoring and automation such as lighting and access control.

**ACKNOWLEDGMENTS**

During my studies, there are many people who have contributed to my overall university experience. I would like to acknowledge some of my professors and assistants who sparked a greater interest in my academic journey because of their sincere devotion to teach us everything we know today. Those go to Elma Avdic, Dino Keco and Aldin Kovacevic.

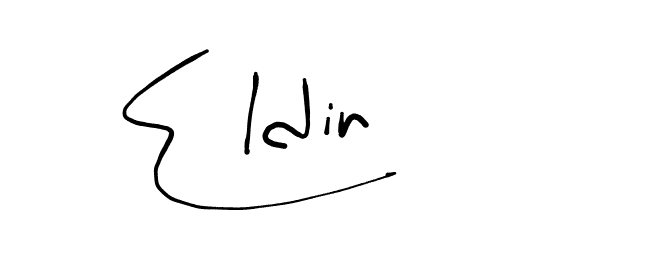
I would also like to acknowledge my mentor Nedim Bandzovic for guiding me through each step of my senior project.

Of course, at the end of the journey it’s the friends we made along the way, so I want to specially thank Basar Carovac for being the best friend I could have these past few years. I would also like to mention and thank Orhan Huseinbegovic, Anja Sehovac and Ridvan Vejzagic, for every lunch we had in the caffeteria, help and guidance, every project we went through together and every laugh we had that made university so much more fun and interesting.

Thanks to my parents Dzenana and Zlatko Guzin and my sister Amina, who have shaped me into the person I am today with their endless support.

At last, as a special thank you, I want to thank my girlfriend Alejna Hasanagic who truly made my university years blossom by her support in everything I do and who I aspire to be.

**DECLARATION**

I hereby declare that this undergraduate project titled **Software assisting teaching staff in testing procedures using RFID technology** is based on my original work except quotations and citations, which have been duly acknowledged. I also declare that this work has not been previously or concurrently submitted for the award of any degree, at International Burch University or any other university or institution.

|  |  |
| --- | --- |
|  | Eldin Guzin  June 26th, 2025 |

**TABLE OF CONTENTS**

[**1. INTRODUCTION 1**](#_3rdcrjn)

[**2. SYSTEM ANALYSIS 3**](#_1y810tw)

[2.1. System Overview and Scope 3](#_o3q3ydcfi66d)

[2.2. Functional Requirements Analysis 4](#_9bd8nhb5q41j)

[2.3. Non-Functional Requirements 4](#_8dx6ogbkivtv)

[2.4. Technology Stack Analysis 5](#_tu58eu39xv7m)

[2.5. Risks and Mitigation 6](#_m8xiy3bikl29)

[**3. APPLICATION DESIGN 7**](#_1ci93xb)

[3.1. Use Case Diagram and System Actors 7](#_mr0c4sflwzuw)

[**4. METHODOLOGY 13**](#_3whwml4)

[**5. SYSTEM TESTING 22**](#_2bn6wsx)

[**6. RESULTS 28**](#_qsh70q)

[**7. CONCLUSION 33**](#_3as4poj)

**TABLE OF FIGURES**

[Figure 3.1 Use Case Diagram 8](#_9hdxc611zdzx)

[Figure 3.2 System architecture diagram 9](#_in048lxc1wj9)

[Figure 3.3 Class Diagram 10](#_jf7hob3cqtlh)

[Figure 3.4 Entity relationship diagram 11](#_32wa9126dmqb)

[Figure 3.5 Entity relationship diagram 12](#_rfjznly6dy6t)

[Figure 4.1 File Tree 15](#_e9391b4lew7n)

[Figure 4.2 ArduinoApiController scan method 17](#_tfs2i0lk40uk)

[Figure 4.3 Arduino code snippet 18](#_wxozvixkcrh4)

[Figure 4.4 Database schema.rb file code snippet 20](#_nbl64ystedad)

[Figure 4.4 db/migrate folder showing database migrations 20](#_lycgwvh7p5r)

**LIST OF ABBREVIATIONS**

**API** Application Programming Interface

# INTRODUCTION

The management of student attendance in educational institutions has long been a manual, time-consuming process prone to errors. Traditional methods of recording attendance, particularly during examinations, often require significant administrative overhead and can disrupt the learning environment. As educational institutions continue to grow in size and complexity, the need for automated, reliable attendance monitoring systems becomes more important.

Modern educational institutions face several challenges when implementing traditional attendance systems. Paper-based sign-in sheets are vulnerable to fraud, difficult to manage for large classes, and time-consuming to process. Digital alternatives often require specialized hardware or complex setup procedures that create barriers to adoption. Furthermore, existing solutions frequently lack integration with institutional systems and fail to provide the real-time monitoring capabilities that educators require.

Radio Frequency Identification (RFID) technology presents an interesting solution to these challenges. Most educational institutions already issue student identification cards, and the integration of RFID chips into these cards provides a foundation for automated attendance tracking. RFID technology offers several advantages over alternative identification methods, including rapid scanning capabilities, durability, and cost-effectiveness. Unlike biometric systems, RFID does not require direct physical contact and can process multiple scans quickly without creating bottlenecks at entry points.

This project develops an RFID-based attendance system that addresses the specific needs of educational institutions while maintaining simplicity and cost-effectiveness. The system leverages existing student identification infrastructure and integrates seamlessly with standard institutional workflows. By combining Arduino microcontrollers with a modern Ruby on Rails web application, the solution provides both reliable hardware interfaces and sophisticated software capabilities.

The primary objectives of this project include: developing a user-friendly interface for educators to monitor attendance in real-time, creating a scalable system architecture that can accommodate institutions of varying sizes, implementing secure authentication and data protection measures, providing comprehensive reporting and analysis capabilities, and ensuring compatibility with existing institutional technology infrastructure.

The system addresses several key use cases within educational institutions. During examinations, the system enables rapid student check-in while maintaining accurate records for academic integrity purposes. In lecture halls, students can quickly scan their cards upon entry, allowing instructors to focus on teaching rather than administrative tasks. The system also supports flexible subject management, enabling institutions to organize attendance data across multiple courses, semesters, and academic programs.

From a technical perspective, the project demonstrates the integration of Internet of Things (IoT) principles with modern web development frameworks. The Arduino microcontrollers function as edge devices that collect RFID scan data and transmit it to the central web application through HTTP APIs. The Ruby on Rails framework provides robust data management capabilities, user authentication, and real-time updates through WebSocket connections. This architecture ensures both reliability and scalability while maintaining reasonable development and deployment costs.

The significance of this project extends beyond immediate attendance management benefits. The system generates valuable data that can inform institutional decision-making, including attendance pattern analysis, space utilization optimization, and student engagement metrics. Additionally, the project serves as a foundation for future enhancements, such as integration with learning management systems, automated grading workflows, and student information systems.

# SYSTEM ANALYSIS

In this part of the report, I will be breaking down how the RFID-based attendance system works. This includes a general overview of the system, what it’s supposed to do, the key requirements, and how practical the system is from a technical and operational point of view. It’s basically the groundwork that needs to be done before the actual system gets built, identifying everyone involved, what the system needs to do, and any potential limitations we might face.

## System Overview and Scope

The main idea behind this RFID attendance system is to make the whole process of tracking student attendance faster and more accurate by using RFID tech alongside a web app. The system includes physical components (like RFID readers), an online interface for managing data, and backend services that handle storage and processing.

There are a few main users of the system. Students will use RFID cards to scan in when they arrive. Teachers can check and manage attendance records and control subject-related settings. Then there are the admins who manage the users and system settings overall. On the technical side, we’re also dealing with the Arduino board, which reads the RFID data and sends it to the system. It could also be connected to other tools, like data analysis platforms.

The system is built to fit in with what schools and universities already have in place, using standard networking and hardware to make installation easier. Our goal is to cut down the time it takes to record attendance, achieve very high scan accuracy and provide a real-time view of who is present.

## Functional Requirements Analysis

The system needs to meet a range of functional needs to work properly in a school setting. First, there’s a login system so that users can access their accounts securely. Different types of users (admins, teachers, monitors) get different levels of access. The login process includes registering new users, handling sessions, and logging out automatically after a certain period.

Then there's subject management. Users should be able to create and manage as many courses as they want. Each course keeps its own records and settings. This makes it easier for the system to handle more complex institutions where multiple subjects and sessions are going on at once.

The RFID card system is designed to recognize and register student cards automatically. When a card is scanned for the first time, it gets saved in the system with a unique ID. Admins can also link cards to specific students, though it’s flexible enough to handle different privacy rules depending on the institution.

One of the key features is real-time processing. When a student scans their card, the system processes that scan in a very short time. It checks for duplicates, records the time, updates the database, and sends updates to the user interface using WebSocket connections so that attendance can be tracked live.

## Non-Functional Requirements

Performance-wise, the system should respond quickly—under 100 milliseconds for regular database tasks—and it needs to support over 100 users at the same time without slowing down. It also needs to keep up with multiple WebSocket connections to maintain real-time updates without draining resources.

Scalability is important too. We expect the system to keep running smoothly even when the database grows to 10,000 or more records. It should also be able to handle a lot of scan events, like more than 1,000 per minute, without crashing or delaying the processing.

Security is a major priority. I’ve built in layered protection, including secure logins, encrypted data transfers, and validation to prevent things like SQL injection or unauthorized access—all while keeping the system easy to use.

We also expect the system to be very reliable. It should recover gracefully from connection problems between Arduino devices and the web server, thanks to auto-retry features and error handling.

## Technology Stack Analysis

I chose Ruby on Rails as the framework for the web part of the system. It’s great for fast development and has lots of built-in tools for authentication, real-time updates, and APIs.

For the database, I used SQLite during development because it's easy to set up, but I made sure the system can be moved to PostgreSQL or MySQL when it’s time to go live, depending on what the institution prefers.

The Arduino board was chosen because it's affordable, reliable, and integrates well with RFID readers and web applications. It also uses basic HTTP communication, which keeps things simple.

For real-time updates, I used Turbo Streams (part of Hotwire), which works over WebSockets without needing complex JavaScript. It gives immediate feedback when scans happen and keeps the system efficient and easy to maintain.

## Risks and Mitigation

There are some technical risks, like connection issues between the Arduino and the server. I tackled this with error handling, retries, and local buffering of data. For hardware reliability, I suggest using backup components and standardizing the parts used.

Security-wise, I’ve put in measures to prevent common threats like hacking or data leaks by using access control, rate limits, and input checks.

Scalability risks—like the system slowing down under heavy use—are being managed by optimizing the database and designing the system so it can be scaled horizontally.

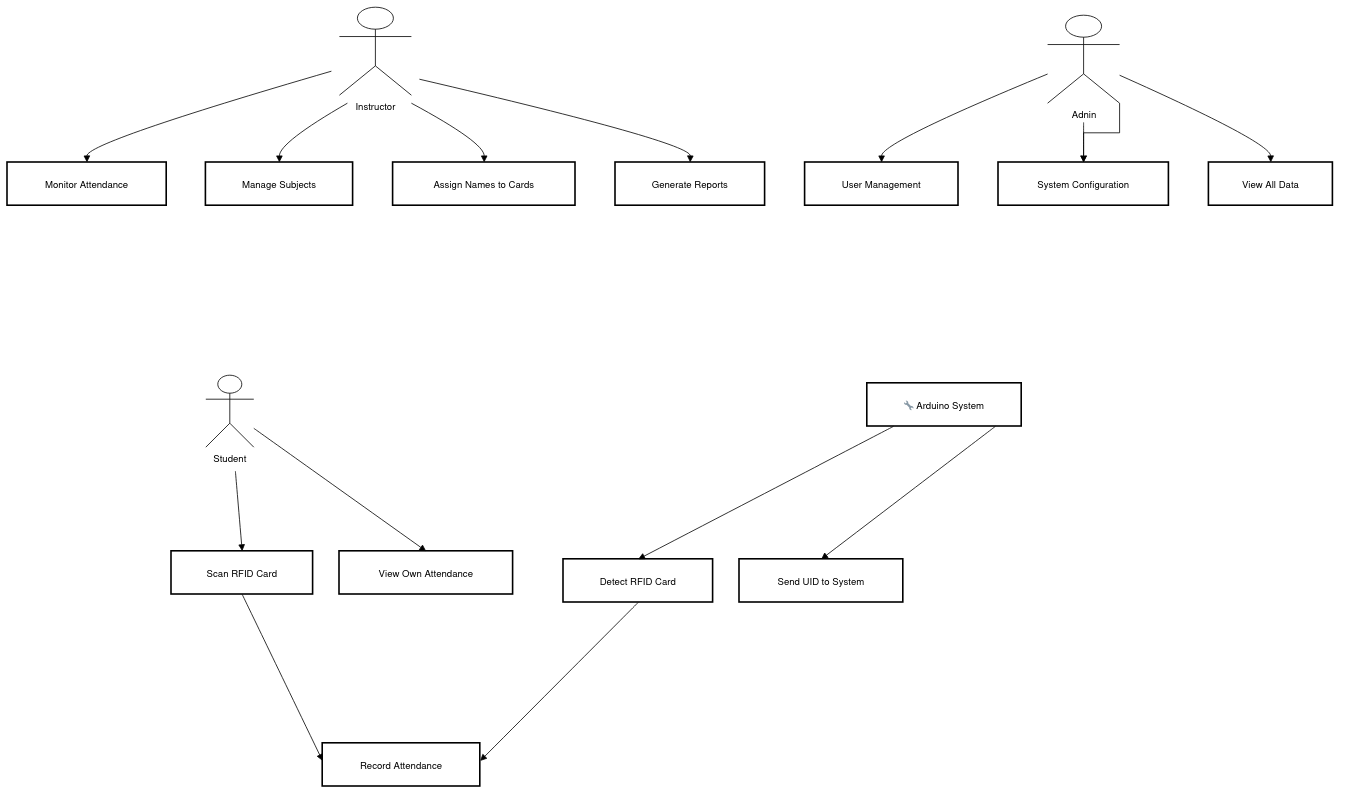
# APPLICATION DESIGN

This chapter deals with functional, structural and behavioral modeling of the RFID-based attendance system through the use of diagrams. The following diagrams provide visualization of system architecture, user interactions, data relationships, and process flows that guide the implementation of the attendance management system.

## Use Case Diagram and System Actors

The use case diagram gives an overview of the main interactions between different types of users (or "actors") and the system itself. The primary actors are:

* **Students**, who use their RFID cards to mark attendance
* **Instructors**, who track attendance and manage subject-related information
* **Administrators**, who handle user roles and system settings
* **The Arduino system**, which reads RFID card data at the hardware level

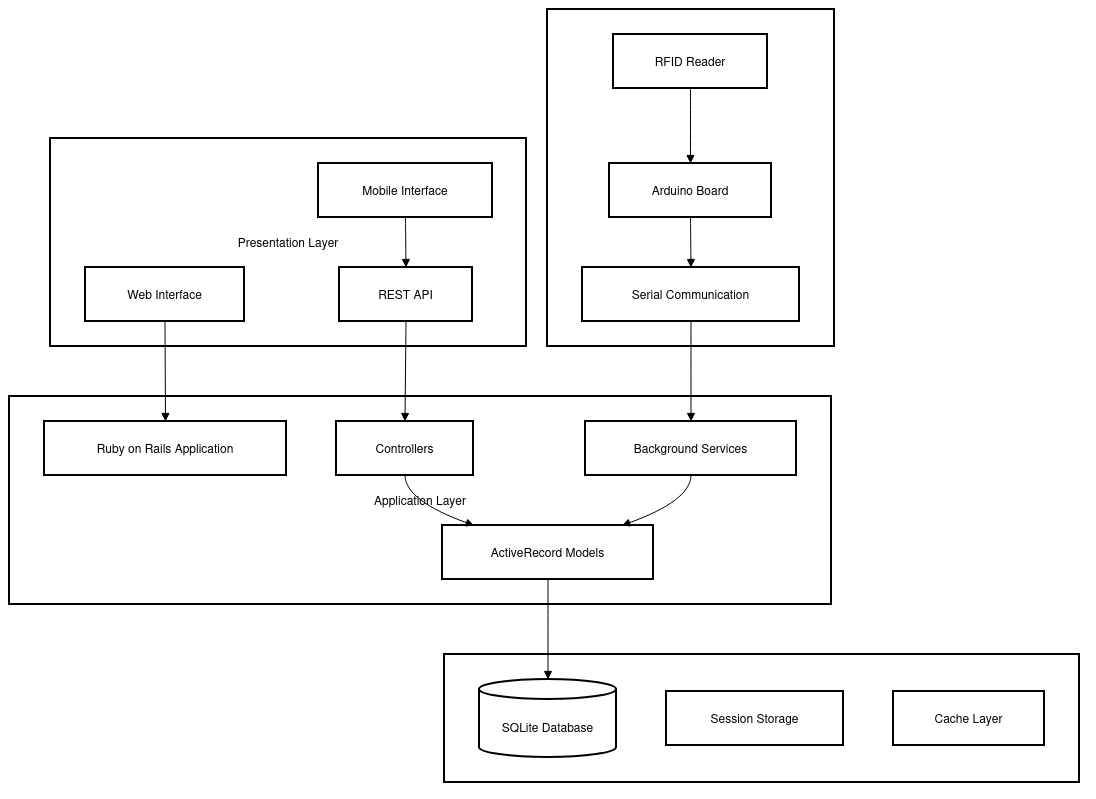


##### Figure 3.1 Use Case Diagram

The diagram highlights the core use cases such as scanning RFID cards, checking attendance, managing subjects, and configuring the system. It also shows how authentication and permission controls are in place to make sure only authorized users can access certain features

* 1. **System Architecture and Package Diagrams**

The system architecture diagram outlines how the system is designed using a three-layer model: the presentation layer (user interface), the application layer (Rails logic), and the data layer (database). It also shows how the Arduino hardware is connected to the rest of the system.



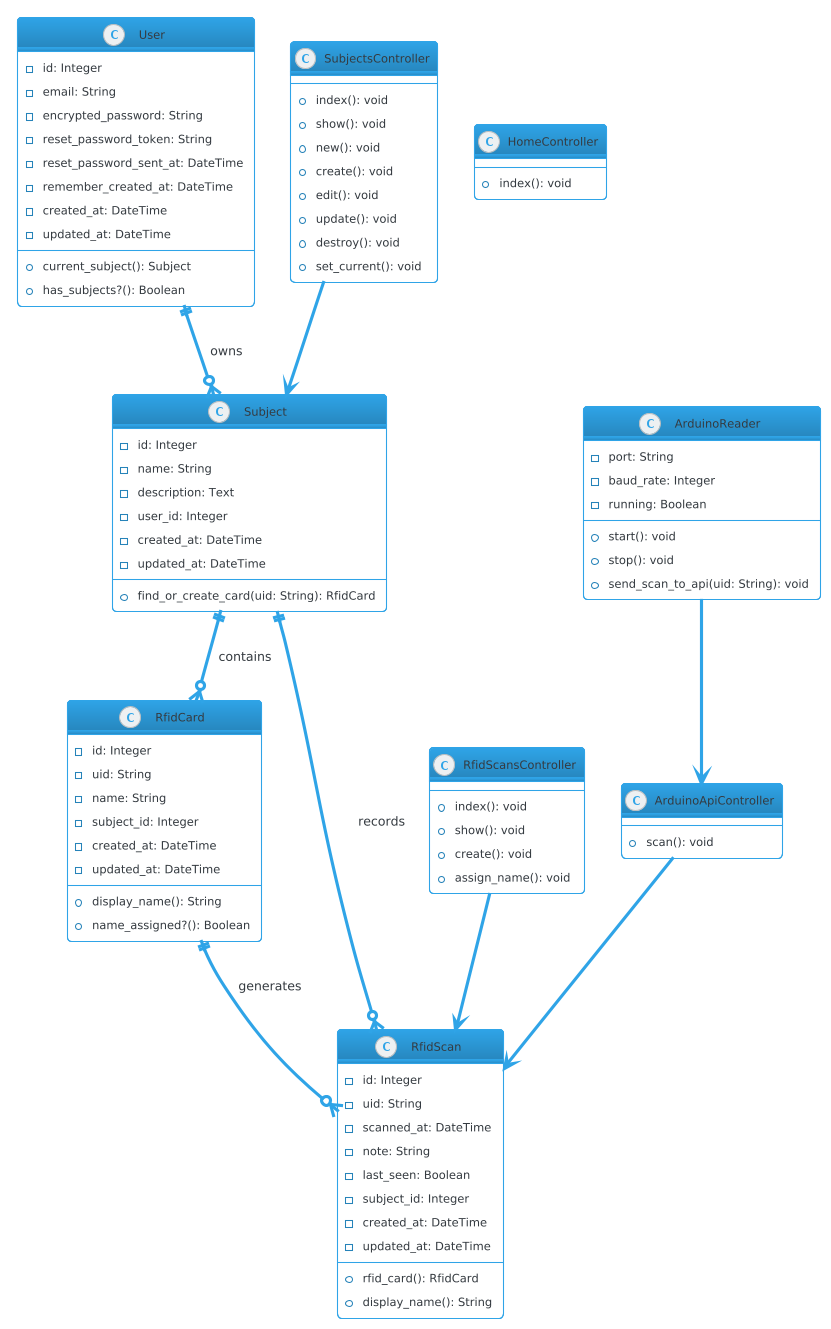
##### Figure 3.2 System architecture diagram

Package diagrams break the system down into its main parts, including controllers, models, views, and APIs. This structure helps keep the codebase organized and easier to update or expand later on.

* 1. **Class Diagram and Object Relationships**

The class diagram explains how the core parts of the Ruby on Rails app relate to each other. Main classes include User, Subject, RfidCard and RfidScan each with their own properties and methods.

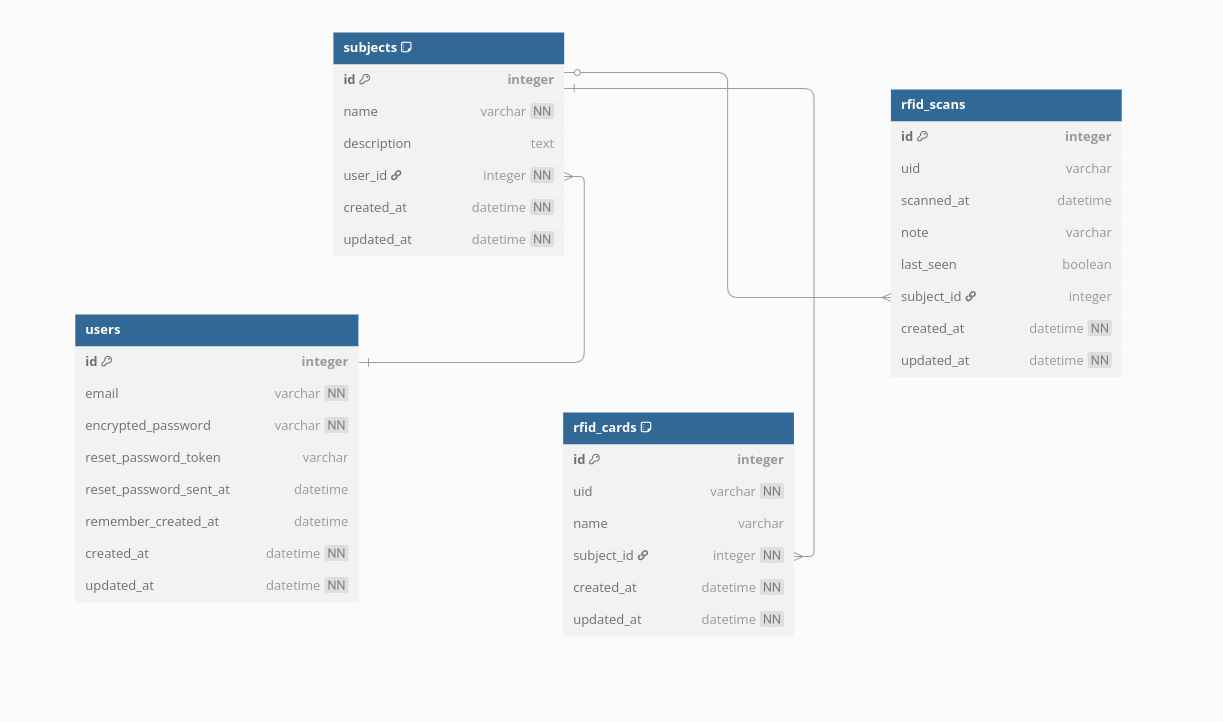
It shows how objects interact, including inheritance, associations (like one-to-many relationships), and dependencies. The diagram also helps illustrate how the MVC (Model-View-Controller) pattern is used in the application, especially how controllers work with model objects to handle user requests.



##### Figure 3.3 Class Diagram

* 1. **Entity Relationship Diagram and Database Design**

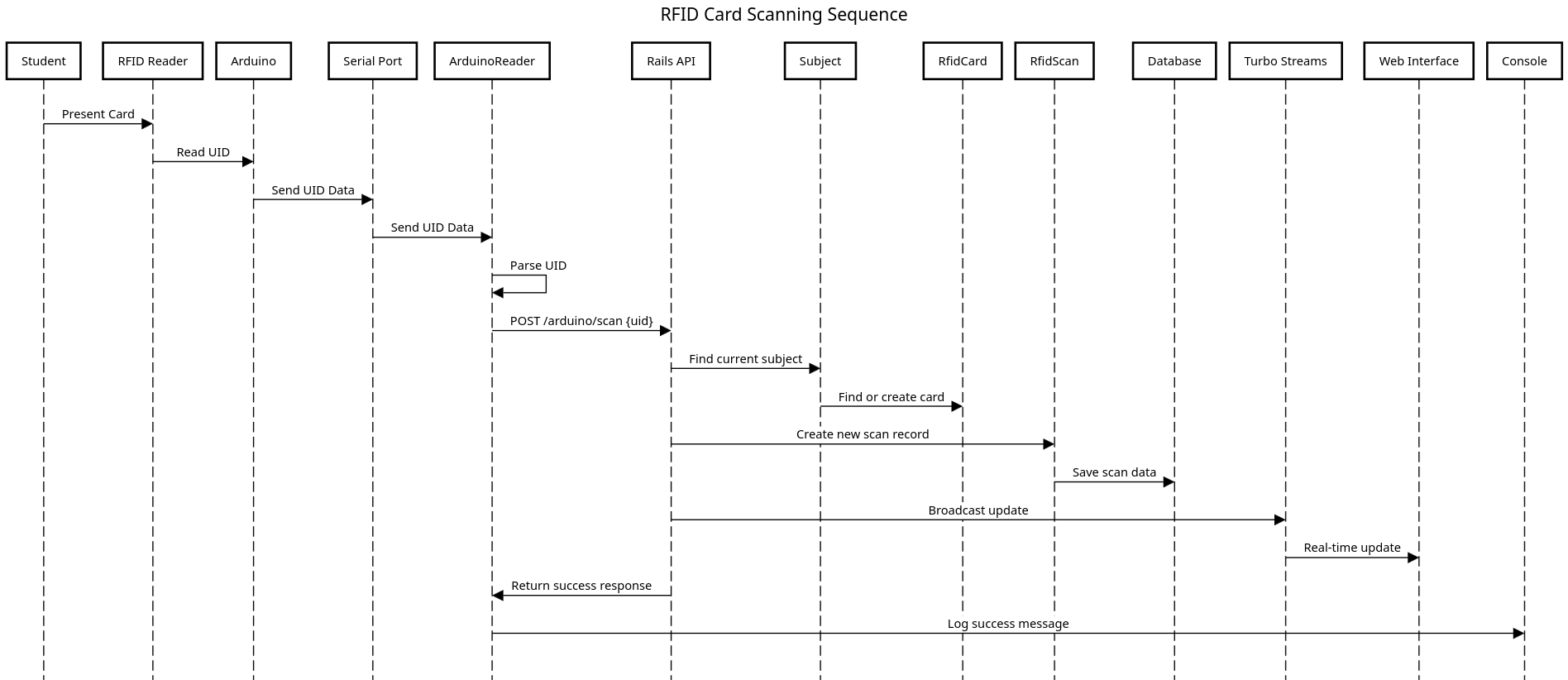
The entity relationship (ER) diagram maps out the system’s database. It defines the tables, primary keys, foreign keys, and constraints that help keep the data well-structured and consistent.



##### Figure 3.4 Entity relationship diagram

* 1. **Sequence Diagrams and Process Flows**

Sequence diagrams focus on the order of steps that happen during specific processes—especially the RFID scanning workflow. From the moment a student taps their card, the system records and updates data in real time.



##### Figure 3.5 Entity relationship diagram

This includes card detection by the Arduino, sending the card ID to the server, updating the database, and showing the result to the user right away. The diagram also includes what happens if something goes wrong, like retry mechanisms for better system reliability.

# Methodology

The methodology part focuses on the practical development of the RFID-based attendance system. It covers the choice of technologies, coding practices, hardware setup, and the way everything works together to build a functional and reliable solution. This section walks through how the system was created, configured, and made operational from writing code to setting up hardware.

* 1. **Development Environment and Technology stack**

For this project, Ruby on Rails 8.0 was used as the main web framework. It's known for its developer-friendly tools, fast prototyping, and solid structure, which makes it a good fit for education-related platforms. Rails uses the Model-View-Controller (MVC) pattern, which helps separate logic and keep the code clean and easy to manage.

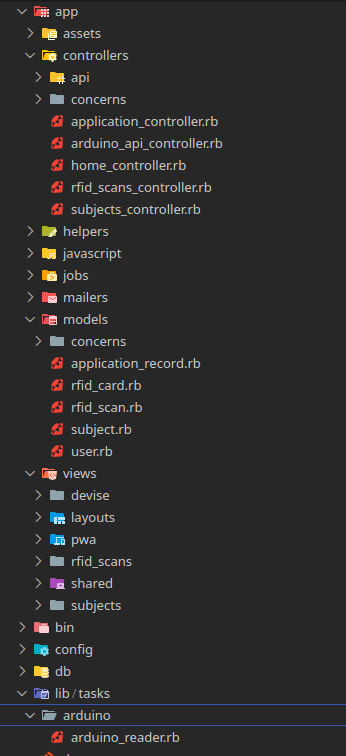
For the database, SQLite was used because it's lightweight and easy to set up. In the future the database can easily be migrated to PostgreSQL if needed. Rails’ built-in Active Record handles the communication between the app and the database, including relationships, migrations, and queries.

User login and permissions are managed using the Devise gem. Devise offers secure login, password encryption, email verification, and session control—all important for protecting student data and access rights. To support real-time updates, like showing a new attendance scan right after a card is tapped, the system uses Turbo Streams and Action Cable (Rails' built-in WebSocket solution). This keeps the interface responsive without needing a bunch of complex JavaScript code.

* 1. **Ruby on Rails Application Structure**

The app is organized using Rails’ MVC pattern, with all the routes, controllers, models, and views neatly structured for easy maintenance and future updates. There’s also support for both the regular web interface and API-based communication.

* The User model uses Devise for authentication and has many subjects. This means each user can manage their own subjects and student data. Validations are in place to prevent incomplete or incorrect information from being saved.
* The Subject model represents a course or exam session. Each subject belongs to a user and can have multiple RFID cards and scans attached to it. It also includes helpful methods to simplify things like report generation.
* The RfidCard model stores info about student cards. Each card has a unique UID, and optionally a name. It includes logic to prevent duplicate entries and to help find or manage cards easily.
* The RfidScan model tracks every scan event. Each scan is tied to a subject, and includes a timestamp. It also supports Turbo Streams broadcasting, so the system can instantly update the user interface when a scan happens.



##### Figure 4.1 File Tree

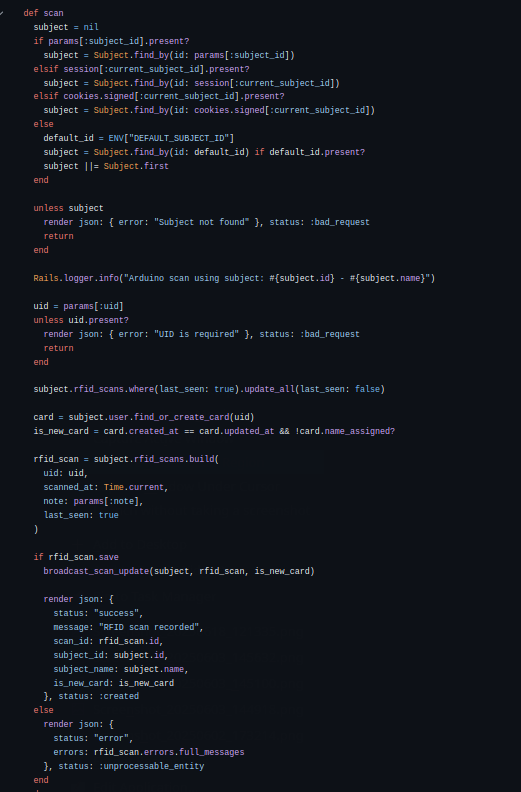
* 1. **Controller Implementation and API Design**

The RfidScansController handles most of the logic for viewing, adding, and managing scans. It supports searching and filtering, automatic card registration, and admin-level actions. The controller uses before\_action filters to make sure only authorized users can perform certain actions.

Error handling is also built-in, with clear messages when something goes wrong—either on the webpage or through the API.

The ArduinoApiController handles requests from the RFID scanner (Arduino). It uses simple authentication and is optimized for fast, repeated requests from the hardware. It also includes safeguards like rate limiting and input validation to keep things running smoothly.

Whenever a new scan happens, Turbo Streams sends live updates to the dashboard—like refreshing the attendance table, updating stats, or showing the latest scanned card. This keeps the system feeling fast and responsive

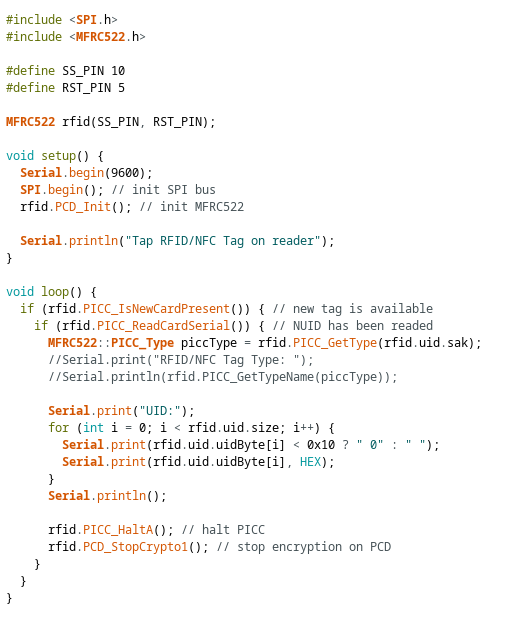


##### Figure 4.2 ArduinoApiController scan method

* 1. **Arduino Hardware Implementation**

The hardware setup includes an Arduino Uno microcontroller connected to an RFID-RC522 scanner module. The device connects to the usb port and sends scanned data to the Rails app using HTTP POST requests.

Cards are scanned using a loop that checks for a valid tag, extracts the UID, and verifies it. This prevents duplicate scans or incorrect data. The RFID-RC522 scanner module includes LED indicators to give feedback.

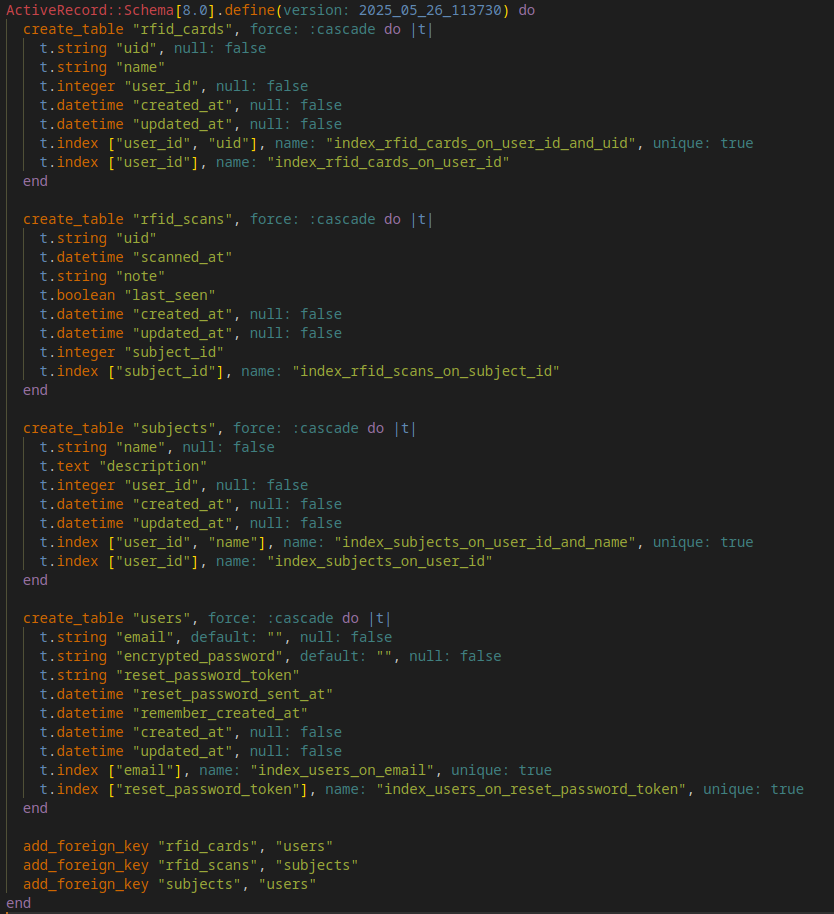


##### Figure 4.3 Arduino code snippet

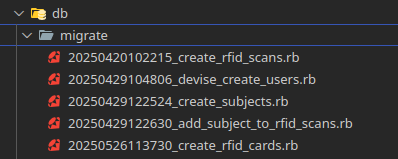
* 1. **Database Schema and Migration Implementation**

The database was created using Rails migrations, which define the structure and rules of the database step by step. This includes setting up relationships between tables, adding indexes for better performance, and applying validations.

* The users table includes Devise fields and extra columns like roles and permissions for different access levels.
* The subjects table links each subject to a user and ensures no duplicate subject names are allowed under one account.
* The rfid\_cards table uses a unique UID per card and links each card to a subject. Indexing helps quickly look up cards when a scan comes in.
* The rfid\_scans table tracks attendance events, including timestamps and related subjects. Extra fields like notes or flags can be used for status tracking.



##### Figure 4.4 Database schema.rb file code snippet



##### Figure 4.4 db/migrate folder showing database migrations

* 1. **User Interface Implementation**

The user interface was built using TailwindCSS. While developing the application I ensured that it has a modern design and that the UI includes clear navigation, simple forms and user-friendly layouts

The dashboard shows real-time updates using Turbo Streams. As soon as a card is scanned, the page refreshes the scan list, updates statistics, and shows system messages automatically.

Searching and filtering tools make it easier to manage large datasets. Users can search by name or UID.

Forms include validation to help users correct mistakes quickly. The interface is also accessible, making it usable with screen readers and other assistive technologies.

* 1. **Security Implementation and Data Protection**

Security is a top priority. The app uses Rails’ built-in features to prevent SQL injection, cross-site scripting (XSS), and CSRF attacks. Additional validation filters help ensure only clean and safe data enters the system.Sessions are managed securely with encrypted cookies, proper expiration rules, and CSRF protection to prevent unauthorized actions.

The API also includes rate limiting and authentication to avoid spam or abuse from external systems like the Arduino device.

Sensitive data is encrypted, and secure communication protocols are used for all network traffic. Backup strategies are in place to protect the data in case of system failure, and all security practices follow current standards for data privacy and compliance.

# SYSTEM TESTING

In this section, you should provide the results of testing conducted for your application, in the form of unit, integration and system tests. Additionally, you can include any other testing methods and frameworks that you used. Moreover, you can also discuss how testing results might impact further modifications of the project implementation.

* 1. **Testing Strategy and Framework**

To ensure that the system works as expected and remains reliable over time, the application includes a detailed testing strategy. It uses Ruby on Rails' built-in Minitest framework, which supports writing both simple unit tests and more advanced system tests.

The testing approach follows the testing pyramid model, which balances fast and focused tests with more complex end-to-end tests. The pyramid includes:

* Unit Tests – These test individual pieces of logic, such as model validations and associations.
* Integration Tests – These focus on testing how different parts of the system (e.g. controllers and routes) work together.
* System Tests – These simulate full user interactions in the browser to test the app end-to-end.

The main tools and libraries used in the testing process include:

* Minitest – Rails’ default testing framework, used for writing and running test cases.
* Capybara – Helps simulate user interactions in the browser during system tests.
* Mocha – Used for mocking and stubbing, so that we can simulate external systems without depending on them.
* Devise Test Helpers – These allow us to log in users during tests, which is essential for checking authentication and permission features.
  1. **Unit Testing**

Unit testing focuses on the models, which represent the core business logic of the application. Each model is tested for:

* Validations (e.g. presence of fields)
* Associations (e.g. has\_many, belongs\_to)
* Custom methods (e.g. searching, filtering)

This helps make sure that data stays consistent and that any future code changes don’t break the existing logic.

#### 

| **Model** | **Test Cases** | **Key areas tested** |
| --- | --- | --- |
| RfidCard | 8 tests | Validation, uniqueness, display methods |
| RfidScan | 7 tests | Associations, scopes, display logic |
| Subject | 6 tests | User associations, card relationships |
| User | 7 tests | Authentication, subject management |

#### Table 5.1. Model test coverage summary

test "should require unique uid per user" do

@card.save!

duplicate\_card = RfidCard.new(uid: @card.uid, user: @user)

assert\_not duplicate\_card.valid?

assert\_includes duplicate\_card.errors[:uid], "has already been taken"

end

test "should allow same uid for different users" do

@card.save!

other\_user = User.create!(email: "other@example.com", password: "password123")

other\_card = RfidCard.new(uid: @card.uid, user: other\_user)

assert other\_card.valid?

end

#### Figure 5.2. Code snippet for the RfidCard validation test

* 1. **Integration Testing**

Integration tests verify controller actions and API endpoints work correctly with the database and authentication system.

#### 

| **Controller** | **Test Cases** | **Key functionality** |
| --- | --- | --- |
| ArduinoApiController | 9 tests | RFID scanning, error handling, card creation |
| RfidScansController | 10 tests | CRUD operations, search, authentication |
| Arduino integration | 2 tests | End-to-end hardware simulation |

#### Table 5.1. Integration test coverage summary

test "should create scan with subject\_id parameter" do

assert\_difference("RfidScan.count", 1) do

post "/arduino/scan", params: {

uid: "ABCD1234",

subject\_id: @subject.id,

note: "Arduino test"

}

end

#### Figure 5.3. Code snippet for the Arduino API integration test

* 1. **System tests**

System tests use browser automation to test complete user workflows.

**System test coverage:**

* 4 tests
* creation
* editing
* deletion
* switching

test "creating a new subject" do

visit new\_subject\_path

fill\_in "Name", with: "Test Subject"

fill\_in "Description", with: "This is a test subject"

click\_on "Create Subject"

assert\_text "Subject was successfully created"

assert\_text "Test Subject"

assert\_text "This is a test subject"

end

#### Figure 5.2. Code snippet for the Subject management system test

* 1. **Test Results**
* **Total tests:** 47 test cases
* **Test Success Rate:** 100%
* **Coverage Areas:** Models, controllers, API endpoints, user workflows
* **Testing Framework:** Minitest with Capybara for system tests
  1. **Impact on Future Development**

The tests provide a solid foundation for future development through automated CI/CD practices

The project implements a GitHub Actions CI/CD workflow that automatically validates all code changes through multiple quality gates:

* **Security scanning:** Brakeman analysis automatically detects common Rails security vulnerabilities before code reaches productions and this ensures that the code remains secure
* **Code Quality:** RuboCop linting enforces consistent coding standards
* **Automated Testing:** All 47 test cases run automatically on every pull request and push to the main branch ensuring to regressions are introduced during development

**Future Development Benefits:**

* **Confident Refactoring**: If anyone wants to make changes to the existing code they can do so safely knowing that breaking changes will be caught immediately by the automated tests
* **Scalability Preparation:** The testing framework is structured to easily accommodate future enhancements

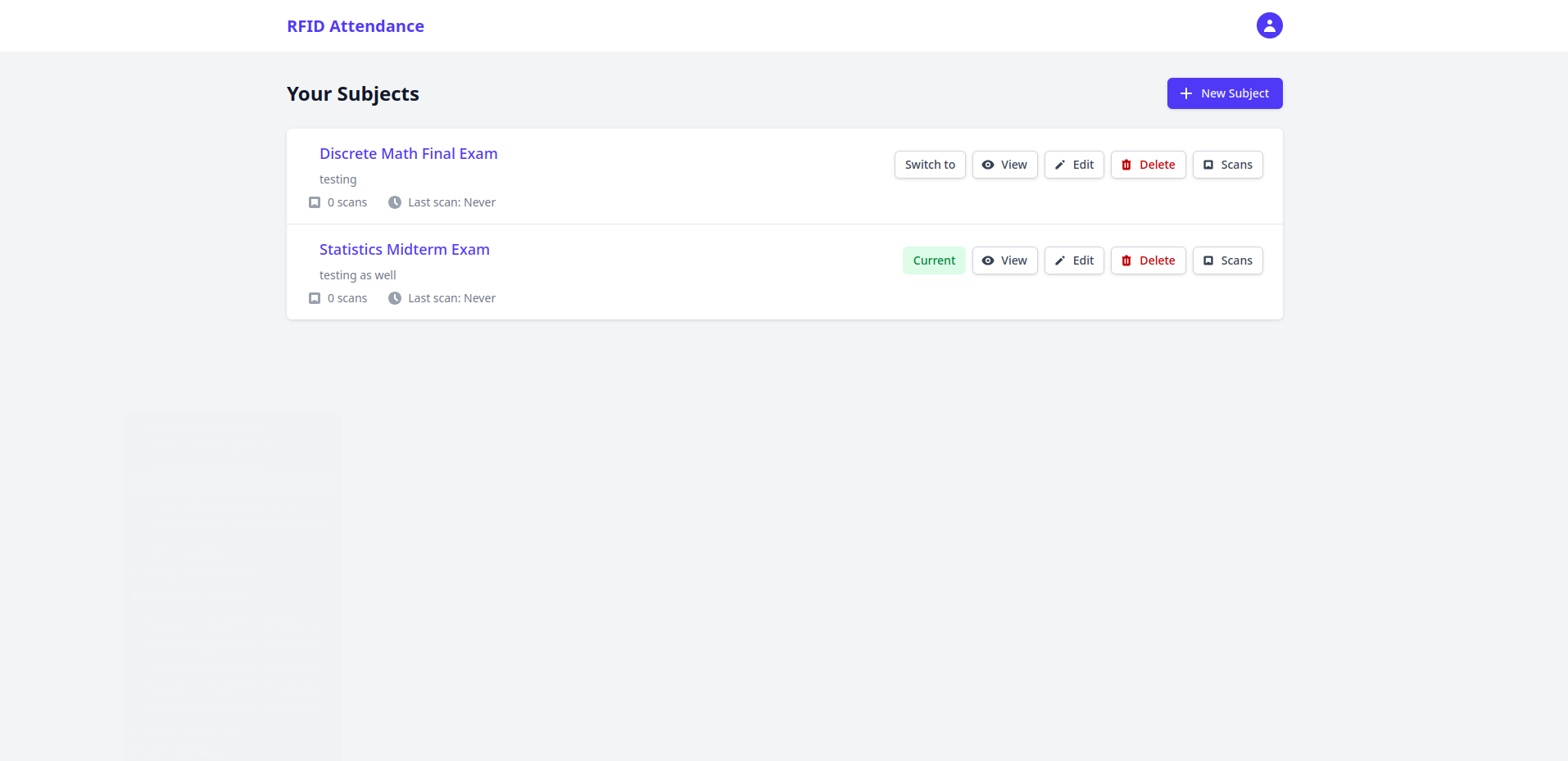
Overall, this testing and CI infrastructure transforms the application from a prototype into a production-ready system capable of supporting continuous improvement and feature expansion.

# Results

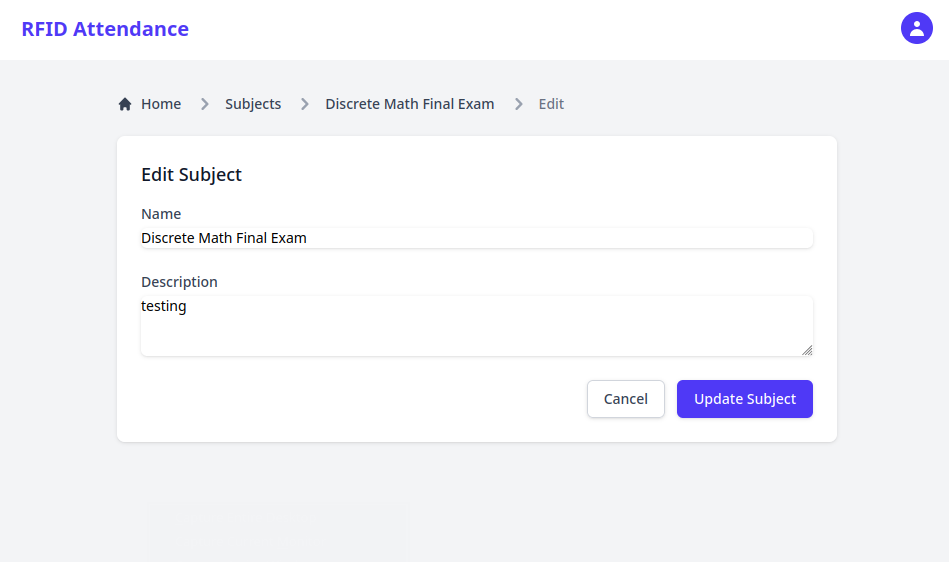
The RFID-based attendance system was successfully developed meeting all core requirements for a real-time, web-based attendance tracking solution tailored for educational institutions. It integrates RFID hardware (Arduino-based) with a Ruby on Rails web application, offering real-time data updates and user-friendly interfaces.

* 1. **Subject Management**

Once logged in users can create and manage multiple subjects such as classes or exam sessions with independent tracking for each



#### Figure 6.1. Main Dashboard

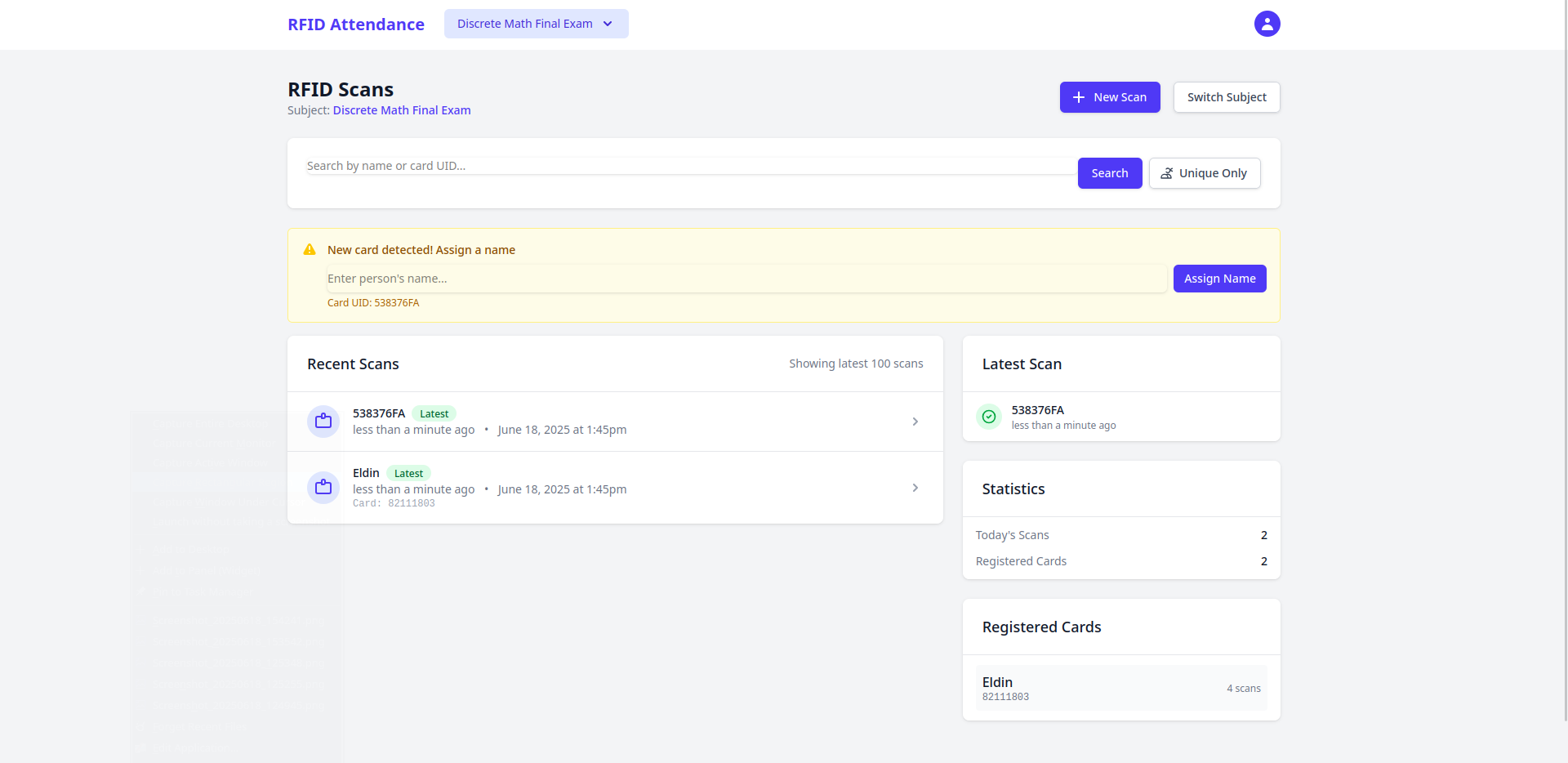


#### Figure 6.2. Subject edit form

Users can organize their courses clearly and manage attendance separately for each session.

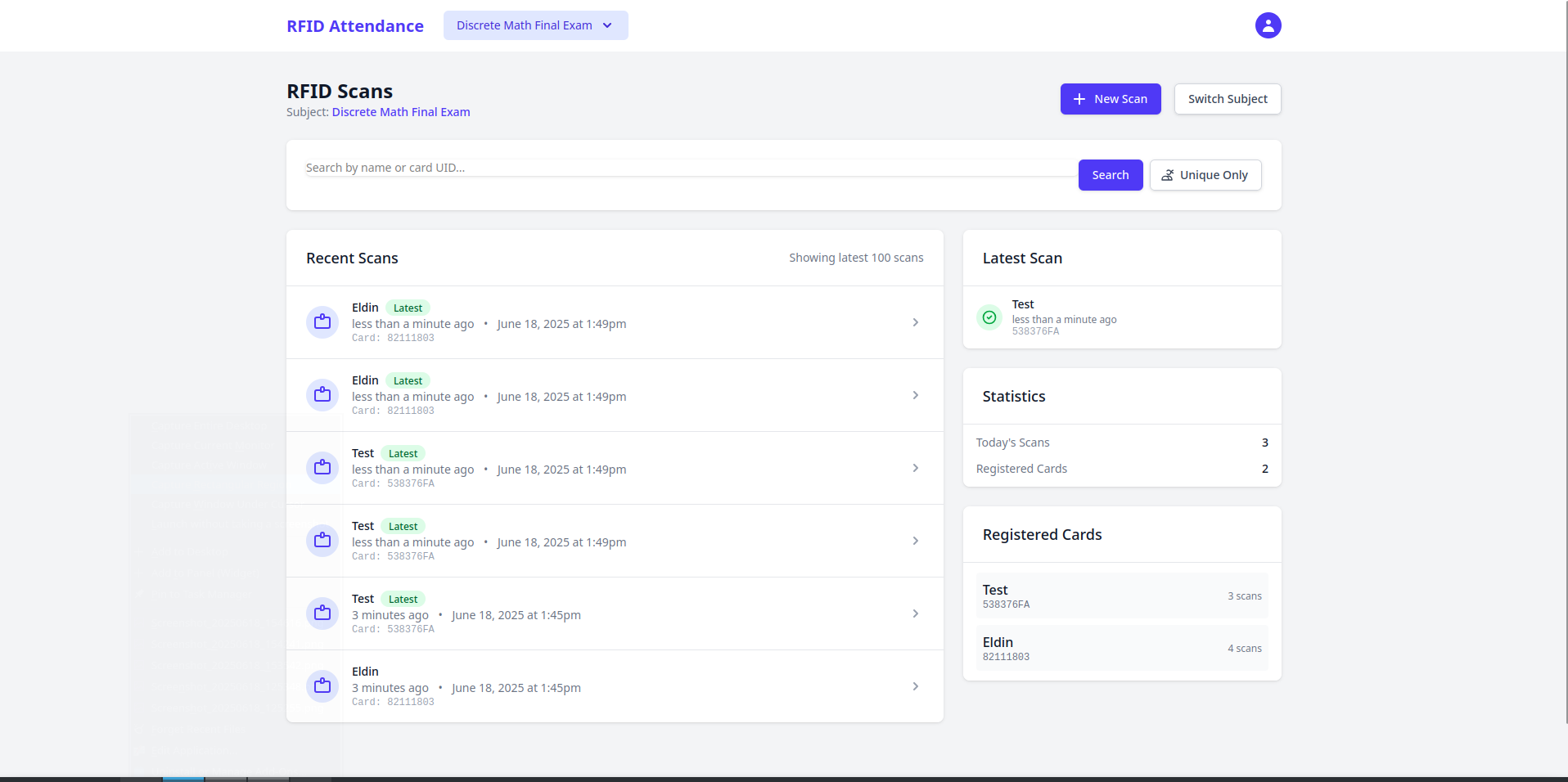
* 1. **RFID Scans**

The system automatically registers new cards when scanned, linking them to subjects and allowing optional name assignments

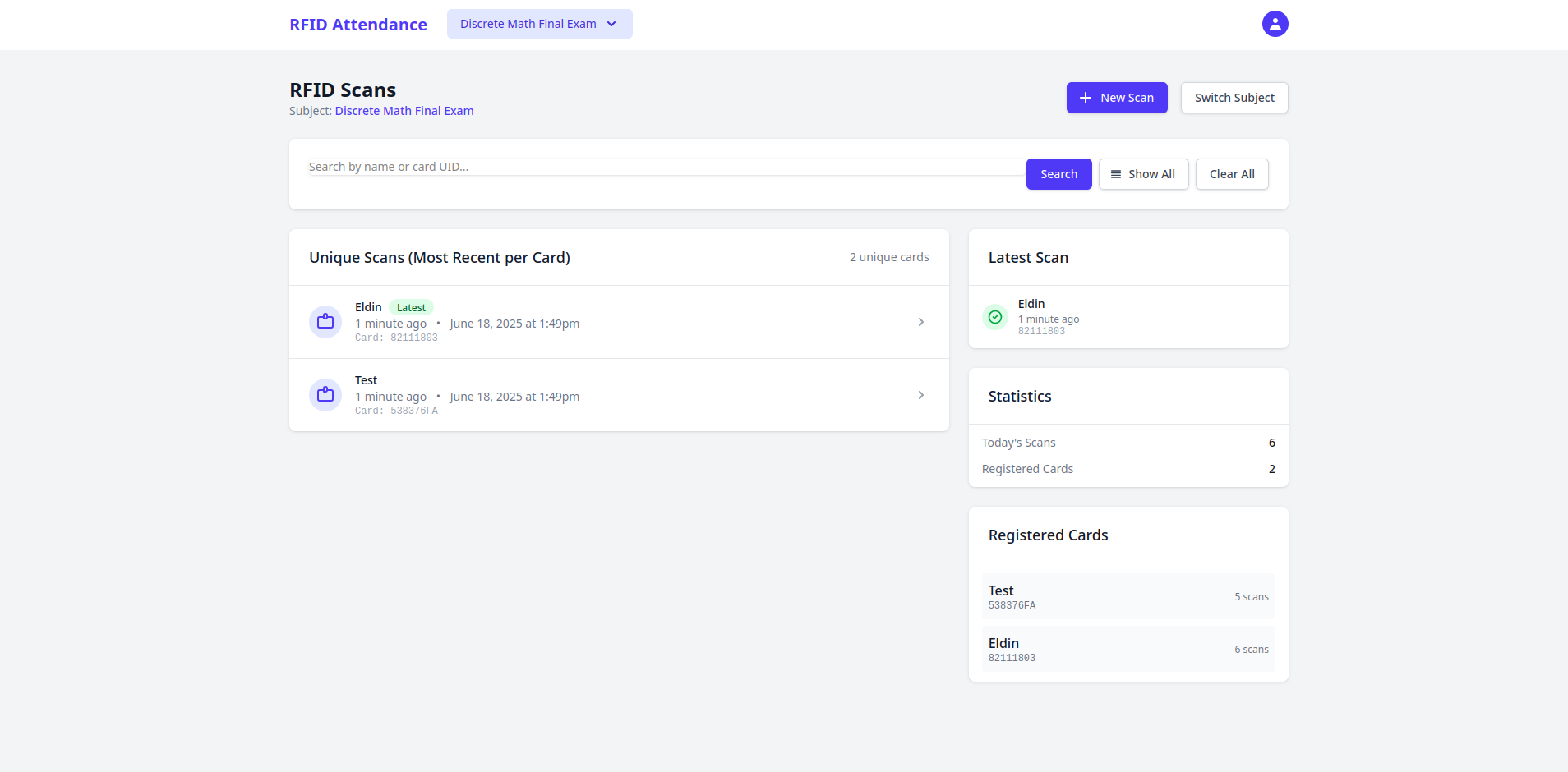
****

#### Figure 6.3. RFID scans page showcasing realtime updates and name assignment feature

Users can also filter cards to only show unique scans

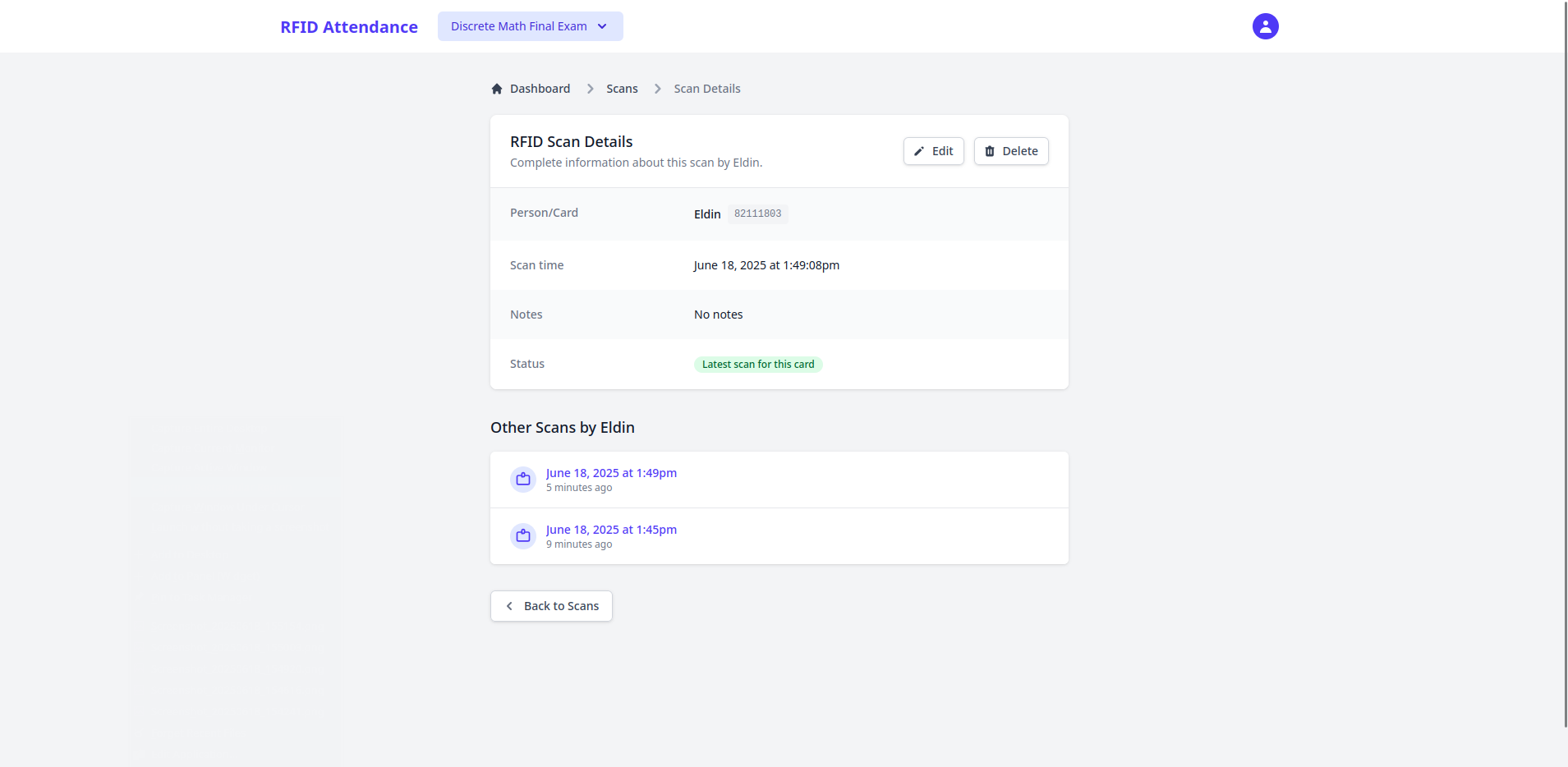


#### Figure 6.4. Scans without filtering showcasing all of the recent scans



#### Figure 6.5. Scans filtered showcasing only unique ones

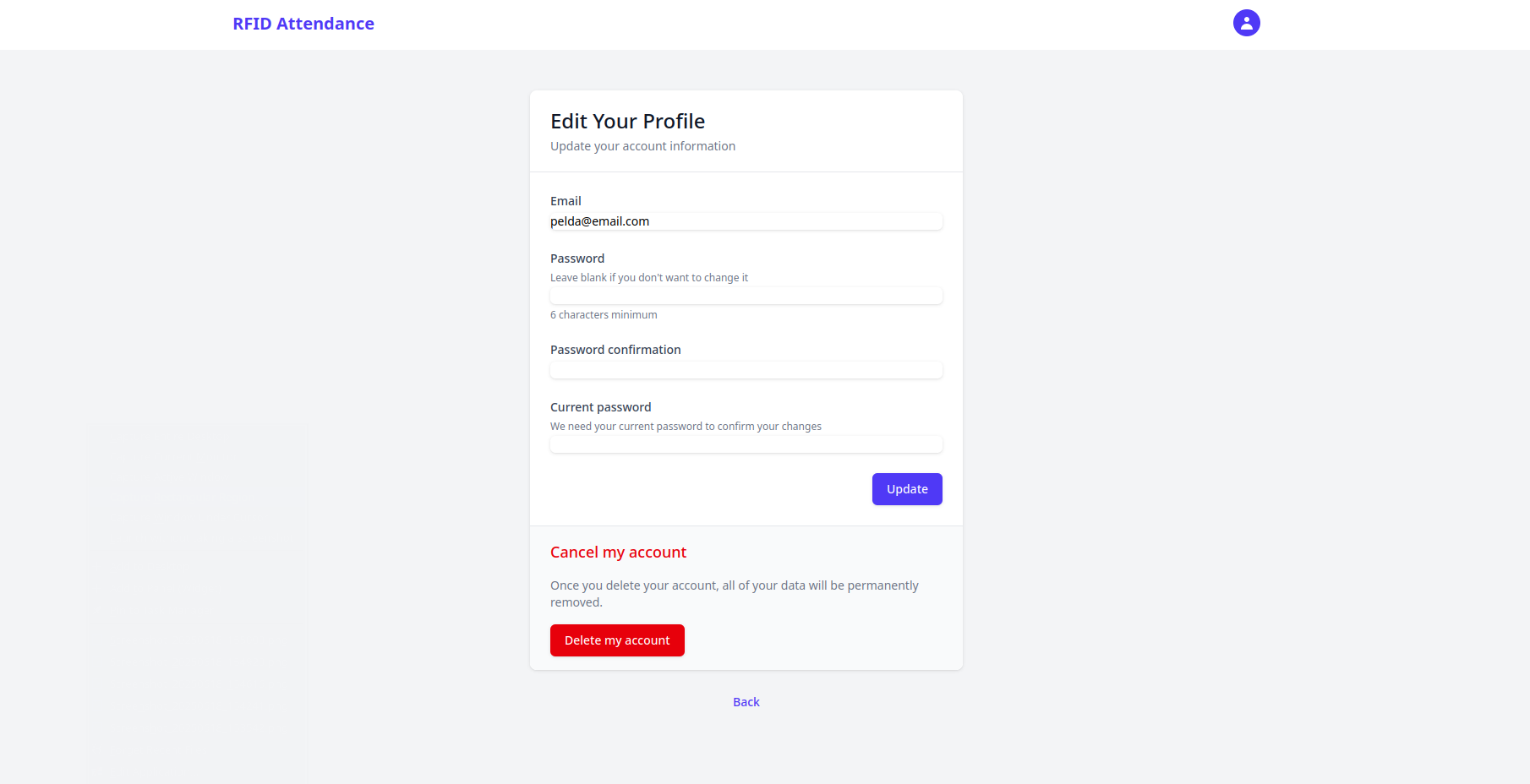
Users can see information about individual scans



#### Figure 6.6. Individual scan details page

* 1. **Profile Page**

Users can edit their account information if needed



#### Figure 6.6. Edit Profile page

# CONCLUSION

While thinking about an idea to do for a project I wanted to actually do something that would solve a real life problem and this is exactly what this project is about. It made it much more interesting knowing that this project could potentially, with some adjustments, be used in the future. One more thing that made it unique is that I connected a hardware component (Arduino with the rfid chip) with an external web application. Connecting those two was challenging since I needed to figure out a way to make the connection between those two stable and how to handle faulty scans. In the development process I used the real student cards that we have and luckily they have rfid chips inside of them but unfortunately they don’t return the student id but rather some different unique id and since I don’t know how the student data works internally and I did not have access to the student database I couldn’t do much there in a sense to show real student data upon scanning but this is something that could be built upon in the future if the opportunity to collaborate with the university presents itself. My initial project idea was to build a rfid tracking system which assists teaching staff (and that is built and it is functional and ready to use !) but what I am proud of is that this can also be used as a very solid foundation to build upon and make different kinds of rfid tracking systems.

**REFERENCES**

Ruby on Rails Core Team. (2025). *Ruby on Rails Guides*. Retrieved June 2025 from<https://guides.rubyonrails.org/>

Arduino Team. (2025). *Arduino Documentation*. Retrieved June 2025 from<https://www.arduino.cc/reference/en/>

Plataformatec. (2024). *Devise: Flexible Authentication Solution for Rails*. Retrieved June 2025 from<https://github.com/heartcombo/devise>

Hartl, M. (2022). *Ruby on Rails Tutorial: Learn Web Development with Rails* (7th ed.). Boston, MA: Addison-Wesley Professional.

TeamCapybara. (2024). *Capybara Documentation*. Retrieved June 2025 from<https://teamcapybara.github.io/capybara/>

Lin, J., & Wen, S. (2022). *Physical Education Attendance Based on Wireless and Passive RFID Technology*. Retrieved August 22, 2022.

Gaikwad, S., Abdare, P., Bansode, R., & Darkunde, R. (2024). *Smart Attendance System Using RFID*. Retrieved January 2024.

Choudhari, A. V., Nimbalkar, G. C., Potdar, O. M., Keskar, S. S., Patil, V. R., Umarani, S. S., & Potdar, S. B. (n.d.). *Attendance Management System Using RFID*.

Kumar, P., Raj, L. K., Karthik, M., Barker, C. S., & Manoj, R. (n.d.). *Enhancing Laboratory Experience: A Combination of RFID and Machine Learning Techniques to Track Attendance and Upgrade Hardware*.

Girish, H., Hemanthi, H., Bai, I., Jagruti, J., & Kumari, L. (n.d.). *Student Attendance Using RFID System*.

Shubham, V., Gaurav, G. K., & Singh, H. (n.d.). *Students Attendance Management System Using RFID and GSM*.

Vinodia, A., Mishra, A., Goyal, A. K., Iyer, D., Jani, H., & Modi, B. (n.d.). *RFID-Based Attendance System Using Arduino UNO*.

Kumbhar, S., Jindal, M., Gunjal, T., & Lokhande, S. S. (n.d.). *Smart Attendance System Using RFID and Arduino*.

Al Tarshia, A. R. K., Al Sadia, N. S. I., & Vimbia, V. (n.d.). *Attendance Tracking Using RFID and IoT*.

Ashok, S. H. V. K., Divya, N. D., Samuel, F., & Mande, U. (2022). *RFID Based Attendance System*. Retrieved December 2022.

Mukherjee, T., Mitra, A., Giri, L., Khan, S., & Datta, R. (2021). *RFID Based Attendance Management System*. Retrieved June 2021.

Sahoo, P. (n.d.). *A RFID Based Attendance Tracking System*.