ID Scanner Design Document

Clarkson University CS350

Spring 2025

Delara, Hunter, Samantha, Soliat, Nick

Contents

Project Overview & Objective:

Project Overview:

The ID Scanner Project aims to provide a fast, and easy scan-in/out system for the MakerSpace, replacing the existing inefficient Google Form with a simple hardware solution. Currently, visitors are required to fill out a Google Form that includes the machines that they intend to use during their visit. Additionally, if a student has not been to the MakerSpace before, they are also required to fill out a short waiver in the form of another Google Form. With our system, students will be able to simply scan-in/out their student ID cards at an RFID scanner. For the moment, scanning alone is sufficient for entry. At check-out, students will scan their ID again, and the screen will prompt them to select what machines they used during their visit. This will be achieved through a custom interface rather than a Google Form to improve simplicity, and will also capture more accurate usage data, as some students may change their mind about which machines they want to use after signing in. There will also be a basic Admin Dashboard (MVP), in which the admin could view event logs and export usage reports in CSV format.

Objectives:

- Improve the Makerspace scan-in process by using an RFID scanner included with a touchscreen interface.
- Improve the accuracy of machine usage records.
- Enhance user experience by integrating waiver submission directly into the system.
- Provide data export and analytics

Future Objectives:

- Integrate with university systems other than Makerspace (auto-fetch the student name/email address)
- Real-time display of machine or room usage (crowd meter)
- If they are new, the screen will prompt them to fill out a waiver, which will then be stored in a database that will remember they have completed it.

Identification of Modules

This project consists of four modules.

I. **Hardware Interface Module**
   A. **Function**: Communicate directly with the RFID scanner and touchscreen
   B. **Responsibility**: Detect RFID scans from the passed scanned ID and display UI accordingly

II. **User Interface (Front-end) Module**
   A. **Function**: Display a simple form with checkboxes for equipment selection and additional prompts for user interaction
   B. **Responsibility**:
      1. For new users: Prompt to waiver completion
      2. For returning users: Confirm entry or prompt for machine selection during checkout

III. **Back-end & Database Module**
   A. **Function**: Store and manage all usage information
   B. **Database Management System**: SQLite
   C. **Data Handling**: Use Flask API endpoint that receives scan-in data and user details
   D. **Export Data**: Implement a function that queries the database and formats the data as CSV

IV. **Admin Dashboard Module (MVP**
   A. **Function**: Provide monitoring and data export
   B. **Responsibilities**:
      1. Secure admin access (authentication & authorization) to view statistics/data
      2. View check-in/check-out logs
      3. Export reports as CSV files

Module Interaction (Integration Strategy)

I. RFID Scan Event

A. When an RFID card is scanned, the RFID module detects the card, sends the data to a Python script to read the UID.

II. User Routing

Flask app checks database:

A. New user would prompt waiver

B. Returning user prompts the entry log

C. For checkout, prompt the machine selection screen

III. Data Handling

A. The UI gathers inputs and sends the data to the back-end via an API call

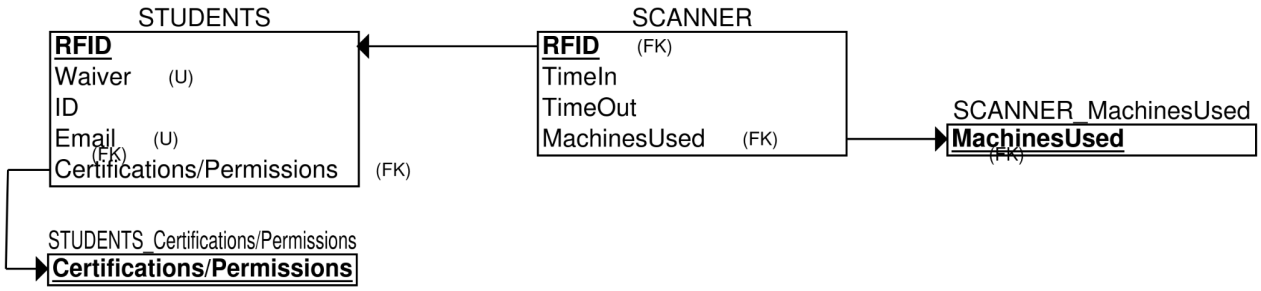
B. Data is stored in SQLite DB

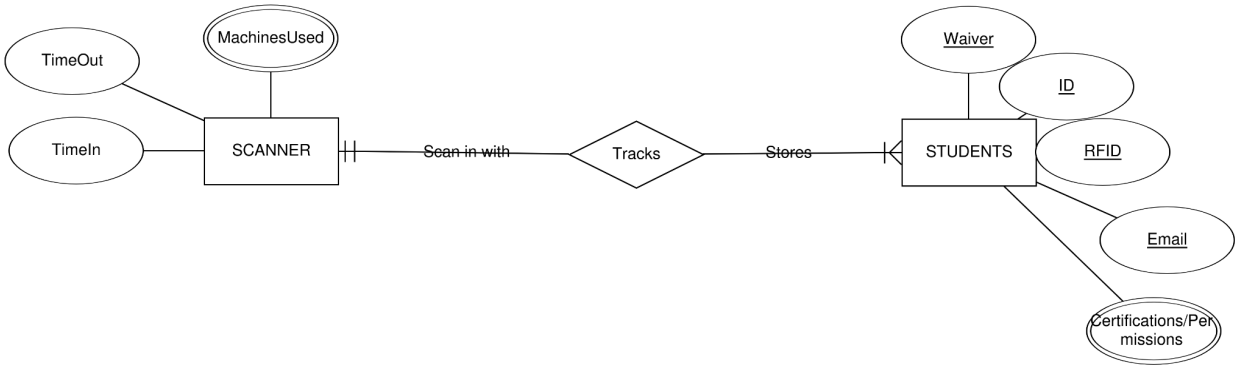
IV. Data Export & Analytics/ Admin Access

A. An export function allows CSV generation from the stored data

B. Role-based restriction in place

Database Schema



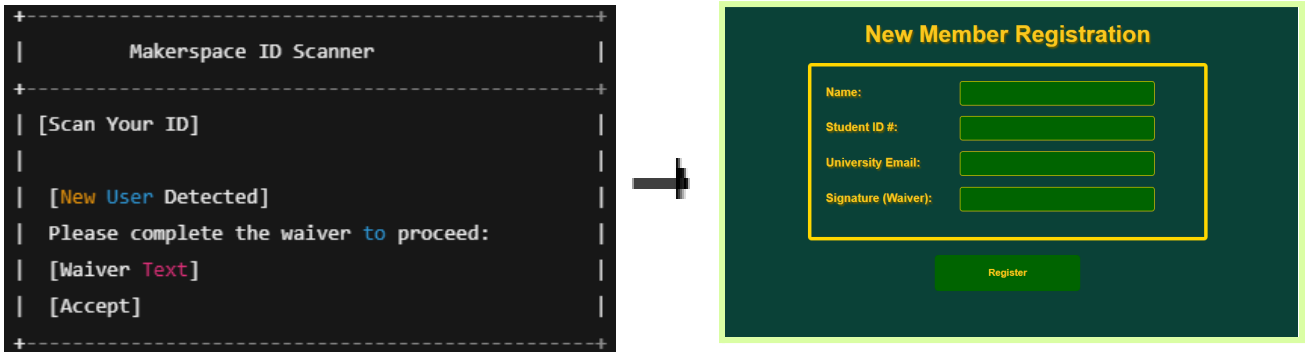


UI & Interaction Diagrams

User Screens

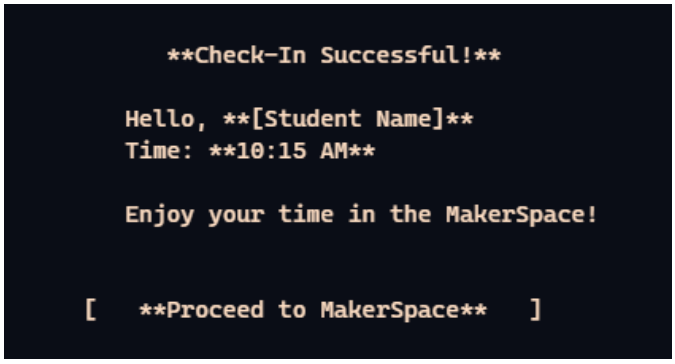
New User Example:

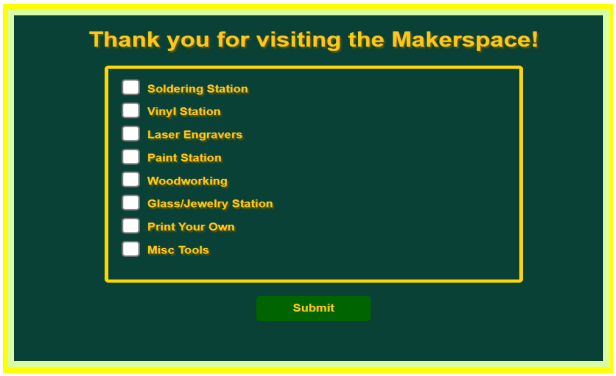
1. Scan ID → "Sign Waiver" form (name, email, checkbox)
2. Register → Access granted.



Returning User Example:

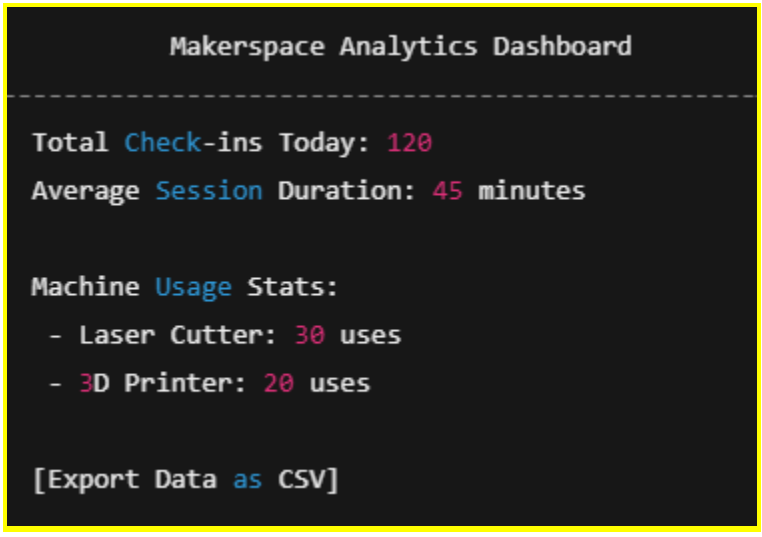
1. Scan ID → "Welcome back!" → Log entry time.
2. Scan-out → Select machines → Submit.





Admin Screens

Analytics Dashboard (Example)

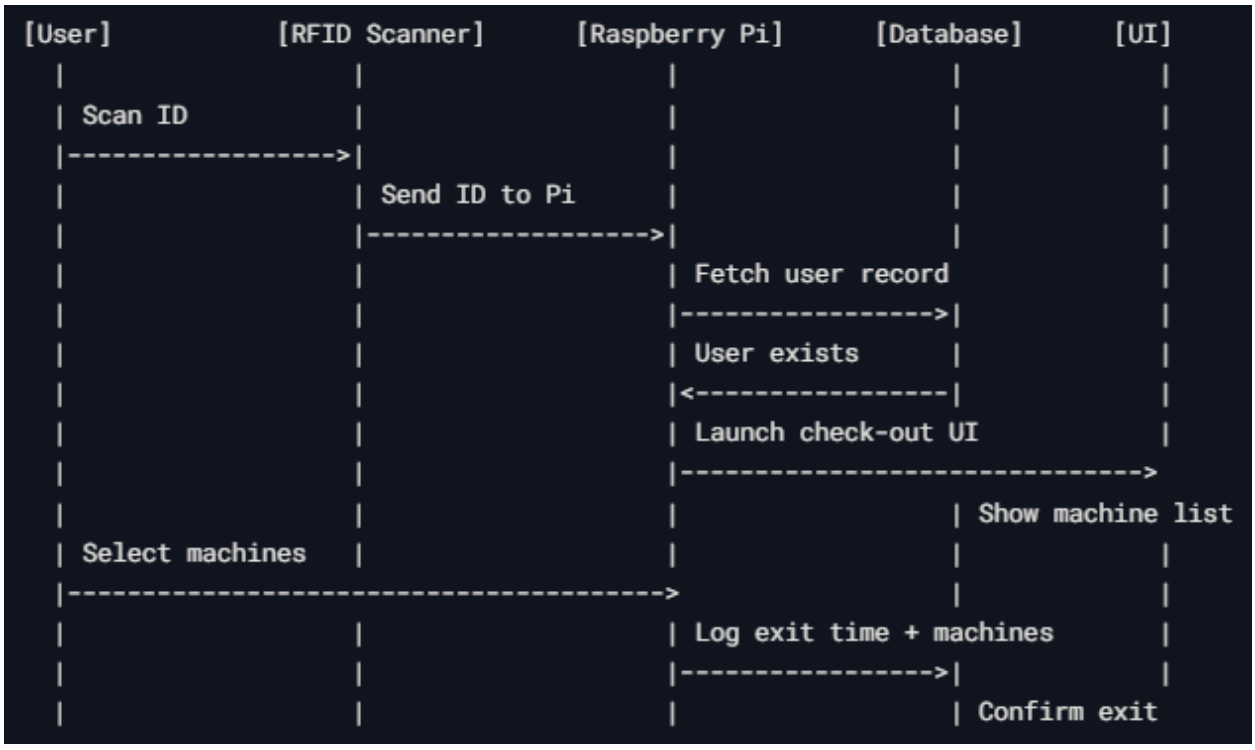


Sequence Diagrams

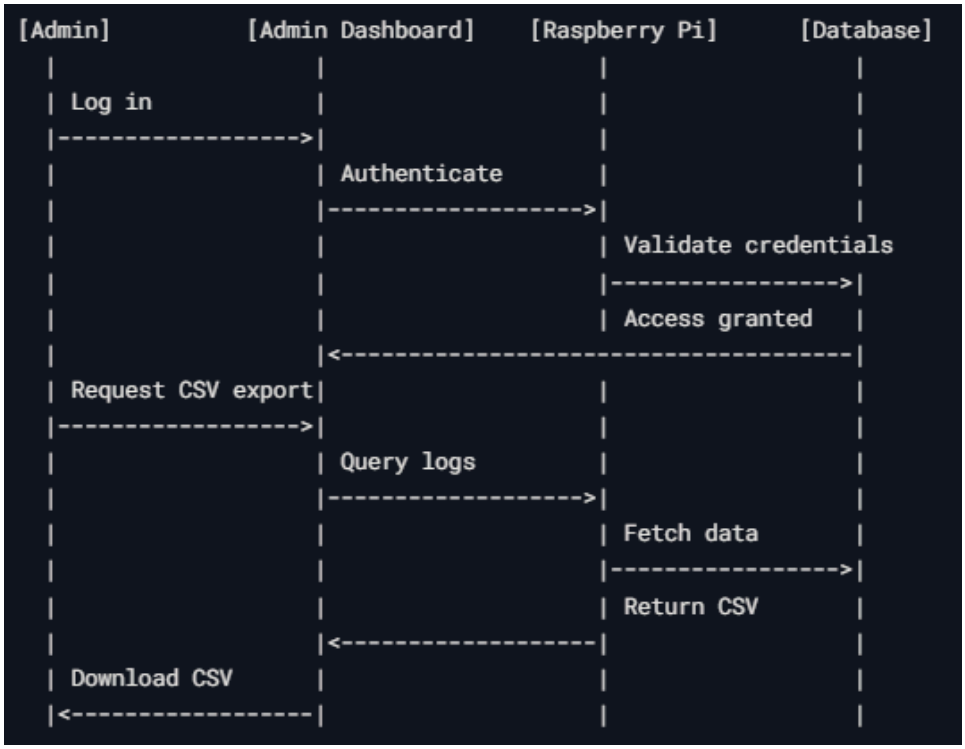
New User:

Returning User:

(Show interactions when scanning out as there is not much interaction when scanning in)



Admin:
(Shows interactions when an admin exports data)



Identification of Technologies

I. **Programming Languages**
   A. Python for hardware interaction, API, and backend logic
   B. HTML/CSS for frontend

II. **Frameworks**
   A. Flask for backend API and web serving
   B. Optional: Bootstrap for enhanced UI design

III. **Database**
   A. SQLite

IV. **Hardware**

A. USB RFID reader
B. Raspberry Pi 4

V. Additional Tools

A. GitHub, SSH for remote access

Security & Access Control

For the web-based user UI, security will be implemented using a multi-layered approach. We will combine RFID-based authentication with role-based access control (RBAC) to ensure that only authorized users can check in and out or manage data. Regular users scan their ID to check in/out, with new users prompted to complete a waiver before access. Admins have additional permissions to manage data. Secure communication via HTTPS, input validation, and session timeouts prevent unauthorized access. Data is encrypted, and audit logs track all activity for security monitoring. It is best to stay away from admin UI because it would be messier and make it less secure to be able to access the data, outside people could SSH into the Pi and copy the data onto a random computer.

Connections to Other Systems

If time permits, we plan to develop an additional front-end web page that connects to our database to display live Makerspace usage data. This feature would include a simple “crowd meter” built with HTML/CSS, to display accurate live crowd activity. This is not our primary focus though, as we are prioritizing giving MakerSpace mentors a more efficient sign-in process.