SPL User Management System: Requirements Report

Team 1: Ben Crane, Evan Handy, Hunter Hamrick, Clara McGrew, Estlin Mendez, Kaden Ramer, & Renee Rickert

Table of Contents

Preface

The purpose of this project is to develop a user management system for controlling access permissions for users of the Student Projects Lab on Western Michigan University's Parkview campus. The system is used to control passage through a light fence into the lab's machine shop, which is connected to a tower light and alarm system, as well as several computer consoles. Access is granted to users who have passed the requisite safety training modules via permissions associated with their Bronco IDs.

Introduction

Allin Kahrl is the laboratory supervisor of the Student Projects Lab, located in room G-112 in Floyd Hall. Current users of the lab swipe their Bronco ID using the swipe reader at the door. From there they are free to enter the lab. The lab also has a lab attendant. The job of the attendant is to monitor who is in the lab. To make sure they are following safety procedures and have had the required training. Finally, Allin Kahrl is the lab administrator who oversees the attendants in completing their responsibilities.

The current system of users swiping for entry at the door is suboptimal. The main issue is that the swipe at the outside door is not controlled by the lab. There is no way to have a photo on record for who swipes either. There is also no easy way to associate lab training with users swipe ids. The solution which we will be implementing is overriding the swipe outside of the lab door in the hallway with a new swipe in the small entry area of the lab. The current door you swipe to enter will be open at all times. From there, a new swipe system will be installed with a light bar sensor and a card reader. This system will be connected to a computer in the attendants' room as well as the administrators' room. Users will have the ability to make an entry associated with their swipe ID. They will be photographed in the attendants' office and any lab safety training information can be filled out. Once the users account is created they will swipe at the new swipe. From there they may walk through the laser sensor to register entry into the main lab where strict safety requirements are in place.

Through this new system the attendant will be able to have a live count of all the users who have been swiped into the lab. Associated with each user is a headshot of them and the machines they are safe to use. The system is going to be installed completely offline and disconnected from the WMU swipe system. This is to ensure system reliability.

Glossary

- Bronco ID – A photo ID card issued to all students, staff, and faculty at WMU.
- Console – A computer station with keyboard, mouse, and monitor that allows interacting with system.
- Light fence or optical laser gate – A beam of light connecting a laser emitter and detector that transmits a signal when something passes between the emitter and the detector.
- Tower light – A series of lights (usually of different colors) stacked on top of each other in a tower, used to communicate rudimentary signals via turning on and off particular colors.
- UI/GUI – User interface/Graphical User Interface – The point of communication between human and computer, particularly a visual display on a monitor that a user can manipulate by interacting with different elements via keyboard or mouse input.
- WIN – Western Identification Number, a nine digit number that is unique for each student, staff, or faculty member at WMU.

User requirements

There are three types of users who will be expected to use this system: regular lab users (students and others permitted access to the lab), attendants, and administrators. The system should only be accessible from the consoles in the lab, with no part of it accessible via the internet or even on campus wireless access, therefore no risk of exposing the personal information of users.

Lab users

Already existing lab users can expect to interface with the system via a swipe entry in front of an optical laser gate with a tower light and a small screen attached. These users will be able to swipe for access with the Bronco ID. If they are approved for lab access, the tower light will turn green to indicate that they are permitted entry, at which point they will have 10 seconds to pass through the gate before the light turns red again. The attached screen will display the user's name, photo, and a summary of which tools in the lab the user is permitted to use.

If a Bronco ID is swiped that is not permitted for lab access, the tower light will turn red and the screen will display a message saying the user does not have permission to enter and must see an attendant. On the exiting side, a user will need to swipe their Bronco ID again before leaving the lab, again waiting for the tower light to turn green to indicate they can pass through the light gate.

Users who attempt to enter or exit the lab without swiping, after the tower light has turned red, or after their 10 seconds to enter have expired, will cause an alarm to sound and an alert to display for the lab attendant and administrator to let them know of unauthorized access to the lab. The light gate will not physically stop users from entering or exiting the lab, only play a sound and display an alert, so as to prevent obstructing movement in the event of an emergency.

If a user is new and has done the requisite training to use the lab but has not yet been added to the system, they will need to see an attendant to be added to the system. That process is further described in the next section.

Attendants

Attendants will be able to do everything standard lab users can do with regard to passing through the light gate, but they will also have access to an additional console with attached webcam in the attendants' office. An attendant will gain access to this console by swiping their Bronco ID and entering their password to log into the system.

Once logged in, the attendant will have a display of the names, photos, and approved equipment of all the currently existing users, as well as a count of all the users currently swiped into the lab. This display will also be viewable on a screen inside the lab so attendants can view this information without entering their office. Attendants can click on a specific user to see any notes left about that user by other attendants, leave new notes about the user, or edit information about the user. When someone passes through the light gate without swiping in or without having approved access, an alert will display on the attendant's screen to make them aware of the unauthorized access.

Attendants will also be able to click on a button to add new users to the system, having the new user swipe their Bronco ID, then manually enter their name and approved equipment and take a photo with the attached webcam to add to the user's account.

Administrator

The final and most powerful user type is the administrator. The administrator will have their own console in the administrator office, with the ability to swipe in using their Bronco ID and password, similar to the attendants' console. The administrator can see the same displays as the attendants and similarly be able to add new users to the system. The administrator will have access to all of the attendants' notes on users, as well as a set of administrator-only notes, which attendants will not be able to view.

The administrator can also promote standard lab users to an attendant or administrator role, as well as demote attendants or administrators down to standard lab users. An administrator is not able to demote themself if there are no other users holding the administrator role.

Finally, the administrator can import an up-to-date list of user equipment certifications to the system by importing a downloaded report from Brightspace/Elearning, where users take training modules to be approved to use various lab tools and equipment, as well as manually edit tool permissions for individual users.

Non-Functional Requirements

Our system will be hooked up via a wired connection because we would like for the response from the system to be as close to instantaneous as possible when a student swipes in. The attendant and student should instantly know if they are allowed to access the lab or not. Along with this we want the attendant to be able to know which students are trained on which machines just by clicking on their profile. We also want to prevent students from double running the light fence to sneak into the lab when they are not supposed to be, however this will also rely on vigilance from the attendant.

As this is a student lab, we want the system to be quite scalable. As students graduate or leave the program the hope is that they are removed from the system to allow for new students to be added when needed with no extra issue. We also want the attendant or admin to be able to insert/update data in the database in a secure way. This will involve using SQL transactions, which will roll back a SQL operation if certain conditions are not met to maintain the integrity of the database. For example, the rollback can be used for error handling if the server crashes for some reason, and it can make sure that an admin or attendant does not insert duplicate data. Additionally, to maintain the privacy of the users, the system will not be connected to the internet or accessible via a wireless to connection, thus removing any ability for outside actors to gain access to any personally identifying information in the database.

Our general design will be basic, allowing for straightforward operation by both the attendant and student, and quick access to the lab. This will allow for a higher level of maintainability since we all will not be here to help maintain the system, or database. Because this is a stationary system, the location conditions are not subject to change, so allowing for portability or for extreme cases of cold or heat is not necessary.

In this system, sensitive information about a given student will not be accessed easily by unauthorized users. Only the lab attendants and administrators will have access to the database. There will be a log-in system in place for the attendants and administrators to access the database, and the administrators will be given special privileges to promote or demote a user. When a user swipes their card to access the lab successfully, a screen will show up that will only be visible to the attendant with the user's identifying information. After 10 seconds the screen with the person's identifying information will disappear, and the system will switch back to a welcome screen.

If a student has a problem swiping their card, the attendant will be given access to an error screen where they can log the student in manually, provided the student is authorized to use the lab. This will help to ensure that any issues that come up will be handled in an efficient manner.

Functional Requirements

This software will be used to control hardware including a light fence, tower lights, and auditory alarm system. This software will also write to a database that can be manually controlled by administrators in a terminal. Attendants will also have a limited ability to modify the database, to add new lab users and take lab users' photos. Appealing UI is not required, as this software is for logging students present in a room, and the only direct interaction students will have with this software is swiping a card for room access. There will be a screen in the office that will update to display live info about which students are in the lab, and what equipment they have access to.

Depending on severity of errors, a warning light or alarm will sound to indicate problems. All possible errors students could make, such as using the wrong type of card, swiping wrong, or crossing through the active light fence, will all be caught and handled. All attempts at entry, both authorized and unauthorized, will be logged.

System Requirements

This system is going to use the latest version of Python (3.13.0) with a set of libraries for software. Pandas will be used to work with data, possibly along with SQL Alchemy. SQL Alchemy is going to be used to interface with the SQL database, while MySQL will be used as the actual relational database management system. PyQT is the library that will be used to design the user interfaces on the consoles / displays. OpenCV is used for parsing the images that are taken to register student images with their account. The Swipe processor and event broker are going to be controlled by a Raspberry Pi Pico, which was chosen for both its versatility and its cost effectiveness. This board will be connected to the attendant and administrator consoles as well as all the card readers, displays, tower light, and light gate. Through this connection the various devices will be able to communicate with events. The specific models of card readers, light gate, and tower lights are not yet known, but will be determined by the client before programming begins in earnest. There will be several card readers in use: one on each side of the light fence, one attached to the attendant console, and one attached to the administrator console. When a Bronco ID is swiped with the card reader, the data of the swipe is paired with the associated WIN. This event is then sent to the swipe processor to check for the existence of the student and their permissions by querying the database. The administrator computer is where the database is stored.

System architecture

Our application uses event-driven architecture that consists of three major modules: event producers, event broker, and event consumers.

The event producers include the card reader module, gate module, attendant computer module, and the swipe processor. The card reader module can send either a “swipe in” or “swipe out” event with the user’s ID. The swipe processor can send either an “authorized” or “unauthorized” event. The gate module can send a “gate crossing” event. The attendant computer module can send “new user” or “edit user” event with the new data. The system will wait for one of these producers to create an event that can then be sent to the event broker.

The event broker determines where to send event signals, and the data they carry based on event type. Events, and their corresponding data are sent to all accepting consumer modules.

The event consumers accept events and react accordingly. Event consumers include the gate module, the gate screen module, the tower light module, the alarm module, the attendant screen module, the database module, and the swipe processor. The gate module accepts the “authorized” event type and allows the user to pass through the gate. The gate screen module accepts both the “authorized” and “unauthorized” event types and displays the user’s information on the gate screen. The tower light module accepts the “authorized”, “unauthorized”, and “gate crossing” event types, activating the green light for “authorized” and activating the red light for “unauthorized”, and “gate crossing”. The alarm module accepts the “gate crossing” event type and activates the alarm. The attendant screen module accepts the “authorized”, and “swipe out” event types updating the information displayed on the screen accordingly. The database module accepts the “new user” and “edit user” event types and updates the database accordingly. The swipe processor accepts the “swipe in” and “swipe out” event types and checks the database for the user. If no user is found or the user is not authorized to work in the school, it sends an “unauthorized” event, otherwise it sends an “authorized” event with the user’s data.

This design allows the application to react to events as they occur and respond accordingly. Additionally, due to the separation of the different modules, they can be developed individually. This architecture also has the benefit of easy scalability should the system need to be expanded in the future.

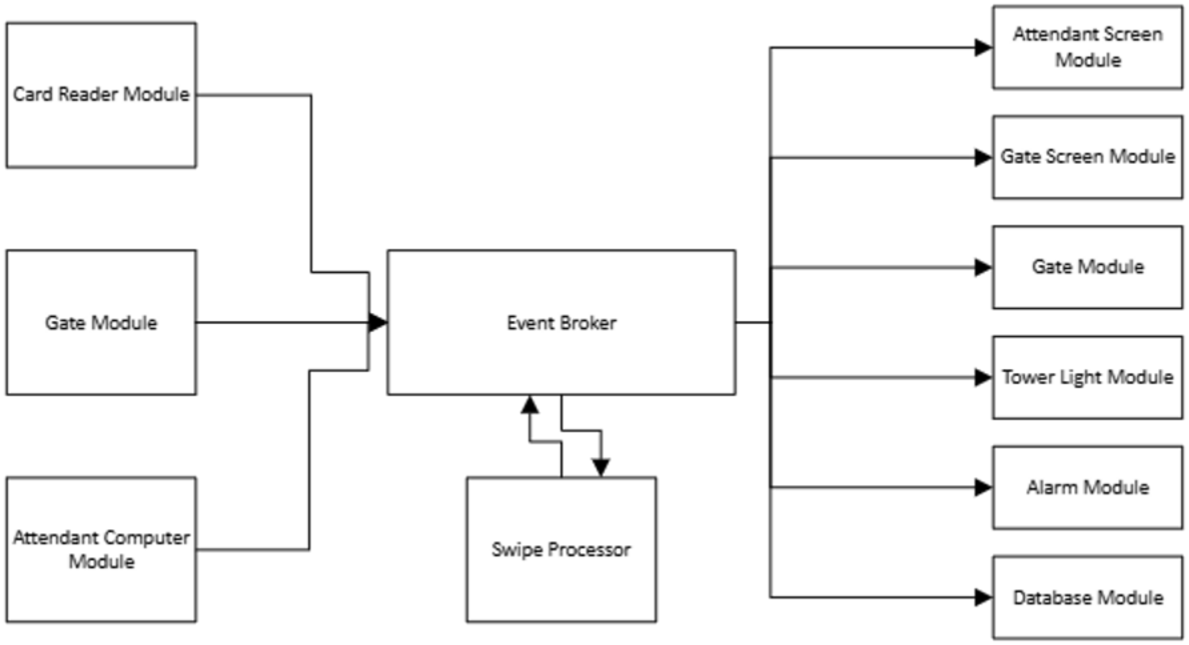


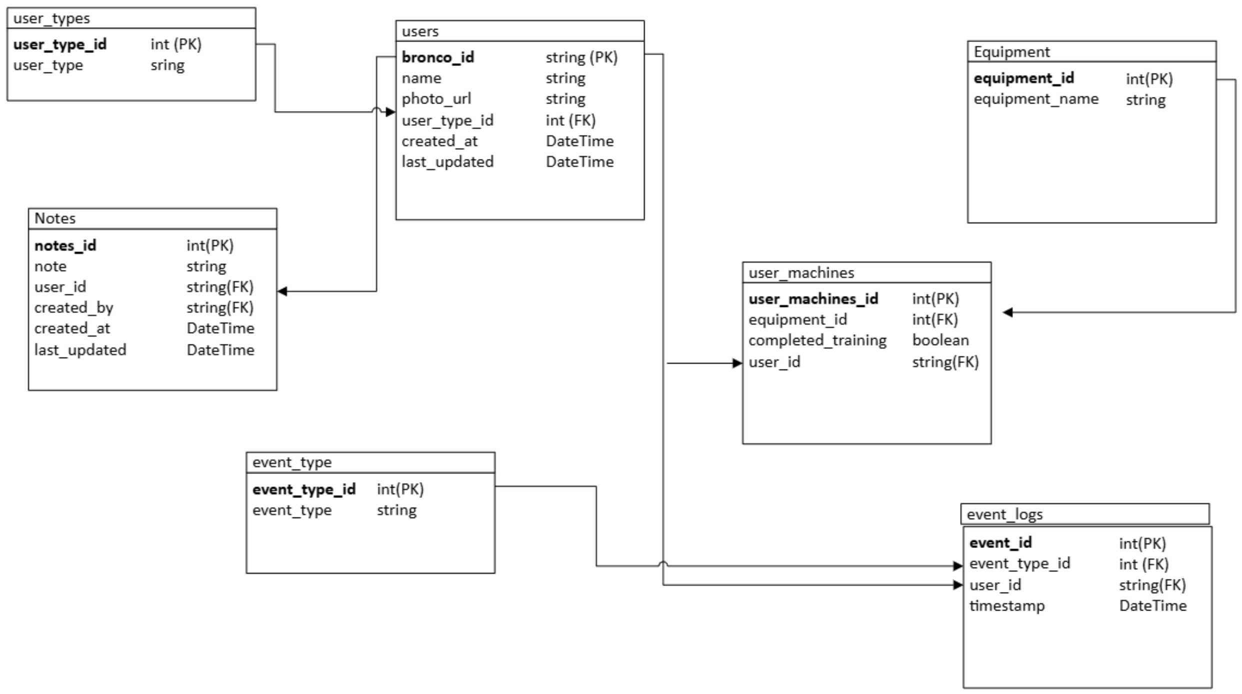
Figure 1: System architecture

System models

We are using an entity-relationship model for the database design, with primary keys (PK), and foreign keys (FK) for the relationships and with each table in the database representing a different entity. Above, we have the user_types, users, notes, equipment, event_logs, and event_types tables, and they are connected to each other via one-to-many relationships. For example, the user_types table and the users table have a one-to-many relationship because the user_type_id is a foreign key in the users table and can be used multiple times. The user_types table will have different user types such as “admin”, “attendant”, and “regular”, to indicate the kind of user a person is. Every time a new user is created, we will have their bronco id, as the primary key, followed by the name, photo_url, user_type_id, and so forth. Each user can have multiple machines, and there will be a boolean (completed_traning) value that will indicate whether or not a user has completed the required training for that machine. There will also be a notes table on file that an attendant/admin can access and create notes on a specific user. The different event types will represent card swipe events, as to whether or not a card swipe was successful or not. Every time a user swipes the card, the event_logs table will log the event that occurs. Below is a graphical representation of our database design. Lines between the tables represent the relationships.

Figure 2: Database design

We plan to use the SQL Alchemy Python library to work with the database. Its Object-Relational Mapper will enable us to create models based on the tables in



the database and work with them in an efficient way. We can use the SQL Alchemy library to create a “session” that will connect to the database and create queries that will insert, update, delete, and filter the data as needed.

System evolution

This system will be designed in a way that allows for easy maintenance and adjustment by any administrator with at least rudimentary programming skills. The program isn’t likely to need much if any maintenance, except removing users from the database to prevent it from growing too large. This will possible using the GUI or through traditional database modification via SQL statements.

Should future users wish to develop more features for the system or fix any incompatibilities that may come from operating system or hardware updates, the Python source code will be well commented and minimally complex to allow for future administrators to make desired changes with ease. Below are several feature updates that our team has envisioned the client possibly wanting in the future.

Custom Notifications

Notify the on-hand attendant or administrator of unusual activities like unauthorized swipes and prolonged stays. This would allow attendants/administrators to keep better track of who should be in the lab, as well as reduce the chance of injury for a user who has been in the lab for an extended period.

Automatic Training Verification

Interface with eLearning/Brightspace to automatically update approved equipment for users once they complete a certification. This would automate the process instead of having an administrator have to import that information into the system manually.

Machine Monitoring

Add IoT-enabled tools to track real-time usage of equipment, preventing misuse or overuse. This could interface with the notifications system as well, notifying attendants or administrators of unusual use of a certain piece of equipment.