1 Materials

to be filled

2 Methods

Device engineering

to be filled

RFID-based lock system algorithm

The locking system follows the algorithm depicted in the Figure below. It was programmed using its native `adruino` programming language.

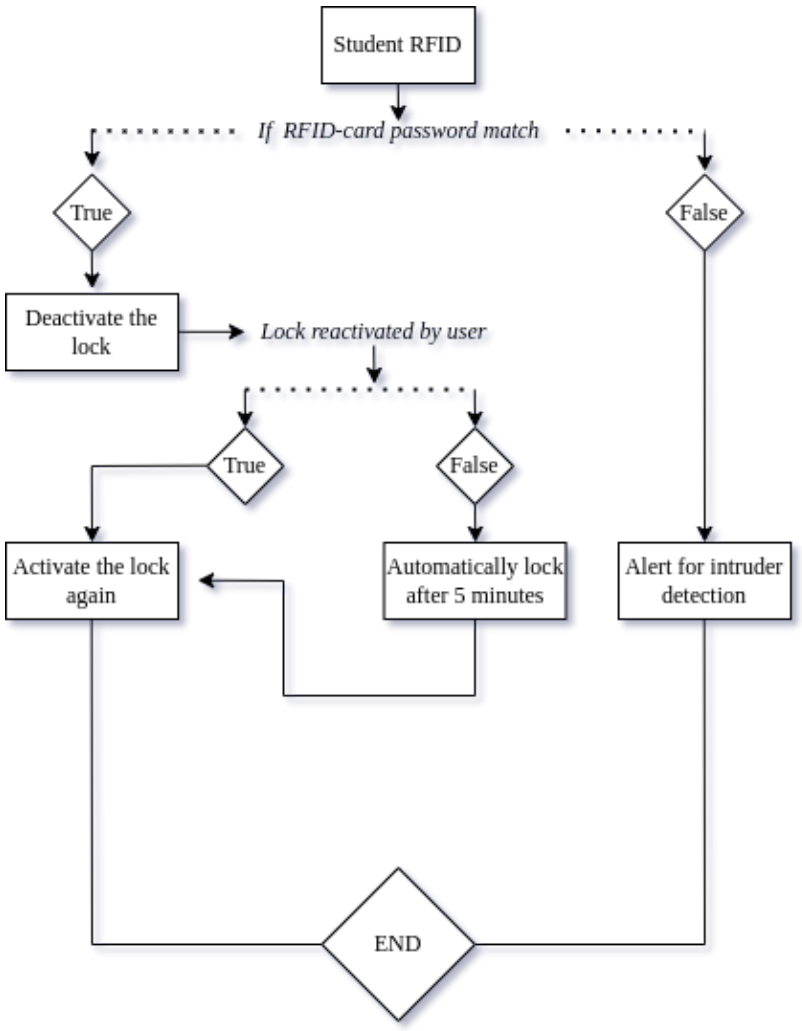

Figure 1: Algorithm of RFID-based lock.

RFID-based attendance algorithm

The main algorithm as shown in the Figure below, of the system will be written in `Python -v 3.9.1` with the student database stored offline in JavaScript Object Notation (JSON) file.

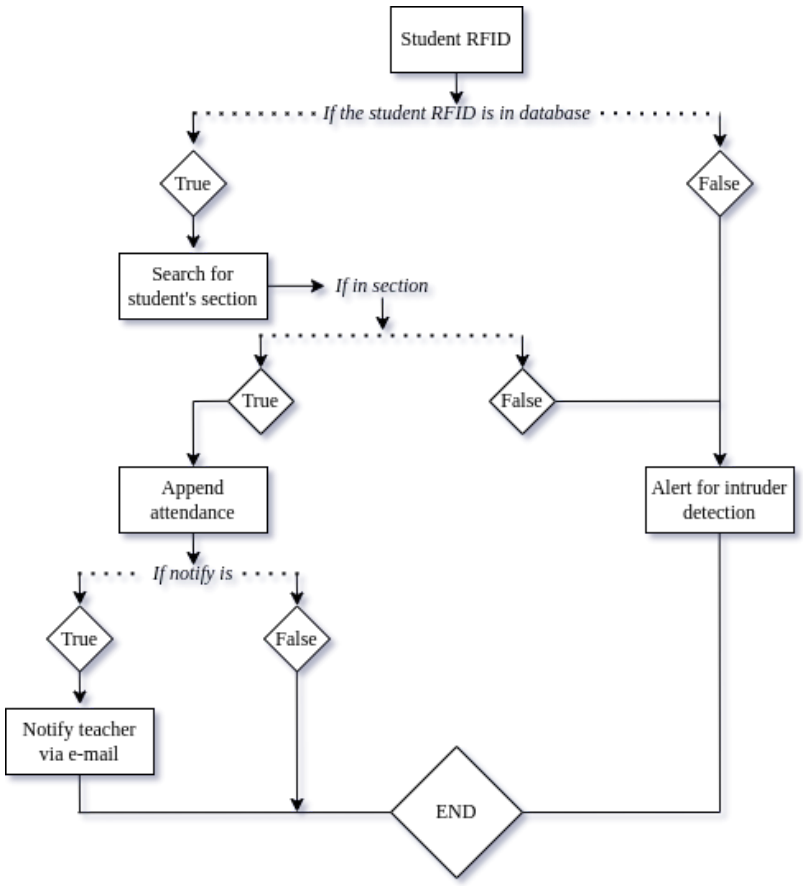

Figure 2: Algorithm of RFID-based Attendance.

There are various third party modules used to aid the python standard library was for successful execution of the function. `pandas.DataFrame` was used for attendance of the students, as well as `pandas.to_excel` function for everyday reporting to teachers. The email was sent using `Python`'s `smtplib` and `ssl`, Simple Mail Transfer Protocol, and Secure Sockets Layer, respectively.

Github or local repository was alsoTr setup to be able to modify the data through graphical interface, whereas it was cloned every $24hours$ using `git`. By iterating over the JSON database of the student ID and teacher's email address to be used for cases of intruders, the checking was performed.

A module was first created that will pull an update from the repository and return the changes to the current database. This was done using `os.system` module from `Python`'s standard library.

```
""" other imports here
... """

# function.py

import json
from os import system as sys

class System:
    def __init__(self, PWD, repo):
        self.PWD = PWD
        self.repo = repo

```

```
    def pullData(self):
        try:
            if exists(f"{PWD}/<file_name>") is True:
                sys("rm <file_name>")
                sys(f"git clone {repo}")
            else:
                sys(f"git clone {repo}")

            return True

        except SystemError, KeyboardInterrupt, OSError, ConnectionError:
            return false


    def getData(self):
        try:
            with open("<file_name>") as data:
                studentDATA = json.load(data)

            with open("<file_name>") as Data:
                teacherDATA = json.load(Data)

            return studentDATA, teacherDATA

        except FileNotFoundError:
            while True:
                if pullData() is True:
                    return True, True
                else:
                    continue
```

Then an equation was devised to compare two strings. ID comparison system works by comparing the percent difference of two strings using `SequenceMatcher` function provided by `difflib` :

$$\Delta_{\%} = \frac{|x_{\forall\in\{\theta_1\}} \equiv y_{\forall\in\{\theta_2\}}|}{|\sum\theta|}$$

Where the value can be classified and interpreted based on their range. The action can also be defined based on the value of $\Delta_{\%}$:

$$intepretation = \begin{cases} +\ \Delta_{\%} = 1 & : \text{Perfect comparison, pass} \\ +\ 0.8 < \Delta_{\%} < 0.95 & : \text{Medium difference, error} \\ +\ 0.5 < \Delta_{\%} < 0.8 & : \text{High difference, error} \\ +\ \Delta_{\%} < 0.5 & : \text{Error} \end{cases} \tag{1}$$

```
...
    cardID = INPT[0]

        for ID in studentDATA:
            if SM(None, cardID, ID).ratio == 1:
                pass
            else:
                # send email

        continue
```

Finally, the functions defined in `function.py` can be called in `main.py`, the program the transforms into :

```
# main.py

from funcion import System
from os import system as sys
from os import chdir
from sys import argv as INPT
```

```
import os.path
from difflib import SequenceMatcher as SM

PWD = chdir(path.dirname(__file__))
sysINIT = System(PWD, <repo_link>)

# initiate the system
try:
    while True:
        if sysINIT.pullData() is True:
            break
        else:
            continue
except:
    raise SystemExit(0)
else:
    studentDATA, teacherDATA = sysINIT.getData()

    if (studentDATA is not True) or (teacherDATA is not True):
        pass
    elif (studentDATA is True) or (teacherDATA is True):
        sysINIT.getData()

finally:
    while True:
        cardID = INPT[0]

        for ID in studentDATA:
            if SM(None, cardID, ID).ratio == 1:
                pass
            else:
                # send email

        continue
```

Regression model of RFID-based systems cost effiency

The regression model was based on the previous prices of materials used for the device, and traditional tools needed (e.g. Attendance book, padlock, etc.). The data were preprocessed with goal of increasing its sensitivity to be able to capture the precision and accuracy in the regression model :

$$\beta_f = \frac{\beta_i}{\ln(\sqrt{\beta_i})} + |(\theta_1 - \theta_2)| \tag{2}$$

where β_f is the processed value of dependent variable, and $\theta_1, \theta_2, \ldots, \theta_n$ are the parameters. Based on this and mathematical laws governing the economics, the regression model were defined, where the algorithm will learn the hypothesis using the existing given dataset to predict $\mathbf{y_i}$ using $\mathbf{x_i}$ and the parameters of the hypothesis, $\theta_1, \theta_2, \ldots, \theta_n$, that can be presented in matrix form :

$$\begin{aligned} \mathbf{y_i} &= m\mathbf{x_i} + b \\ h_\theta(\mathbf{x_i}) &= \theta_n + \theta_o(\mathbf{x_i}) \end{aligned} \tag{3}$$

$$\mathbf{x_i} = \begin{pmatrix} x_{11} & x_{32} & . & . & . & . & x_{1n} \\ x_{21} & x_{32} & . & . & . & . & x_{2n} \\ x_{31} & x_{32} & . & . & . & . & x_{3n} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ x_{m1} & x_{m2} & . & . & . & . & x_{mn} \end{pmatrix} \qquad \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ \theta_j \\ . \\ . \\ \theta_m \\ \theta_n \end{pmatrix}_{n+1,1} \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ y_j \\ . \\ . \\ y_m \\ y_n \end{pmatrix}_{m,1} \tag{4}$$