AARS Team Project Written Report (Draft)

By: Jimmy Garcia, Kevin Karvandi, Surabhi Singh, Phoenix Henry

Introduction:

This is going to be our written report of everything we have done for the AARS project. This written report will be contributed by everyone on the team. This is not our final draft and will get more stuff added on in the future; this is just to plan for the final written report and to see what else we would need completed. Any questions can be forwarded to the team leader.

Meet the Team:

Surabhi (Researcher): Hello, I am Surabhi, the lead researcher of the group. I focus on collecting data and information on how to do the project and what is needed for it to succeed. The research I gather is anything related to the project and will later be compiled together into a written report and presentation for our project. I make sure to research necessary things that will benefit the project altogether.

Phoenix (Notetaker): Hello, I am Phoenix, I am the note taker of the group. I personally love taking notes and keeping things organized, that is why I was chosen for this role. I record notes, discussions, topics, and important information. The notes I take helps the group remember what we have talked about, will talk about, and things we need to talk about. This role is highly important so we can keep track of our discussions and ideas. I make sure notes are taken each time we have a group discussion.

Kevin (Lead Programmer): Hello, I am just going to steal the same intro as Jimmy and introduce myself. My name is Kevin and I will be the lead programmer if you could not guess by the name of my job. I was picked for this job because mainly I knew a lot python and Java, plus I am good at problem solving. I as the lead programmer I will be building the DB that the project will have to be based on. I hope our team will work well.

Jimmy (Team Manager & Programmer): Hello, I am Jimmy and I am the team leader and programmer. I organize and keep tabs on everyone to make the teamwork the best it can. I am the original proposer of the AARS project, so I help lead the development of the project. I am also the programmer which means that I have worked on building the website and database code. I am only one of two available programmers which is why my focus is on the database code. As the team leader, I am proud of having Surabhi, Phoenix, and Kevin on my team.

Project Purpose:

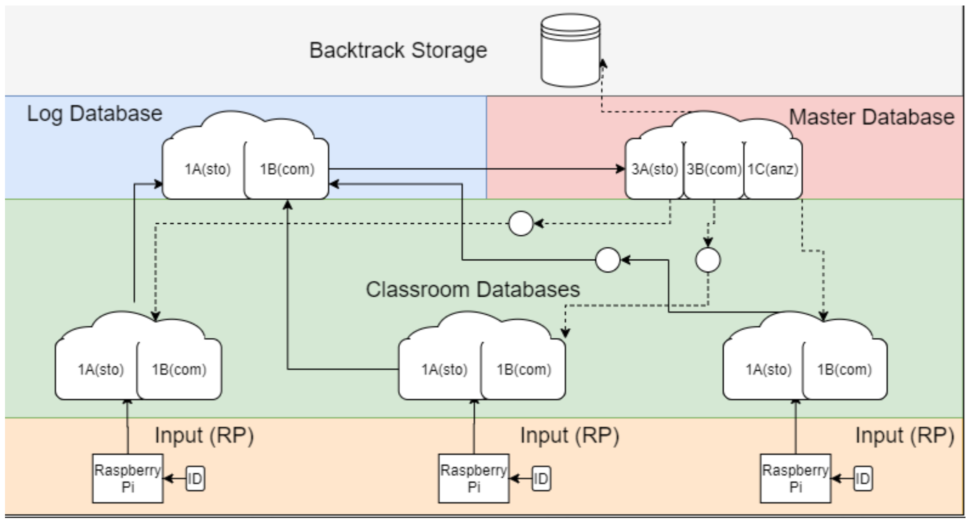
This project was created in order to address several issues teachers have complained about. It also addresses I as a student see happening with substitute teachers. I (Jimmy) drew inspiration for the project when about a year ago, I had a discussion with my English teacher about what were some common issues with teaching. He explained to me that he loses a lot of time taking attendance from students, which kind of bothered him. He said he wished that there was a system that could automatically mark attendance without him doing anything. We talked about ideas and one of my proposed idea was something very similar to AARS. He said it was a good idea, but we never really talked about it again nor I thought about making it a reality. When I first had this talk with him, I struggled to understand how attendance could make a big difference on a teacher's life. This question got answered later-on when I had a substitute for my English class (which was on this current academic school year) and she struggled to take attendance. I had to help the teacher identify who was here and who wasn't. This is when I finally realized that an attendance system would be important in fixing this issue. A few months after that, for my NWIT 101 class, I was given the task to come up with an idea that would incorporate hardware, Wi-Fi usage, and that it addressed a problem. This was the right time to make a system to automate attendance possible, and so, AARS was born.

Product Features & Specifications:

The project was designed with a school setting in mind, but it may also be used on an office setting. We know that our project might have already been widely implemented for security (such as HID systems) but we wanted to make a widescale and cheap alternative; AARS does exactly just that. Each sensor (1 unit) will cost around $70 USD compared to $200 USD average of industry grade controllers. The software on most industry grade systems is also paid separately, compared to AARS which uses free and open source software. This project relies more on the software than hardware so you know that you will not lose much if anything happens to any sensor, it can get replaced and set up with our software at no additional charge.

The original software was supposed to be built in Python, but we opted to build it in JS to make it easier and save time. The project is fully integrated with a user interface (UI) which makes it easy for a teacher to check attendance fast and efficient. We also have status highlighting for certain names to pop out. The database and the UI can both be customized by either forking (preferred method) or by cloning the GitHub repository. Our team likes to be transparent and want to ensure you are satisfied with your product which is why we offer the software as open source and give resources to help you with your product. We also offer a way to contact us in case you need more direct support with something. At team AARS, you aren't just a customer, you make part of our team!

Here are the current specifications of how the original project is supposed to work:



This is the model that is planned for future implementation of AARS. AARS is a system of multiple RPs, not a singular product. As I stated above, we will only build one of each database and we will exclude backtrack storage (more why below). This model will be explained below:

Collection of Input:

The collection of input will happen by having the ID (RFID) card be near the RFID sensor. The Raspberry Pi (RP) will then store this information and upload it to the classroom DB. The information that will be collected will include card ID, time it was scanned (metadata), and location (of RP).

Classroom DB:

1A (Store):

1A shall be broken into two parts, initial storage and final storage

Initial storage will compare with data from 1B (compare) to get student ID info of student

Final storage will store all final comparisons and send it to 2A

Received data from 3B will be stored here

1B (Compare):

Will keep info and compare IDs of cards and names (given to the card ID)

Any card ID that is not supposed to be in the class will be flagged.

Flag can be as an object with a Boolean value of true

Infor that is kept will include name, grade, and school ID number

Classroom DB is the very minimum that needs to be built!

Log DB:

2A (Store):

This will only include a final storage, unlike 1A. The final storage will include information of all classroom DBs; only information from 1A will be here. This will be then be compared 2B.

Information sent includes name, grade, presence status (absent, present, late, etc.), location metadata, and school ID number

2B (Compare):

This will have all student ID numbers (not card IDs) and a class schedule of where they should be at a certain time. If any ID numbers are not in the correct location as their schedule indicates is will then be flagged and sent to 3B.

After school session is ended (at around 5 pm), all info on 2A will be forwarded to 3A.

Master DB:

3A (Store):

This will have both initial and final storage, just like 1A

Initial storage will have all those IDs that have been flagged. It will be stored for 3 minutes and then forwarded to 3B

Final storage will have info of all A sections of all Log DBs. It will be stored for a maximum of 1 cycle (24 hours). This info may be forwarded to 1C where it will be analyzed. This info may also be pushed to Backtrack Storage without going to 1C.

After 1 cycle, all data will be deleted from here if it is not pushed to Backtrack storage.

3B (Compare):

The job of this is only to forward flagged names to the appropriate location of classroom databases. It will be direct without intervention of log DB and sent to 1A final storage part of classroom DBs.

This is the main communication with all other DBs if desired; this is an optional feature which will be fully compatible (not included in demo).

1C (Analyze):

This is the part where one can analyze all data. Additional software can be made to work with this part of the database and can give more info to school systems.

Info analyzed here may also be pushed to Backtrack DB.

This part has not been made active in demo and is optional

Backtrack Storage:

This is only theoretical and not yet implemented nor ready to be implemented yet. This can be cloud storage or a local server. All info will be stored here until there is no more storage space. This storage will be used to keep logs of every cycle within the AARS system. It will be available within future versions of the revised AARS system.

Quick notes:

All databases have a push & pull mechanism

Pull part has a dashboard (except Backtrack storage)

All dashboards will get info from final storage parts of A DBs.

All pushes are made clockwise (shown in diagram)

JSON is highly encouraged for data

DB environment will be local on the web server as a demo

Information that will be gathered from each database:

Input from ID card:

ID from card

//Location that was scanned (location must be a string; a constant variable number must be associated with it)

Time it was scanned (will used to be marked late or on-time)//

/*This info will be forwarded to classroom DB*/

// means this can be done on classroom DB

Classroom DB:

List of student ID numbers that correlate with a card ID (ID numbers will be locally stored). These student ID numbers will include:

Name

Grade

Card ID info

Status of wrong location (forwarded from Master DB)

All info forwarded from input of card will go here too (except card ID number)

/*This info will be forwarded to Log DB*/

Log DB:

List of student ID numbers that correlate to their student schedule (ID numbers are also locally stored here). The schedule will include:

Name & Grade (from student ID numbers)

Class periods and room number (which will have a location number from RP or classroom DB)

Time each period starts

Flagged students not in correct location

/*This info will be forwarded to Master DB*/

Master DB:

All information forwarded from Log DB

Will forward info to classroom DB based on location for those students that are on another location

Analyze software (for future iterations of AARS)

/*This info will be forwarded to Backtrack storage*/

Backtrack Storage:

All info that has been collected from Master DB will be stored here forever

Only for future iterations of AARS

Project Development:

Before you start:

Before you start, it is important that you have enough time to complete the project. A timeline of fulltime dedication of one to two weeks should be enough. This project will require knowledge in both programming and electronics. It is best if you have a team, but you can also do this on your own if you please. Depending on your skill level, you can build certain parts of the AARS system. Below you will find this list of parts and skill level. Patience will be a very big factor into successfully completing this project. You are bound to have many mistakes and re-dos; this is normal. If you get stuck on something you may view our resources for reference.

Skills & Prerequisites:

Local Raspberry Pi RFID detector simulated (Beginner):

-Basic familiarity with HTML and CSS (programing language)

-Basic familiarity with web design

-1 or 5 days of full-time dedication

Local Raspberry Pi RFID detector original (Beginner):

-Basic familiarity Python (programing language)

-Basic familiarity with electronics

-1 to 3 days of full-time dedication

Raspberry Pi RFID detector with database dashboard (Intermediate):

-Intermediate familiarity with Python

-Intermediate familiarity with JavaScript

-Basic familiarity with JSON

-Basic familiarity with HTML and CSS

-Basic familiarity with electronics

-5 days to 1 week of full-time dedication

The whole project with 3 databases (Advanced):

-Intermediate to advanced familiarity with Python

-Intermediate to advanced familiarity with JavaScript

-Intermediate familiarity with JSON

-Intermediate familiarity HTML and CSS

-Basic familiarity with electronics

-1 to two weeks of full-time dedication

Materials (for original project):

1 computer or Raspberry Pi

1 Raspberry Pi

1 RFID sensor

5 RFID tags

1 Breadboard

~50 Resistors with varying resistance

~100 Breadboard cables

~10 Led lights (RGB or red and green)

1 or 2 Piezo Buzzers

Peripherals for Raspberry Pi or computer

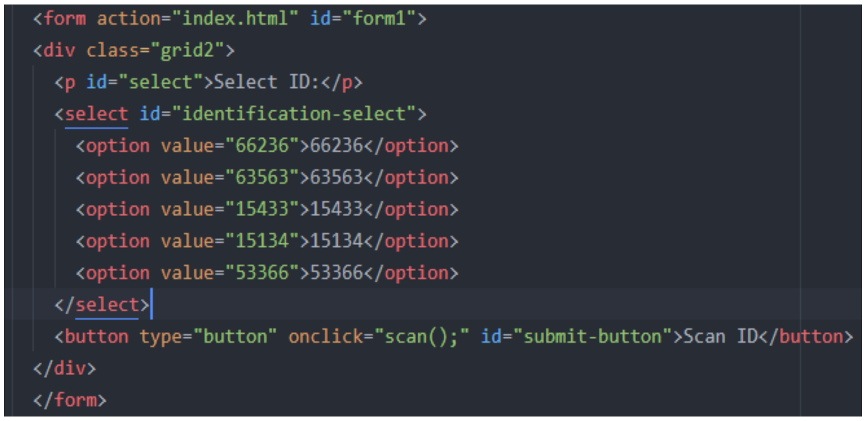
Note: Original material list is located on team project proposal (final) discussion thread

Instructions (current iteration):

Note: These are the instructions for what we built. The previous materials and skill requirements are for those that were planned. Current instructions are for what we have done.

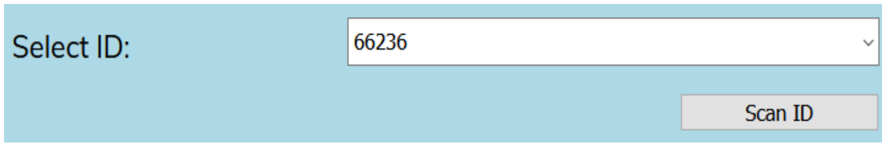
Building Local Raspberry Pi RFID detector (simulated):

In order to simulate the RFID detector, we will use the capabilities of HTML and JS in order to achieve this. We will create the webpage first in order to recreate the “scan” action and to get the input for JS. We will only create a small form which will have a ‘select’ option as well as a button to scan the ID. The IDs will be placed in the select input and the button will be linked with an onclick event. Of course, we will have id options for our html tags in order to use it with JS later. Here is what the code should look like:



```
<form action="index.html" id="form1">
<div class="grid2">
  <p id="select">Select ID:</p>
  <select id="identification-select">
    <option value="66236">66236</option>
    <option value="63563">63563</option>
    <option value="15433">15433</option>
    <option value="15134">15134</option>
    <option value="53366">53366</option>
  </select>
  <button type="button" onclick="scan();" id="submit-button">Scan ID</button>
</div>
</form>
```

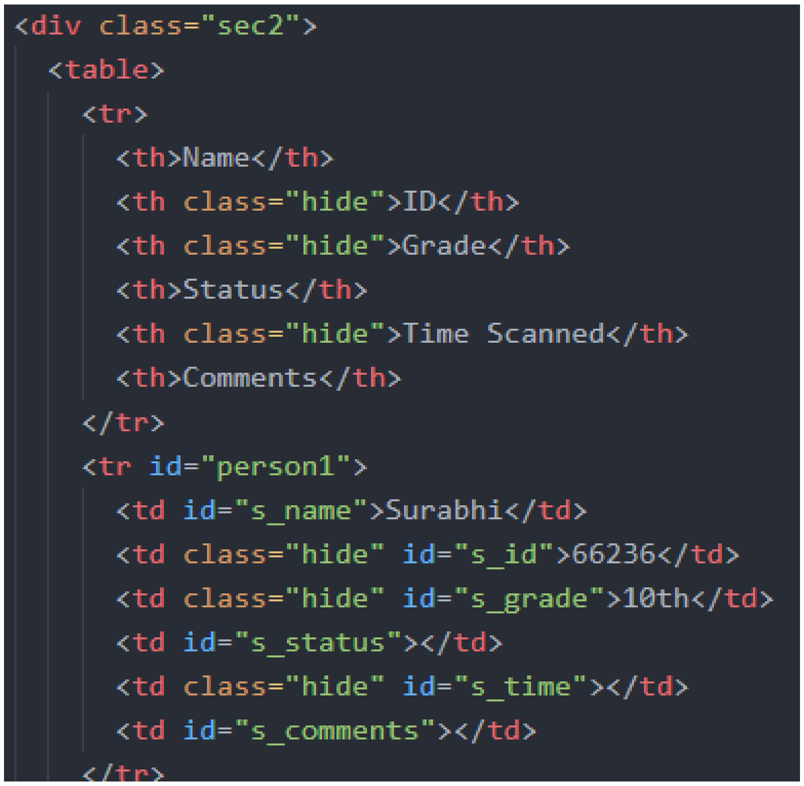
This is the simple view and should look something like this:



You should be done with the UI part of the RFID simulated detector

Building Dashboard UI:

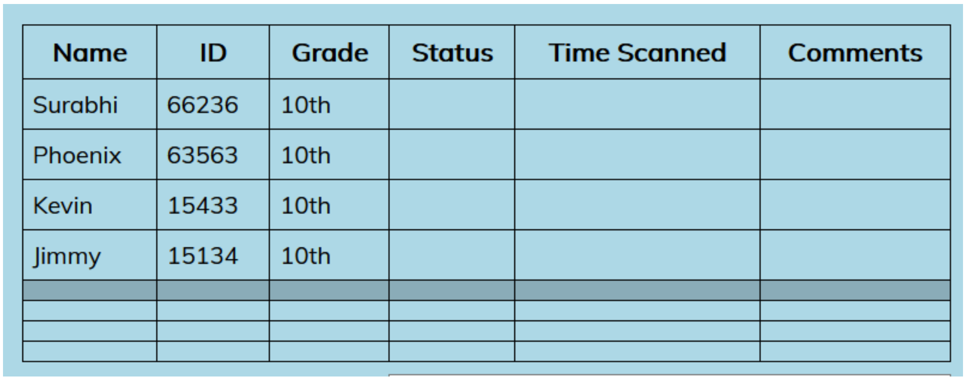
So, in the first step, we only did the UI for the input, so we will now build the UI for the output. In order to do that, we decided to use a table to make it teacher friendly. The table will be long depending on how many students you have in a class. To create the table please refer to this site. We will also include id options for all html tags used in the table (in order to write input later). Your html should look something like this:



```
<div class="sec2">
  <table>
    <tr>
      <th>Name</th>
      <th class="hide">ID</th>
      <th class="hide">Grade</th>
      <th>Status</th>
      <th class="hide">Time Scanned</th>
      <th>Comments</th>
    </tr>
    <tr id="person1">
      <td id="s_name">Surabhi</td>
      <td class="hide" id="s_id">66236</td>
      <td class="hide" id="s_grade">10th</td>
      <td id="s_status"></td>
      <td class="hide" id="s_time"></td>
      <td id="s_comments"></td>
```

Please note: this image is just a snippet of the whole scale of the table. Class is also used to hide elements from mobile view using .hide {display: none;}

The UI of the database should look something like this:

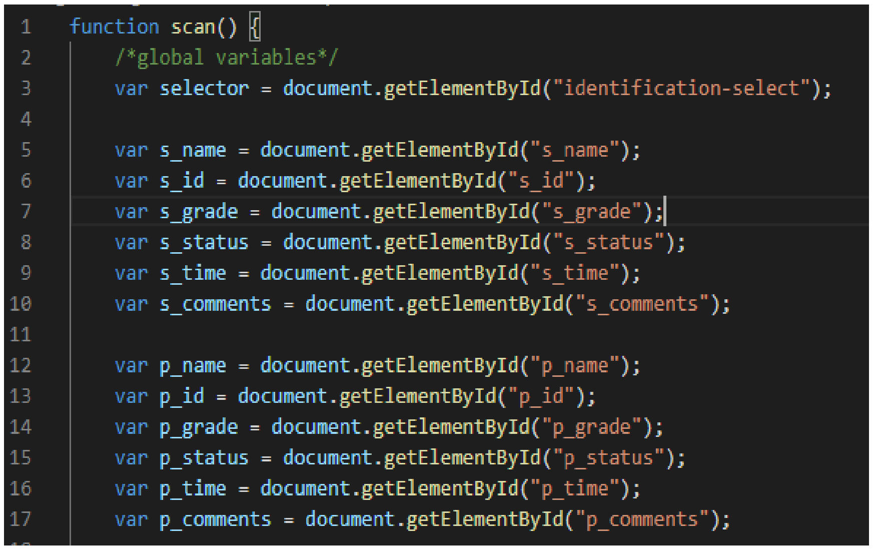


Name	ID	Grade	Status	Time Scanned	Comments
Surabhi	66236	10th			
Phoenix	63563	10th			
Kevin	15433	10th			
Jimmy	15134	10th			

Once you are done with that, you can go to the next step (don't forget the ids).

Hooking up dashboard to JS:

Once we have the table dashboard set up, we are now ready to hook that up to give it functionality. You will use JS so we will make a new file. After you made the new file, you will make a function that will get called when you click the button. After that, you will need to get the ids of all the tags from the table (which is why we require all ids on all tags of the table). To do that use *var [any name] = document.getElementById("[id of tag]"). Note: don't copy the [] it's there so you can place something in there.* The declaration should look something like this:

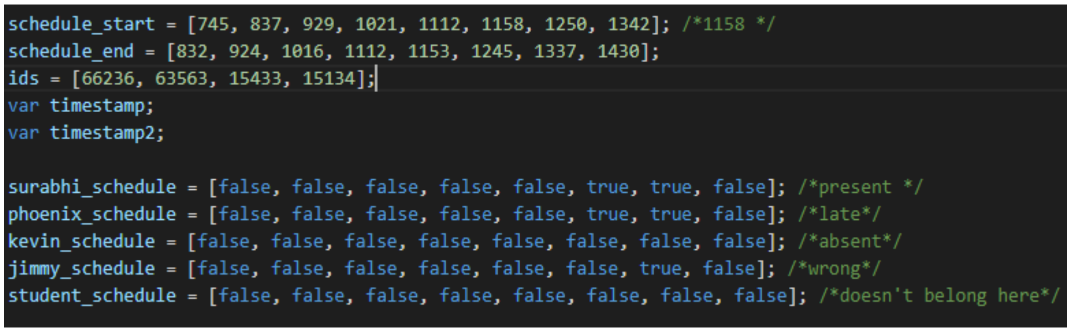


```
function scan() {
    /*global variables*/
    var selector = document.getElementById("identification-select");

    var s_name = document.getElementById("s_name");
    var s_id = document.getElementById("s_id");
    var s_grade = document.getElementById("s_grade");
    var s_status = document.getElementById("s_status");
    var s_time = document.getElementById("s_time");
    var s_comments = document.getElementById("s_comments");

    var p_name = document.getElementById("p_name");
    var p_id = document.getElementById("p_id");
    var p_grade = document.getElementById("p_grade");
    var p_status = document.getElementById("p_status");
    var p_time = document.getElementById("p_time");
    var p_comments = document.getElementById("p_comments");
```

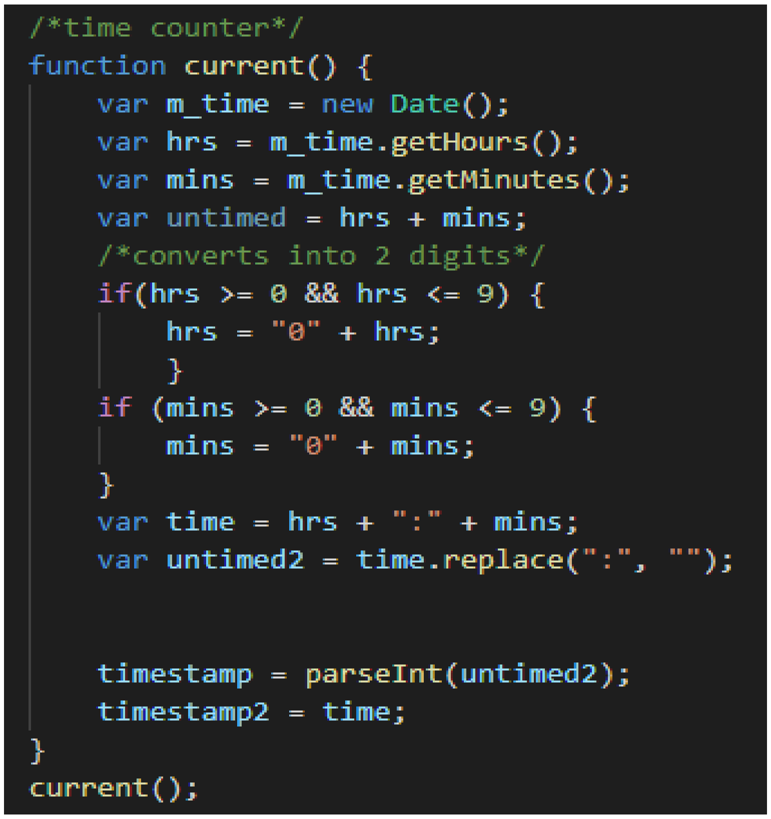
We will now add the schedule of the school manually. To do this, we will make two arrays, one for the start time of each period (or block) and another one for the end. We will input the time to these arrays without the colon (:) and in military time (24hr format). Say class ends at 1:30pm you will input 1330. You will also want to input whether someone is at that classroom or not with a Boolean of false or true (example is below). Make sure to also input all IDs of all students. Do all of this in an array, everything should look like this:



```
schedule_start = [745, 837, 929, 1021, 1112, 1158, 1250, 1342]; /*1158 */
schedule_end = [832, 924, 1016, 1112, 1153, 1245, 1337, 1430];
ids = [66236, 63563, 15433, 15134];
var timestamp;
var timestamp2;

surabhi_schedule = [false, false, false, false, false, true, true, false]; /*present */
phoenix_schedule = [false, false, false, false, false, true, true, false]; /*late*/
kevin_schedule = [false, false, false, false, false, false, false, false]; /*absent*/
jimmy_schedule = [false, false, false, false, false, false, true, false]; /*wrong*/
student_schedule = [false, false, false, false, false, false, false, false]; /*doesn't belong here*/
```

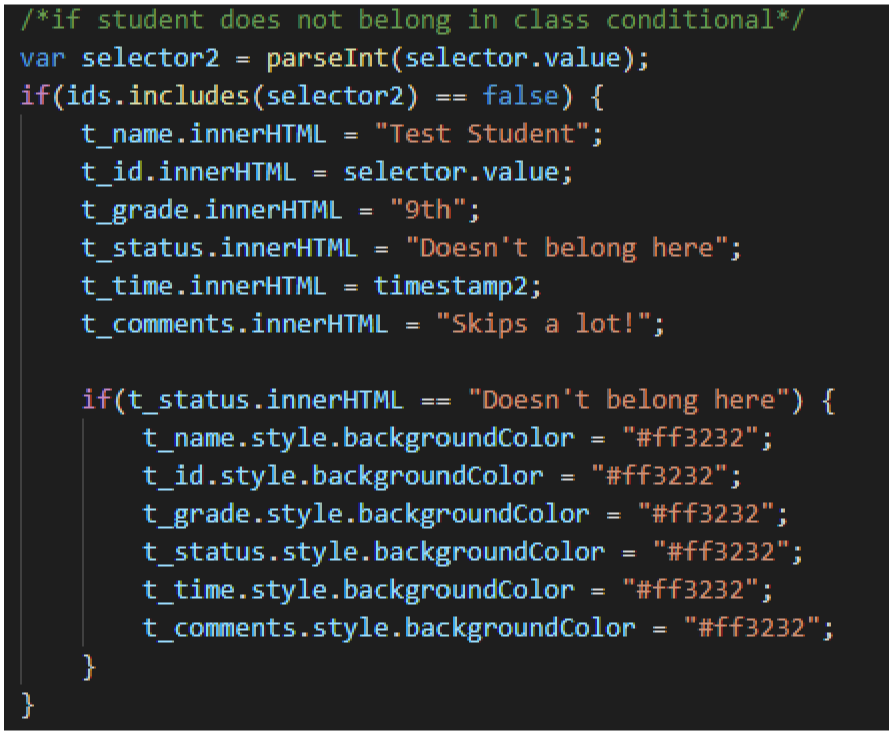
We will now be creating the timer for the timestamp. This is required as it is the bases on how the program determines whether a person is present, absent, or if they do not belong there. Time in JS is relatively simple (compared to other languages like Python) and uses the time function to get the time. We will create the function like shown below:



```
/*time counter*/
function current() {
    var m_time = new Date();
    var hrs = m_time.getHours();
    var mins = m_time.getMinutes();
    var untimed = hrs + mins;
    /*converts into 2 digits*/
    if(hrs >= 0 && hrs <= 9) {
        hrs = "0" + hrs;
        }
    if (mins >= 0 && mins <= 9) {
        mins = "0" + mins;
    }
    var time = hrs + ":" + mins;
    var untimed2 = time.replace(":", "");


    timestamp = parseInt(untimed2);
    timestamp2 = time;
}
current();
```

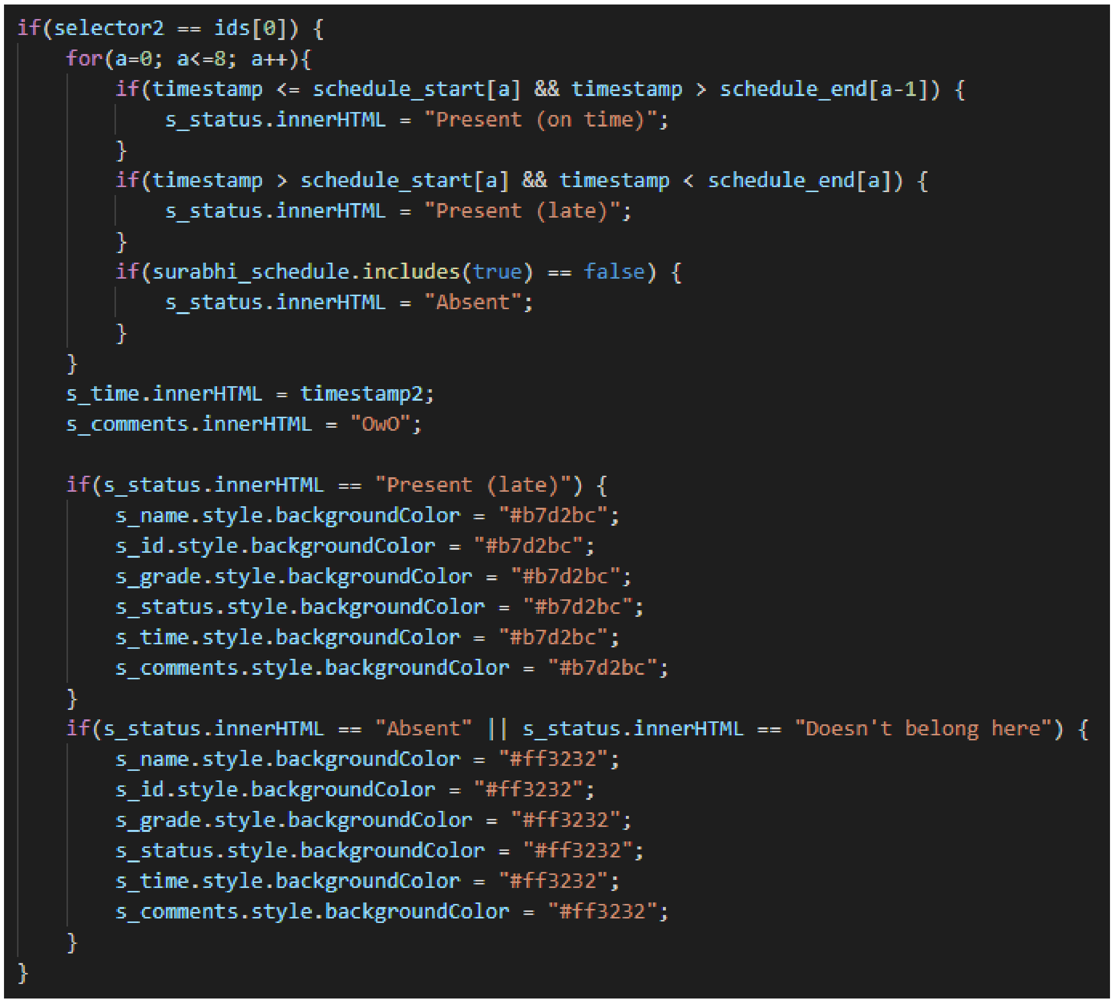
After that time function, we will now create the conditionals for all students in the id array. For this, we need to get the value from the choices on the input simulator from html and will get it here with the *.value* selector. We need to check whether the person is even in the class, if it isn't, it gets tagged red. This is what that code looks like:



```
/*if student does not belong in class conditional*/
var selector2 = parseInt(selector.value);
if(ids.includes(selector2) == false) {
    t_name.innerHTML = "Test Student";
    t_id.innerHTML = selector.value;
    t_grade.innerHTML = "9th";
    t_status.innerHTML = "Doesn't belong here";
    t_time.innerHTML = timestamp2;
    t_comments.innerHTML = "Skips a lot!";

    if(t_status.innerHTML == "Doesn't belong here") {
        t_name.style.backgroundColor = "#ff3232";
        t_id.style.backgroundColor = "#ff3232";
        t_grade.style.backgroundColor = "#ff3232";
        t_status.style.backgroundColor = "#ff3232";
        t_time.style.backgroundColor = "#ff3232";
        t_comments.style.backgroundColor = "#ff3232";
    }
}
```

After we are done with that, we need to then get another conditional to tell the status of the person that is getting scanned as well as inputting manual data into the database. To do this, make sure to copy this code below:



```
if(selector2 == ids[0]) {
    for(a=0; a<=8; a++){
        if(timestamp <= schedule_start[a] && timestamp > schedule_end[a-1]) {
            s_status.innerHTML = "Present (on time)";
        }
        if(timestamp > schedule_start[a] && timestamp < schedule_end[a]) {
            s_status.innerHTML = "Present (late)";
        }
        if(surabhi_schedule.includes(true) == false) {
            s_status.innerHTML = "Absent";
        }
    }
    s_time.innerHTML = timestamp2;
    s_comments.innerHTML = "OwO";

    if(s_status.innerHTML == "Present (late)") {
        s_name.style.backgroundColor = "#b7d2bc";
        s_id.style.backgroundColor = "#b7d2bc";
        s_grade.style.backgroundColor = "#b7d2bc";
        s_status.style.backgroundColor = "#b7d2bc";
        s_time.style.backgroundColor = "#b7d2bc";
        s_comments.style.backgroundColor = "#b7d2bc";
    }
    if(s_status.innerHTML == "Absent" || s_status.innerHTML == "Doesn't belong here") {
        s_name.style.backgroundColor = "#ff3232";
        s_id.style.backgroundColor = "#ff3232";
        s_grade.style.backgroundColor = "#ff3232";
        s_status.style.backgroundColor = "#ff3232";
        s_time.style.backgroundColor = "#ff3232";
        s_comments.style.backgroundColor = "#ff3232";
    }
}
```

We are now done with everything and you have simulated the hardware and one portion of the database. If this was the original project, there would be 2 more databases to build. Hope this guide was useful! Please check our resources page if you are still stuck and need more help.

Product Demonstration:

We had enough time to do the presentation and a simulation to what our system would have worked like if it was built. If you go to our website, there is a demonstration that you can try for yourself. This demo simulates how the project would have worked if it was hooked up to the actual hardware. Because this demo is a simulation, it does not completely reflect the whole model of the original project that was proposed. It does, however, show similar functionality of the UI that would have been in the original project. Please click here to access our demo from our website!

Lesson Learned:

Surabhi: This experience helped me understand how to succeed even when obstacles are in the way. Our group was able to successfully complete our project even through schools closing. I also learned that collaboration is essential for project efficiency. If our group did not assign certain roles and assignments for each other, the project would not have been completed so smoothly. I gained communication skills through this process and further strengthened my researching skills.

Phoenix: This experience taught me the steps of how to make a successful product. O learned that a lot of thinking and interacting with your group members is important with projects. We had some times were some of us had to leave meetings or couldn't attend, but we still made sure they knew what was going on in the group and stuff they needed to complete. Also, we had to figure out ways to make AARS come alive. Our group assigning jobs and roles to everyone helped the project run smoother and more efficiently. I gained information, note taking, and communication skills through this project.

Kevin: This project was a way to change my view on computers and how they interact with each other. It also proved a bond between the group members even though we could not see each other face to face but still got this project done.

Jimmy: There were a lot of lessons I learned from working with my team. I realized that teamwork is essential to creating such a complex project. If there is someone who fails to do something, everyone will be affected in one way or another. It is important to distribute the work based on the abilities of each team member, this way, everything gets done as fast and efficient as possible. Apart from that, I learned that I also need to adapt to other people for me to efficiently work on a team (I don't really like group projects). I also learned that it is important to plan the assignments I am supposed to do. I ran out of time when I was planning to do things for the website, so planning with the time I had is something that I learned is very important. I also learned that It is important to be clear and concise; sometimes I had trouble explaining things with my team members which was not good. I learned a lot and like the experience.

Future-Plans:

Future plans for the project will be to get the hardware in places. This way, the project that we have been planning for weeks will be able to come to life. Once the hardware is put together, we will test the project to ensure that it is working the way we planned it to. Once the project is perfected and able to be used, we may begin to implement it into some classrooms to gather real data so that we can see how effective our product is.

Concluding Remarks:

Surabhi: I am satisfied with how our project went because we were able to (somewhat) bring AARS into life. Although we did not get to put together the hardware essentially, we were able to successfully report and document what our product would be if we did so.

Phoenix: I am happy with how our project turned out. We were able to plan out how AARS would be with details and information. We worked well together even on quarantine. I've gained a lot of different skills like communication skills and note taking skills. Seeing everyone complete their jobs and roles it showed me how many steps, roles, and jobs it would to get to make a productive product.

Kevin: This was fun while it lasted, I learned that I am better than Jimmy at programming but not a grammar. Stay safe and may the world be kind to us all. Also, I hope we can make this in real life and get government sponsorship so that life will be easier to our teachers.

Jimmy: I really liked the experience, specially the plans we had and the final product we were able to complete. The Covid-19 pandemic also affected a lot of our plans in which it was interesting to find new ways to complete at the very least a demo. Teamwork was a very enjoyable part of this whole project. Working with different people was interesting and had to learn to understand and express myself with them. It was a good experience because it allowed me to understand how I can change myself in order to work with a group. This project also gave me joy because it is something that I had planned on doing for about a year ago. The very fact that people wanted to work on my project felt good and it is something that I will never forget. This was the part that I enjoyed about most of the class and when I'm older I will probably look back into this.