Criterion A: Planning
Word Count: 535

Defining the problem

Client: Executive Assistant of Middle School for my school
Advisor: My computer science teacher

Current system:
My school currently implements the late arrival policy using pen and paper. For each of the two buildings, late students are directed towards the office for either the middle school or senior school executive assistant. Here, they fill in their name, class, date and time of entry and reason for being late into a slip.[1] Another slip is then given to the student to submit to their teacher so that they can be informed.[2] The initial slips are collected by the assistants and a manual entry is made into registers. Once a student has had three entries (per month for middle school students and per semester for senior school students), a warning email is sent to their parents. Students are sent back home when they are late for the fourth time.

Flaws with current system:
Firstly, early in the morning, the office gets crowded and long queues of late students form since obtaining, filling, and getting the slip signed takes a substantial amount of time per student (this issue I personally faced and is what led me to consider automating this system for my IA and have an initial conversation with the client – see Appendix A.1). Secondly, filling the register, finding students that have been late multiple times and writing emails is a laborious process that must be performed daily by the office assistants. The system also results in a waste of paper and ink.

Solution

A web application that can be accessed from both offices, allowing the executive assistants to enter the name of the student and match them with their image, which automatically records late arrivals and sends an email to the teacher of the student informing them, sends a warning email to parents after 3 late arrivals, and informs the assistant when a student is late the fourth time.

Success Criteria (decided after interview in Appendix A.3)

Functionality:

1. Dashboard with button to select building name, can be accessed by faculty members.
2. Allows assistant to look up a student by their name and be able to see the student ID, photo, and grade.
3. Assistant can see previous late arrivals including their date and reason.
4. A notice informing assistant that student has already been late thrice (in previous month/semester) should be clearly visible if that is the case.

[1] Refer to appendix A.2.1
[2] Refer to appendix A.2.2

5. Assistant can click a button to record late arrival and enter the reason of late arrival.
6. Time of late arrival, date of late arrival, and building is automatically recorded for (5)
7. An email is sent to the student's form room teachers every time they are late stating the reason and time of arrival.
8. An email is sent both to student and student's parents the third time they are late, informing that the student will be sent back home next time they are late.
9. Assistant can delete a late arrival by clicking a button in case an entry was made by mistake.

Data and backend:

10. Data for all the students, parents and teachers can be easily imported into the database from a JSON format using a simple command line utility each new academic year.
11. Backups of the database are regularly made.
12. Emails are sent using an Outlook account password in a securely stored file.

Security:

13. Accessing the interface after selecting the building name requires a passkey to ensure only authorized people can access it.
14. Rate limiting for requests to ensure security and minimize impact of accidents or buggy code.
15. Log out button that prevents access to the interface until passkey is entered again in the dashboard.
16. Data both entered by assistant and from JSON file validated.

Other:

17. Intuitive, uncluttered, and usable user interface, with large font size and elements.
18. The code base should be modular, extensible and readable to allow for easy maintenance and future changes.

Rationale:

A digital solution will fix the issues with the physical system. A server and client model will be used instead of storing data locally since both buildings should be reading from and writing to the same data.

A web application will be made instead of a desktop application since the application should be easily accessible by all faculty members (in case executive assistant is not present) without any installation required.

React will be used for the frontend (user interface) since it allows making reusable components with state and properties (such as records of late arrivals with a time, building, etc) that can be easily used in multiple places. Bootstrap was used as the UI framework since elements are simple and intuitive.

Node.js would be a good choice for the backend since it has various packages such as Nodemailer (for sending emails), Mongoose (for MongoDB) and express.js (for the server)

that can be used for the functionality that I need. Similar libraries are available for other platforms as well, but I found the packages in node.js to have all the needed functionality.

MongoDB (a NoSQL database) was used instead of an SQL database. This is because MongoDB works very well with JSON data, which would make importing the student, parent and teacher data from JSON easier. MongoDB also works very well with node.js using Mongoose making it suitable for this solution.