Tech Review

Programming For Underrepresented High School Students

Cristian Mann - Group 67

A. *Objective*

Our project is going to need a Relational Database Management System (RDBMS) implemented. This is because we need a way to keep track of individual students, and teachers, that have account on our system. As well as the information on each lesson that gets created and each student's individual progress on each lesson. The teacher needs to be able to monitor each student's progress and grades, and be able to create and modify lessons for students to complete. The environment required for the database is on a closed wi-fi network being hosted on an Intel NUC. It also needs to be able to handle up to 60 users sending requests to it at once.

B. *Options*

1) MySQL: MySQL is an open-source, relational database API owned by Oracle that utilizes the SQL language. The process of setting up a server requires much configuration, using services similar to Apache, but once setup can be accessed via the programs like PhP My Admin. To access it, you must provide valid credentials which give access to permission-limited accounts created by the host. Another choice is to host it on an external server from various server providers that do all the configuration for you. Being created by Oracle, it is a lot of user support and is compatible with a majority of systems and programming languages. In the case of our project, we will be using their NodeJS package provided by Node Package Manager (NPM) since our project will be mostly created in JavaScript.
Another benefit of being created by Oracle is the inherent security that comes with MySQL. Outside the account system required to access it by default which provides authenticity, it in general has extra measures that can be configured to ensure data integrity. Because of the robustness of MySQL, it can be easily scalable for larger projects.
These are the data types that MySQL supports:

- NUMERIC
- DATE
- DATETIME
- TIMESTAMP
- NTINYTEXT
- BLOB
- TEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TIME
- YEAR

- CHAR
- VARCHAR
- TINYBLOB
- TINYINT
- SMALLINT
- MEDIUMINT
- INT or INTEGER
- BIGINT
- FLOAT
- DOUBLE
- DOUBLE DECIMAL
- LONGBLOB
- LONGTEXT
- PRECISION
- REAL
- ENUM
- SET

2) SQLite: SQLite is an open-source, relational database API contributed to by various companies that utilizes the SQL language. What makes SQLite different from some of the more popular database systems, like MongoDB and Oracle, is that it is self contained and file-based. When accessing a database with an application using the API, it is contained within that application. No host required. It is also extremely portable because of it's file-based nature. Everything can be kept in and exported to a single file. As well, as implied by it's name, SQLite is a very lightweight program that makes up for it's lack of robust functionality common in other RDBMS's with being fast and having a low-intensive workload for whatever system it is on. While it doesn't support as many platforms, it does have a NodeJS package provided by NPM we can utilize. It should be noted though that SQLite does not support a user management system, providing very little in the way of inherent security.

These are the data types of SQLite:

- BLOB
- NULL
- INTEGER
- TEXT

C. Comparison

While MySQL comes with more security, configurations and options, a user management system, and scalability, SQLite provide portability, a lightweight infrastructure, and contained within the application using it. MySQL provides 27 data types, while SQLite only provides 4. MySQL is also supported by way more platforms and has had more contributions to it compared to SQLite. They both do support NodeJS though and have NPM packages for them as well. MySQL can be much harder to setup without the use of external servers, but SQLite doesn't require any kind of hosting to operate and is basically ready to use out of the box.

D. Selection

MySQL may be more robust and compatible with more systems, but SQLite would be the better choice for our project. Since our project is designed to run locally on an Intel NUC on a closed network, we are assuming that there will be no access to the internet. Therefore, using a hosting service for a MySQL server would not be optimal. That would mean we have to host and run a local server alongside our application on the same device. This would take up an immense amount of the processing power that the NUC can provide and take up a lot of storage memory. That would really inhibit the scalability of our project. Also, our project does not need the security provided by MySQL since the only sensitive data contained within it would be the passwords of user accounts, which will already be hashed using the blowfish algorithm in the NPM bcrypt package.

SQLite can run locally, and is contained within the application itself, freeing up a lot of resources and active processes. Since the NUC is our only source of computing, having a lightweight system like SQLite gives is the better solution. While it only provides 4 data types, those are the only ones we will need as the data our project will need to store in a database is relatively simple and falls within these data types. Another thing is the portability of SQLite. It is designed to be duplicated and used in various locations with their own students, teachers, and lessons. We aren't hosting one huge online service that anyone can access. We are creating a local tool for high school teachers to use. So, the portability of SQLite is much more important to this project than the scalability of MySQL.

Therefore, SQLite would be the ideal RDBMS of choice for our project.