TEAM PROJECT

Executive summary

Ruth Allison, Nicholas Bandy, Mika Fowles

Introduction

Uni Prep Ltd requirement for a student management system for use by staff at different points in the students' journeys – admissions, finance, academic staff, support staff and the data team, has been addressed following the original Project Report. Data storage is required for approximately 5000 students and staff across 10 locations with up to 50 users accessing the data simultaneously. The key features required were established using feedback from a range of users of the existing system.

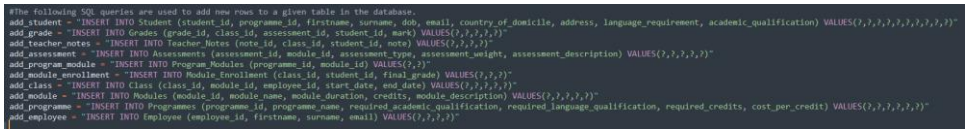
Database creation

Despite originally proposing to use Azure for this project the decision was made to build and test the database using SQLite. This is due to the cost associated with building the database in Azure. SQLite will allow for the same queries and functionality; however, it is not a cloud based solution and will lose some of the benefits of cloud integration such as not requiring staff or hardware to keep the system on-premise and avoiding the risk of hardware failures (Carruthers, 2022).

Uni Prep Ltd was unable to provide sample data for the creation of this database due to privacy concerns. Instead, ChatGPT was used to generate fake data to match the proposed database design. Using Python, the 10 proposed tables were created, and the csv data was inserted into the database file.

Most fields are required in every table by defining them as NOT NULL in the creation of the table. This prevents incomplete or erroneous data from entering the database and removes some need of later data cleaning (Gueta & Carmel, 2016). Although it is expected that any process inserting data into the database should be required to provide complete data without missing values, there are exceptions where a table entry can be inserted with the missing information added after (De Tre et. Al, 2008). An example of this in the database is the module_enrollment table, where the student_id and class_id both must exist when the data is added to the table, however, the final_grade column may be NULL if the entry is added before the completion of the course. This is also seen in the student table, where required information to identify a student such as first name, last name, date of birth, and email must be obtained to identify a student and assign them a student_id. Other fields such as language and academic qualifications may be NULL if they are to be added after the student has provided proof of their qualifications.

Some SQL queries have been prepared to allow a user to add, update, delete, or select data from the database. As discussed in the proposal, most users such as students and teachers should only be accessing the data through a user interface that prevents customization and fills specific values such as the student_id based on the login information to the interface. This prevents any risk of students or teachers gaining access to information that is not directly associated with them or their class. Rows can be added to each table by using the queries shown in Figure 1.



```
#The following SQL queries are used to add new rows to a given table in the database.
add_student = "INSERT INTO Student (student_id, programme_id, firstname, surname, dob, email, country_of_domicile, address, language_requirement, academic_qualification) VALUES(?,?,?,?,?,?,?,?,?,?)"
add_grade = "INSERT INTO Grades (grade_id, class_id, assessment_id, student_id, mark) VALUES(?,?,?,?,?)"
add_teacher_notes = "INSERT INTO Teacher_Notes (note_id, class_id, student_id, note) VALUES(?,?,?,?)"
add_assessment = "INSERT INTO Assessments (assessment_id, module_id, assessment_type, assessment_weight, assessment_description) VALUES(?,?,?,?,?)"
add_program_module = "INSERT INTO Program_Modules (programme_id, module_id) VALUES(?,?)"
add_module_enrollment = "INSERT INTO Module_Enrollment (class_id, student_id, final_grade) VALUES(?,?,?)"
add_class = "INSERT INTO Class (class_id, module_id, employee_id, start_date, end_date) VALUES(?,?,?,?,?)"
add_module = "INSERT INTO Modules (module_id, module_name, module_duration, credits, module_description) VALUES(?,?,?,?,?)"
add_programme = "INSERT INTO Programmes (programme_id, programme_name, required_academic_qualification, required_language_qualification, required_credits, cost_per_credit) VALUES(?,?,?,?,?,?)"
add_employee = "INSERT INTO Employee (employee_id, firstname, surname, email) VALUES(?,?,?,?)"
```

Figure 1: SQL queries to insert a row into a given table

Figure 2 shows how we can select some rows missing data, and provides an example of a query that requires joining multiple tables

```
#The following SQL queries are used to select rows in a table
select_students_missing_data = "SELECT * FROM Student WHERE academic_qualification IS NULL"
select_missing_grades = "SELECT * FROM Module_Enrollment WHERE final_grade IS NULL"
select_student_teacher_module_combinations = "SELECT s.firstname, s.surname, m.module_name, e.firstname, e.surname FROM Student s \
INNER JOIN Module_enrollment me ON s.student_id = me.student_id \
INNER JOIN Class c ON me.class_id = c.class_id \
INNER JOIN Employee e ON c.employee_id = e.employee_id \
INNER JOIN Modules m ON c.module_id = m.module_id \
WHERE s.student_id = 1"
```

Figure 2: SQL queries to select missing student data, or select data from multiple tables

Figure 3 Gives an example of how we can update missing student data once it is received.

```
#The following SQL queries are used to update existing rows in a table. These recommended functions show how to update the tables that have allowed NULL values.
update_student = "UPDATE Student SET language_requirement = ? WHERE student_id = ?"
update_module_enrollment = "UPDATE Module_Enrollment SET final_grade = ? WHERE class_id = ? AND student_id = ?"
```

Figure 3: SQL queries to update student data once it's received

Finally, we can see the queries that would be required to completely remove a student and all of their information from the database in Figure 4.

```
#The following SQL queries are used to delete a student from the database
remove_enrollment = "DELETE FROM Module_Enrollment WHERE student_id = ?"
remove_grade = "DELETE FROM Grades WHERE student_id = ?"
remove_notes = "DELETE FROM Teacher_Notes WHERE student_id = ?"
remove_student = "DELETE FROM Student WHERE student_id = ?"
```

Figure 4: SQL queries to delete a student and all data related to them from the database

Using the QueryTable function in Figure 5, we demonstrate the process of adding a student, updating their information, and then deleting that student through the function calls shown in Figure 6. The results of this demonstration are shown in Figure 7.

```
def QueryTable(query, data = None):
    '''The purpose of this function is to take data stored as a tuple along with a query listed below, to perform that action on the database'''
    conn = sqlite3.connect('Final_db')
    c = conn.cursor()
    # Only pass through no data for select statements, so we want to print the output
    if data == None:
        c.execute(query)
        print('View row: ' + str(c.fetchall()))
        print('')
    else:
        print('input: ' + str(data))
        print('query: ' + query)
        c.execute(query, data)
    conn.commit()
    conn.close()
```

Figure 5: Python function to connect to the database and perform the selected query

```
# Demo for each type of function

#Insert a new student with incomplete data
QueryTable(add_student,(95,1,'Test','Student','1988/07/04','test.student@email.com',None,'72 Fake Road',None,None))
QueryTable(select_students_missing_data)

# Update NULL value from the new entry
QueryTable(update_student, ('English',95))
QueryTable(select_students_missing_data)

# Delete student from student table
QueryTable(remove_student, (95,))
QueryTable(select_students_missing_data)
```

Figure 6: Queries run for database demonstration

```
C:\Users\nicho\OneDrive\Desktop>python db_queries.py
input: (95, 1, 'Test', 'Student', '1988/07/04', 'test.student@email.com', None, '72 Fake Road', None, None)
query: INSERT INTO Student (student_id, programme_id, firstname, surname, dob, email, country_of_domicile, address, language_requirement, academic_qualification) VALUES(?,?,?,?,?,?,?,?,?,?)
View row: [(95, 1, 'Test', 'Student', '1988/07/04', 'test.student@email.com', None, '72 Fake Road', None, None)]

input: ('English', 95)
query: UPDATE Student SET language_requirement = ? WHERE student_id = ?
View row: [(95, 1, 'Test', 'Student', '1988/07/04', 'test.student@email.com', None, '72 Fake Road', 'English', None)]

input: (95,)
query: DELETE FROM Student WHERE student_id = ?
View row: []
```

Figure 7: Results of the database demonstration

As shown in this section, the database meets all of the proposed requirements, and functions to modify the tables are kept simple. It is expected that the queries provided will be a small subset of

what is required while integrating the database with a user interface, this demonstration has shown that the database will be able to easily handle any query requested by the client.

Software choice

The original proposal recommended the use of MS Azure for this project; however, this is not practical for the purposes of this assessment due to the cost of licensing. Two free to use packages were considered for building the database – SQLite and MongoDB.

MongoDB is the most widely known NoSQL database and is ideal for applications with high data volume where low data complexity with horizontal scaling is desired (Educative, 2023). MongoDB utilises documents which are equivalent to tables in relational databases. It uses sharding (storing a large database across multiple machines) to horizontally scale the database and this can also be used for location-based data separation. (MongoDB, 2023)

SQLite is a serverless relational database management system, best for low data volume where low complexity, efficiency and reliability are the primary drivers (Educative, 2023). SQLite does not have a separate server, instead it reads and writes to ordinary disc files allowing a high level of portability (SQLite, 2023)

Table 1. Comparison of key differences between SQLite and MongoDB

	SQLite	MongoDB
Database model	Relational database using tables	Structure determined by documents. Mix of hierarchical and object oriented model
Portability	High Serverless, single file.	Medium Requires MongoDB server on every machine containing the configuration files
Admin requirements	Minimal No configuration required	Server must be installed and configured before use.
Supported platforms	All UNIX based operating systems e.g. Linux, macOS, Android, iOS Default support for Windows 10 onwards	Linux, Windows (2016 and 2019) and macOS Implementable with the use of an SDK for mobile operating systems
Supported programming languages	Most languages covered	Most popular languages covered

(Sibisi, n.d.)

Given the use requirements of the database – colleges in different geographical locations all needing to share data it is clear that MongoDB is the better choice for this project however the group decided to use SQLite for two reasons – familiarity and timescale.

With only 3 weeks to build and test the database we had to consider what was achievable in the time allowed. The team already had some experience with SQL and relational databases but none with MongoDB. A MongoDB account was set up to compare the two systems in practice, however it

quickly became clear that it was not going to be possible to develop the required amount of skill in Mongo DB in the time allowed. Due to these factors, it was felt that existing experience in SQL would allow us to build a working database in the time allowed, which outweighed the performance benefits of MongoDB.

Table 2. Pros and Cons of using SQLite

Pros	Cons
Free to use	Limited security features for multi-user applications
Existing experience	Unsuitable for large data sets
No installation required	Limited features e.g. outer joins are not supported
	Not cloud-based, physical storage required so risk of hardware failure

Legal and compliance requirements

Given the nature of the data collected and processed at Uni Prep Ltd, we recognise the importance of the data protection and secure handling of personal data. As we propose a new database solution for Uni Prep Ltd, it is essential to consider the legal and compliance requirements applied to education sectors in the UK to ensure the privacy and confidentiality of personal information. In this section, we will outline the key considerations in this respect.

Data protection laws

In the UK, the primary legislation governing data protection is the Data Protection Act 2018 (DPA 2018) and the UK General Data Protection Regulation (UK GDPR). Following the UK's withdrawal from the European Union (EU), the EU GDPR is no longer applicable in the UK but its core data protection principles have been incorporated into UK law as the UK GDPR (ICO, n.d.). The DPA 2018 controls how personal information is used by organisations and implements the UK GDPR (UK Government, n.d.).

These regulations establish the rights and obligations of data controllers and processors, ensuring that personal data is processed lawfully and transparently. Our database proposal therefore includes adherence to these regulations to safeguard the privacy of students and staff at Uni Prep Ltd.

Lawful basis for processing personal data

According to the UK Government (2020), personal data for education providers include (but is not limited to):

- contact information about students, guardians, and staff
- health information
- safeguarding information
- passport information
- exam results

As set out in Article 6 in the UK GDPR, the processing of personal data must have a lawful basis. For this reason, it is crucial to obtain consent from data subjects and fulfil a contract.

Individual rights

Individuals whose data is being processed by education providers have various rights, including the right to be informed, right to access, right to rectification, right to erasure, right to restrict processing, right to data portability, right to object, and rights related to automated decision-making and profiling (ICO, n.d.).

Data security threats

Data security breaches could occur due to different causes e.g., intercepted communications, computer viruses, and unauthorised access (Gillenson, 2011). To mitigate such risks, it is of paramount importance that various data security measures are employed.

With respect to the use of SQLite, key data security considerations are as follows:

- **Access control**: It should be noted that SQLite does not provide support for user-level access control. As such, we recommend implementing user authentication and authorization systems.
- **Encryption**: SQLite client library does not support encrypting database files by default. Therefore, it is recommended to use a custom version of SQLite with encryption extensions such as the **SQLite Encryption Extension (SEE)** (SQLite, 2023). It is noteworthy that Article 32 of the UK GDPR specifies encryption as one of the appropriate technical measures for data security.
- **Backup**: To avoid loss of data, we recommend utilising backup solutions. In SQLite, the whole database can be saved as a file by using the **.backup** function (Kreibich, 2010).

Standards

To further ensure data security, employee training on good security practices should be conducted. It is also recommended for Uni Prep Ltd to aim to achieve ISO/IEC 27001, which is an internationally recognised standard for information security management system. Certification with ISO/IEC 27001 is thought to facilitate organisations' GDPR compliance (Lopes et al., 2019).

Ensuring legal compliance and data protection is a fundamental aspect of our database proposal. We recommend aligning data governance with the Data Protection Act 2018 and the UK GDPR by incorporating data minimization, lawful basis, and robust security measures to maximize data security.

Conclusions & recommendations

Although SQLite was chosen for purely practical reasons associated with this assessment, the database is capable of providing all of the required functions outlined in the proposal.

As SQLite does not provide user level access control, all access to the database except for certain administrative staff should occur through a user interface that has security features included.

References

Carruthers, A. (2022) *Building the snowflake data cloud : monetizing and democratizing your data*. Ch. 1. [Online]. New York: Apress L. P. Available From https://learning.oreilly.com/library/view/building-the-snowflake/9781484285930/?sso_link=yes&sso_link_from=university-of-essex

ChatGPT: Model Name: GPT-3.5 Developed by: OpenAI Release Date: [September 2021] Website: https://openai.com/

De Tré, G., De Caluwe, R. and Prade, H.. (2008). Null values in fuzzy databases. Journal of Intelligent Information Systems, 30 (2), 93–114. Available from https://doi.org/10.1007/s10844-006-0021-0

Educative. (2023) MongoDB vs. SQLite. Available from: https://www.educative.io/answers/mongodb-vs-sqlite [Accessed 4 July 2023]

Gueta, T., Carmel, Y. (2016) Quantifying the value of user-level data cleaning for big data: A case study using mammal distribution models, Ecological Informatics, Volume 34, Pages 139-145, ISSN 1574-9541, Available From https://doi.org/10.1016/j.ecoinf.2016.06.001

Gillenson, M. L. (2011) *Fundamentals of database management systems* (2nd ed.). Wiley.

ICO (n.d.) *A guide to individual rights.* Information Commissioner's Office. Available at: https://ico.org.uk/for-organisations/data-protection-and-the-eu/overview-data-protection-and-the-eu/#GDPR [Accessed 8 July 2023].

ICO (n.d.) *Overview: Data protection and the EU.* Information Commissioner's Office. Available at: https://ico.org.uk/for-organisations/data-protection-and-the-eu/overview-data-protection-and-the-eu/#GDPR [Accessed 8 July 2023].

Kreibich, J. A. (2010). *Using SQLite*. O'Reilly.

Lopes, I. M., Guarda, T. & Oliveira, P. (2019) Implementation of ISO 27001 as GDPR compliance facilitator. *Journal of Information Systems Engineering & Management*, 4(2), em0089. Available at: https://doi.org/10.29333/jisem/5888 [Accessed: 9 July 2023].

MongoDB. (2023) What is MongoDB? Available from: https://www.mongodb.com/docs/manual/ [Accessed 4 July 2023]

Sibisi, M. (n.d) SQLite vs MongoDB – What's the Difference? (Pros and Cons). Available from: https://cloudinfrastructureservices.co.uk/sqlite-vs-mongodb-whats-the-difference/ [Accessed 11the July 2023]

SQLite (2023). About SQLite, Available from: https://www.sqlite.org/about.html [Accessed 4 July 2023]

SQLite (2023) *The SQLite Encryption Extension (SEE)*. https://sqlite.org/com/see.html [Accessed 8 July 2023].

UK Government (2020) *Guidance: data protection for education providers*. Available at: https://www.gov.uk/guidance/eu-exit-guide-data-protection-for-education-providers [Accessed 8 July 2023].

UK Government (n.d.) *Data protection*. Available from: https://www.gov.uk/data-protection [Accessed 8 July 2023].

Appendix

Database dictionary


Database Dictionary.docx

Script to create and populate database


builddbpython.py

List of queries and demonstration


db_queries.py

Database


Final_db

ChatGPT data files


OneDrive_2023-07-15.zip