README - Secure School Management System

Description of the application

The Secure School Management System is designed to allow members of staff within a school to view records of students, such as their names, year group, form and the subjects that the are enrolled in. In addition to this, those users with specific admin privileges are able to add students, delete students and update student records (with none admin staff being able to perform this specific function too). The system is designed to be as secure as possible, with a password system being implemented that not only validates logins, but ensures that user can only access the information that they have the right privileges for. Encryption is also used to ensure that any unauthorised access to the system doesn't result in information being disclosed that places users at risk of being in breach of GDPR and ISO/IEC 27000.

Installation of the application

The system is designed to run on a command line interface, with the code and associated libraries being simple to install through the pip function. This provides an easy level of useability and instant feedback on potential errors that may occur. While the application isn't designed to directly interact with an external database, this could be included with minimal effort and would be preferable if a high amount of records are to be stored.

Background of the application

The application is built using Python and primarily utilises Object Oriented Programming techniques, with the vast majority of functions being called from methods written into the programme and the majority of data being stored as objects within classes. Each of the methods written outside of classes correspond to options afforded to users as they navigate the system, such as menu options when logging

in and the selection of access rights once users have inputted their logins. The system offers two classes, one of individual students and one for student records, with the latter calling students from the student class by their ID.

Other data structures are utilised, with a set being used for the classes that a student may be enrolled on and dictionaries being implemented for the recording of login details. Utilising a set for classes ensures that only unique values can be stored, which in turns ensures that there are no repeats. Schools would ideally utilise codes for their classes, ensuring that these are simple enough to be accurately inputted. Dictionaries being used for usernames and passwords ensures that both are paired together uniquely, with keys being able to be encrypted in a simple but secure way.

Usage of the application

When a user initially boots up the application, they are asked to confirm whether they will be logging in as an admin user or not. User accounts are not placed within both the user and admin user dictionaries, so users who select the wrong option at this stage would encounter issues while entering their login details. Once the user selected whether they are an admin user or not, they are provided with the opportunity to enter their login details, with the application calling different login methods depending on the answer given. Text will appear on the screen if the user provides a username that does not exist in the respective dictionary for that login method, or if an incorrect username and password combination is entered. In both of these situations, the application returns false as a Boolean variable.

Once a user has successfully logged in, they are offered a menu of options to choose from. This again varies depending on whether the user is an admin user or not, and are called as methods during the login stage. General users are able to search for students via their student ID, and are able to read these details (potentially add option to add to classes set). Admin users are offered the option to create, read, update and delete student records as required, with methods for each of these being called by entering the associated number for them. Explicit information is provided to users to ensure that they are able to select the correct option at this stage.

Security Features in the Application

Security is built into the application through the use of Python libraries. Utilising Python libraries for this ensures that these are robust, with them having been peer reviewed by the programming industry. The libraries used for security are bcrypt and uuid.

Bcrypt is utilised for hashing passwords in the system. Each password in the respective dictionaries is hashed for security, with a salted hash being used to further strengthen this. By utilising this library, users are reassured that anyone who may get unauthorised access to the system is not able to gain these. While taking steps to build bcyrpt into the application Geeks For Geeks (2022) was used to troubleshoot some of the initial problems encountered with this.

UUID is used to encrypt student IDs within the system, ensuring that these are not called as plain text. Student IDs are a key part of the application, with them being the key used to call further details on students. As such, ensuring that these are not easily breakable ensures that any unauthorised access to the system doesn't result in these being readily available to anyone. For the use of UUID, an article from MayurB at Medium (2023) provided insight on how this should be done, including influencing the use of UUID4 to ensure randomness and full security.

Practical techniques have also been used during the development of the application to ensure that data is only accessible to those who are entitled to it. This is done by ensuring that only admin users are able to operate functions that create or delete records in the application. By ensuring that this is built into the application, the risk of information being disclosed in a way that would put a user at risk of breaching GDPR is minimised, and data subjects are ensured that their personal data is secure. During the development of the application, reference was also made to tools such as OWASP Top 10 Standards (OWASP, 2024), which are designed to ensure that developers keep security at the heart of their work.

Testing of the Application

Unit tests have been written and used to ensure that the application works effectively. The unittest extension is used to run these and generate the results. Unit tests have been completed for both the Student and StudentRecords classes, with each function within these being tested to ensure that they work. Test records for each of these were created and used as the data for test, with the results indicating that all functions work as required.

Manual testing has also taken place with the flow of the application, ensuring that the login system works accurately, and ensuring that the correct functions are called as required. To do this, sample login accounts were created and stored within the respective dictionaries in the application. These login details are then tested to check whether they are linked together, or whether a correct username but incorrect password work. These tests have indicated that the login system works as required.

Linters

The Pylint module has been used within the development of the application to ensure that there are no issues within this. Upon running this once the development was complete, it highlighted that there may potentially be an issue with repeating functions relating to the hashing of passwords, however this does not impact the running of the application. The linter also highlighted that some of the formatting of the code could be improved, which has been taken into consideration for the final product.

Code structure

The code for the application has been written to conform with PEP-8 (van Rossum, 2001) standards. For ease of both reading and use, the unit test code is stored in a

separate file to the main code, calling from this as needed. The code contains comments indicating what each method or class does, with variables and attributes being named in a way that allows for ease of understanding.

Word Count: 1345

References

Geeks for Geeks. (2023) Hashing Passwords in Python with Bcrypt. Available from: https://www.geeksforgeeks.org/hashing-passwords-in-python-with-bcrypt/ [Accessed 13 October 2024].

MayurB. (2023) UUIDs With Python. Available from: https://mayurbirle.medium.com/uuids-with-python-b133cead1b4c [Accessed 13 October 2024].

OWASP (2024) Top 10 Standards. Available from: https://owasp.org/www-project-top-ten/ [Accessed 13 October 2024].