DATA SECURITY AND PRIVACY PROJECT

TEAM 17

Vennela Chadalavada – 811232432

Sreenath Reddy Kurukunda - 811211580

Rohitha Reddy Muppidi - 811232522

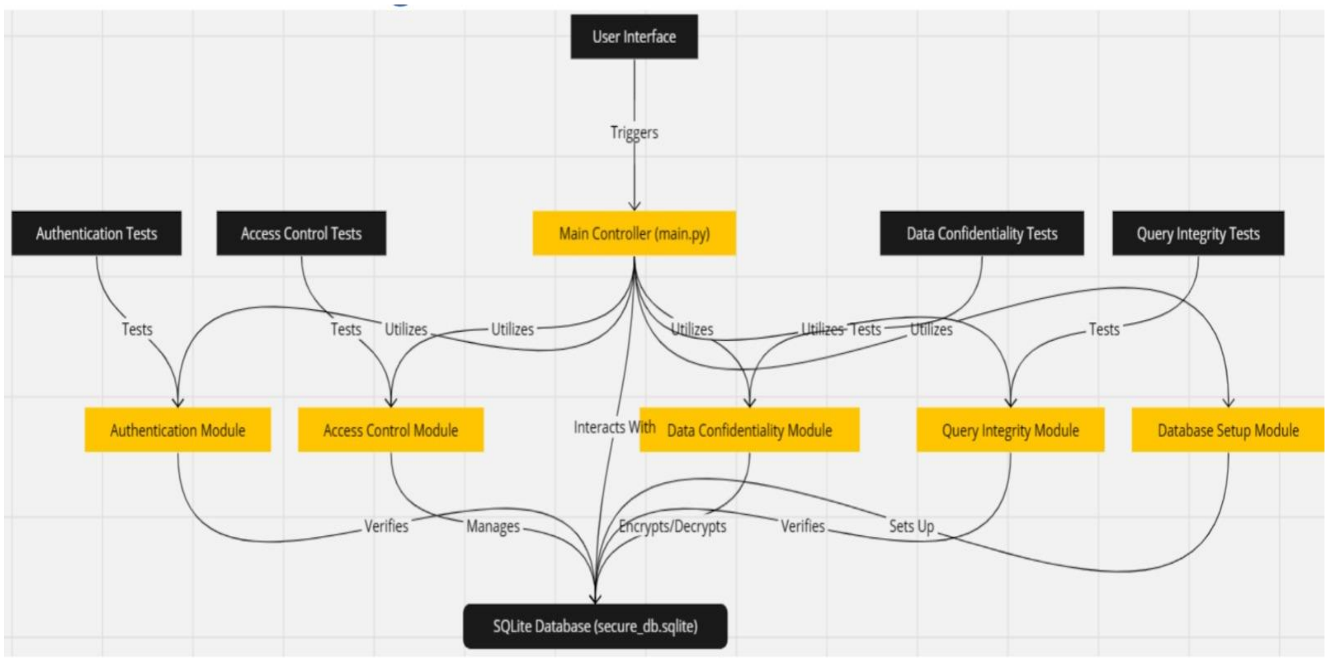
Introduction:

Our objective in this project is to build, put into application, and analyze a safe Database-as-a-Service system that resolves security issues related to cloud-based database solutions. The simple process of outsourcing database administration gives reason for worries concerning data security in the age of cloud computing. The threat of malevolent insiders at the cloud service provider acquiring stored data without the necessary authorization is an important issue nowadays.

The encryption of data before passing on it to the database is a recommended strategy for lowering this security risk. Encryption increases data security, as the data is now encrypted, there are now complications when checking the database. Achieving a balance between database use and data security is the main objective.

Our project primarily focuses on creating a basic database structure with healthcare data. The database will be implemented using a SQL database and has a single table with fields for first and last name, gender, age, height, weight, and health history.

ARCHITECTURE DIAGRAM:



Software Requirements:

Platform: Visual Studio

Database: SQL Lite

Security Features:

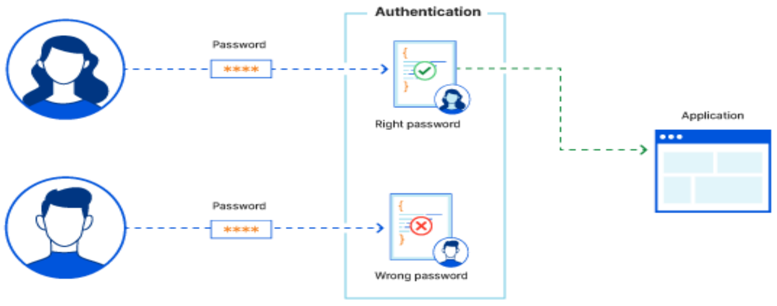
1)User Authentication

Implementation: The authentication mechanism is implemented in the authentication.py file. It uses SHA-256 hashing for storing passwords securely.

How it Works: When a user attempts to log in, the system hashes the entered password and compares it with the stored hash. This ensures that even if the database is compromised, the attacker cannot retrieve the original password.

The system checks if the entered password matches the stored password for that username.

Testing: The test_authentication.py file contains unit tests that verify the correct functionality of the authentication mechanism, including password hashing and user verification.



2)Basic Access Control Mechanism

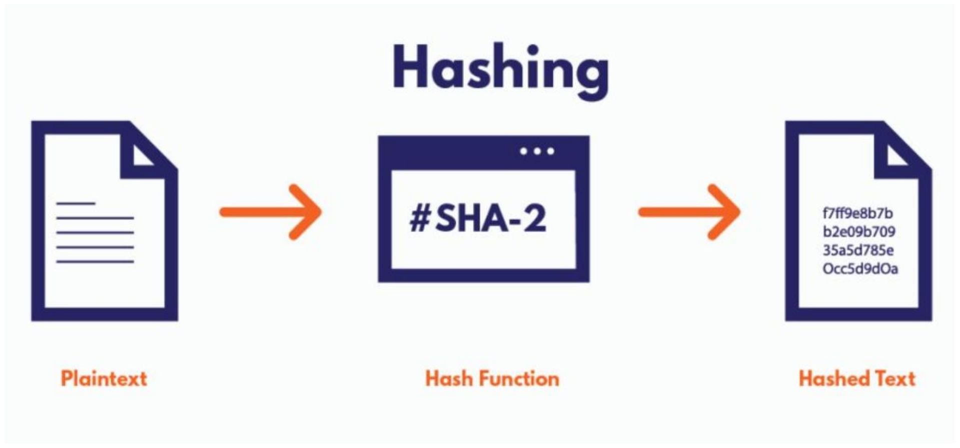
• **Implementation**: Access control is managed in the access_control.py file. It uses SQLite to store user-group mappings.

• **How it Works**: The system differentiates between two groups of users: 'H' and 'R'. Users in group 'H' can access all fields, while users in group 'R' have restricted access. This ensures that sensitive information is only accessible to authorized users.

• **Testing**: The test_access_control.py file contains tests that verify if the access control mechanism correctly restricts or allows access based on user groups.

3)Basic Query Integrity Protection

- **Implementation**: Query integrity is ensured in the query_integrity.py file. It uses SHA-256 hashing to create hashes for data items.
- **How it Works**: The system creates a hash for each data item and verifies it when the data is queried. This ensures that the data has not been tampered with and maintains its integrity.
- **Testing**: The test_query_integrity.py file contains tests that verify the integrity of the data items and the completeness of query result



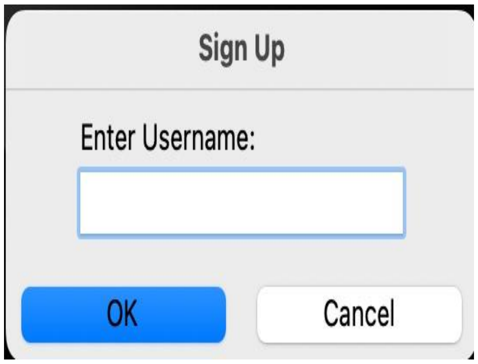
4)Basic Data Confidentiality Protection

- **Implementation**: Data confidentiality is handled in the data_confidentiality.py file. It uses Fernet symmetric encryption for encrypting sensitive fields.
- **How it Works**: Sensitive fields like 'age' and 'gender' are encrypted before storing and decrypted when queried. This ensures that even if someone gains unauthorized access to the database, they cannot understand the sensitive data.
- **Testing**: The test_data_confidentiality.py file contains tests that verify the encryption and decryption mechanisms for sensitive data fields.

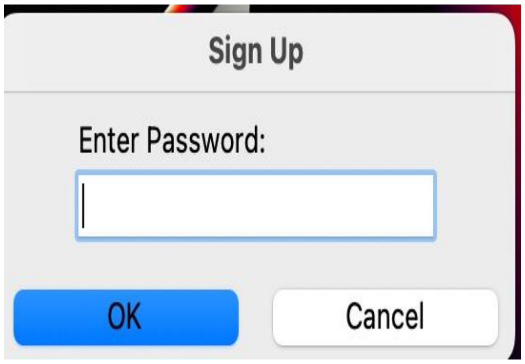
Output Screenshots:

Sign Up:

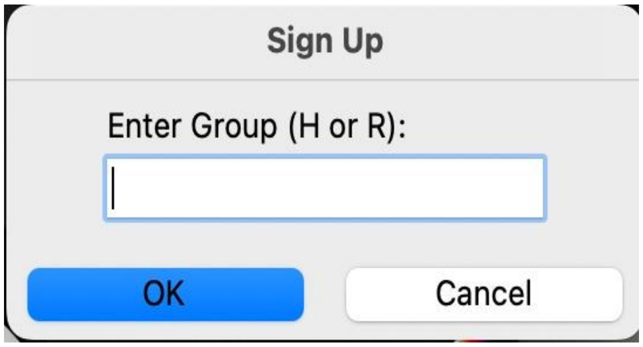
- First, we must enter the username, password and whether we belong to group H or R to sign Up. Then the user is created successfully.


Sign Up

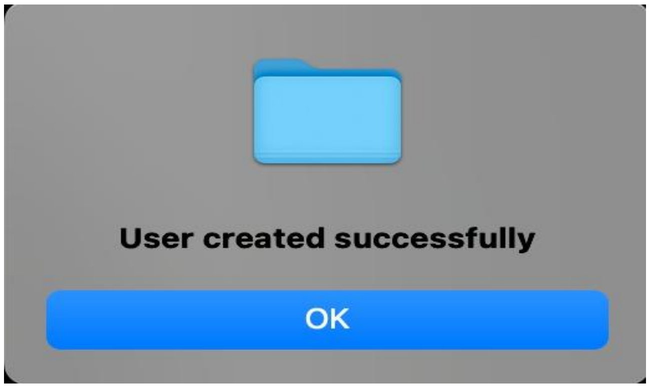
Enter Username:


Sign Up

Enter Password:

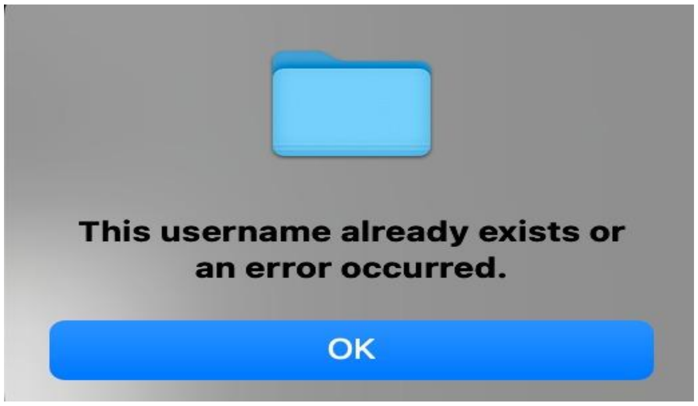

Sign Up

Enter Group (H or R):


User created successfully

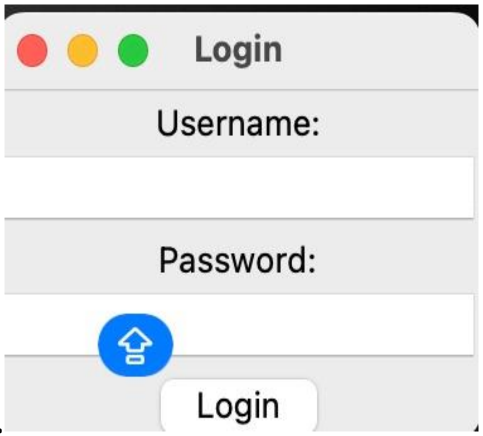
Existing user:

If the username entered already exist in the records, then we will get a pop up as "This username already exists, or an error occurred."

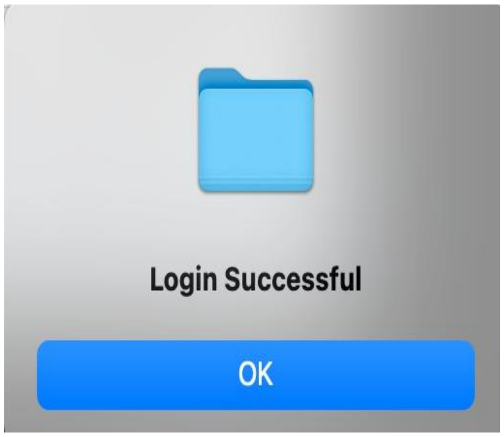

This username already exists or an error occurred.

Login:

Here for Login, we need to enter the above registered username and password.

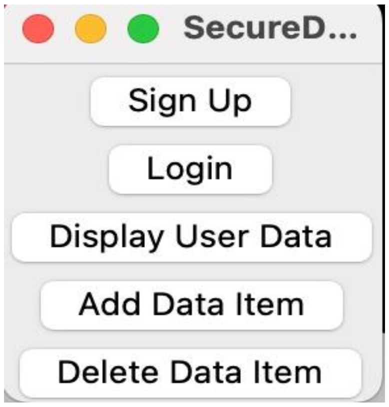


.

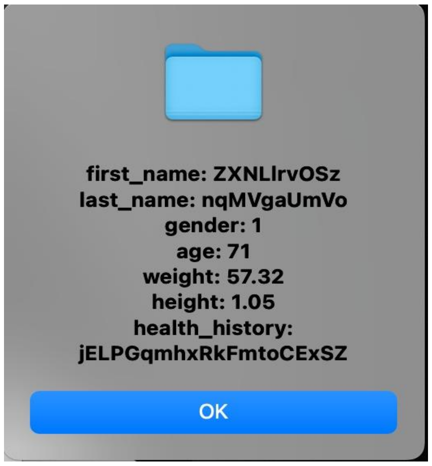

Login Successful

Group H:

Now if the user logged in belongs to group H, then when we hit display user data all the fields will be displayed as shown below and here, we can add or delete the data.

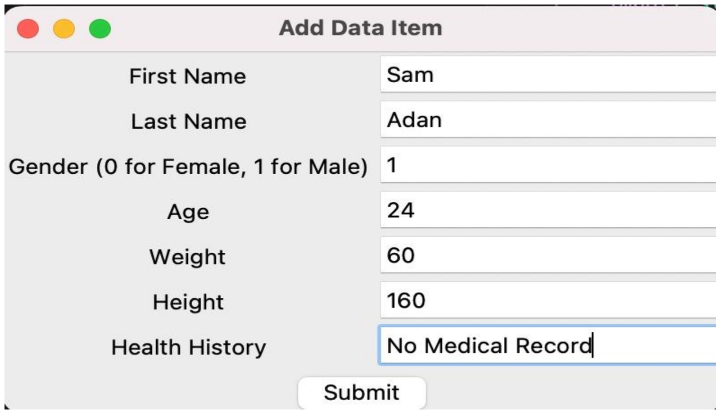


- Display user data



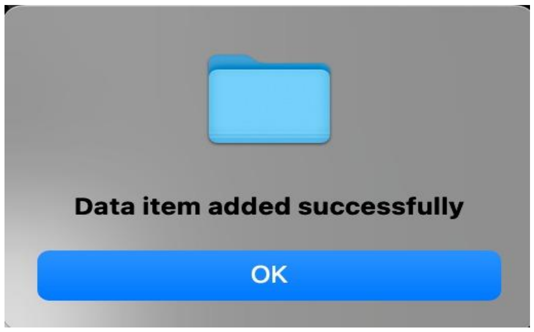
first_name: ZXNLlrvOSz
last_name: nqMVgaUmVo
gender: 1
age: 71
weight: 57.32
height: 1.05
health_history:
jELPGqmhxRkFmtoCExSZ

- Add data item.

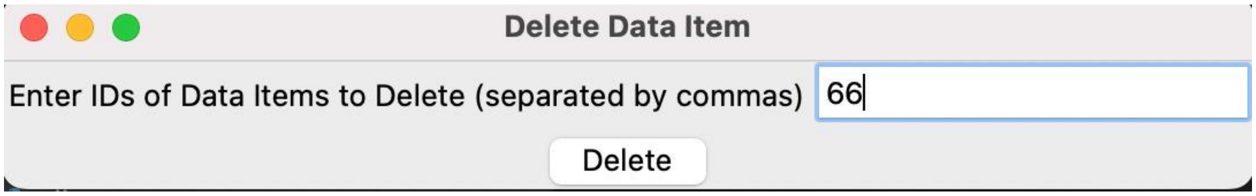


Add Data Item

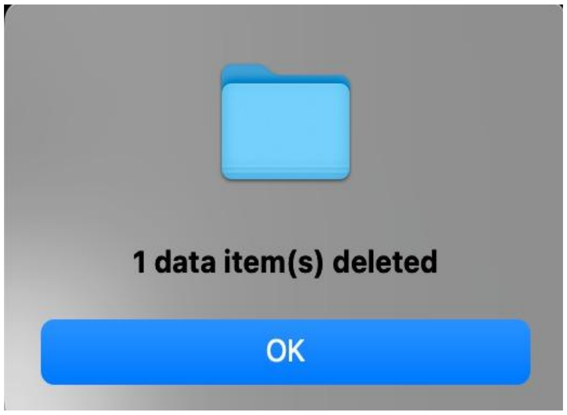
First Name	Sam
Last Name	Adan
Gender (0 for Female, 1 for Male)	1
Age	24
Weight	60
Height	160
Health History	No Medical Record


Data item added successfully

- Delete data item:

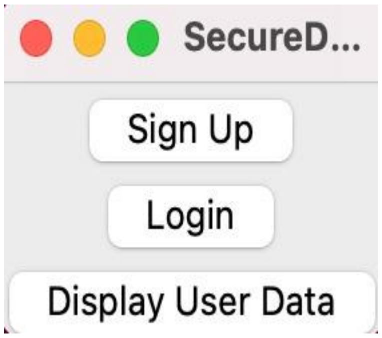


Enter IDs of Data Items to Delete (separated by commas) 66


1 data item(s) deleted

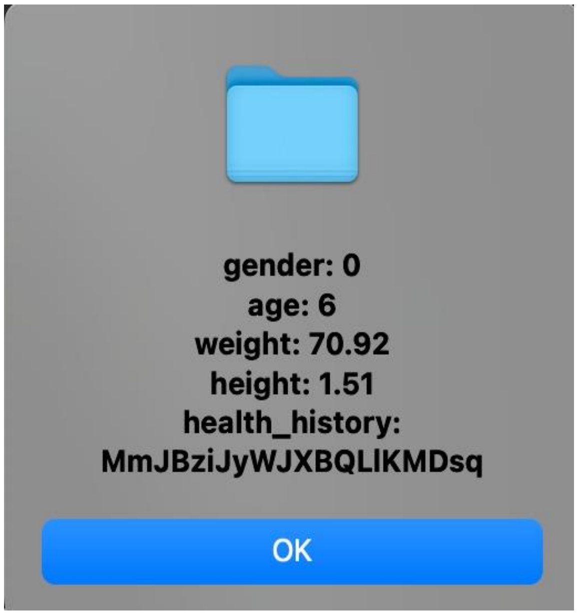
Group R:

If the user logged in belongs to group R, then when we hit display user data all the fields will be displayed except first name and last name as shown below.



- Sign Up and Login are same for group R.

- Display User Data



gender: 0
age: 6
weight: 70.92
height: 1.51
health_history:
MmJBziJyWJXBQLlKMDsq

Limitations Security and Privacy

1. Encryption Overhead: The use of Fernet symmetric encryption for sensitive fields like 'age' and 'gender' may introduce computational overhead, affecting the system's performance.

2. Database Trust: The project assumes that the cloud is semi-trusted. While protocols are in place to protect data, a fully malicious cloud provider could still pose risks.

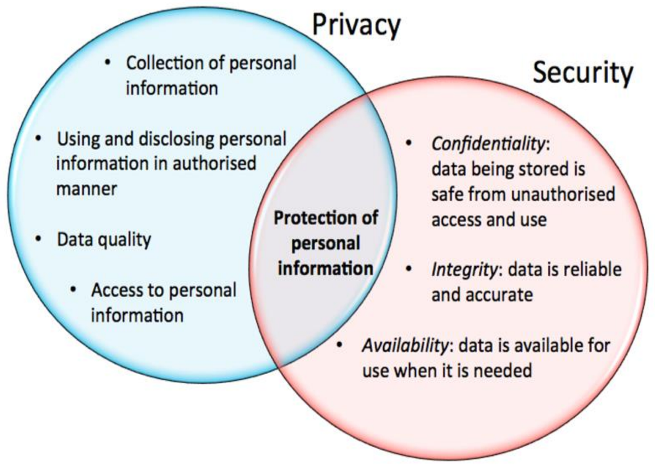
3. Hash Collision: Although unlikely, SHA-256 hashing used for password storage is not entirely collision-free. Two different inputs could, in theory, produce the same hash.

4. Access Control Granularity: The current access control mechanism only differentiates between two types of users ('H' and 'R'). More complex scenarios involving multiple roles and permissions are not covered.

5. Query Completeness: The system verifies the completeness of query results but does not guarantee 100% accuracy. There's a probability factor involved in detecting whether data items have been removed from a query result.

6. Scalability: The project is designed as proof of concept and may require additional features and optimizations for large-scale deployment.

7. Vulnerabilities and Exploits: Software and hardware vulnerabilities can be exploited by malicious actors. Regular security audits and updates are necessary to patch vulnerabilities.


Privacy
• Collection of personal information
• Using and disclosing personal information in authorised manner
• Data quality
• Access to personal information
Protection of personal information
Security
• *Confidentiality*: data being stored is safe from unauthorised access and use
• *Integrity*: data is reliable and accurate
• *Availability*: data is available for use when it is needed

Team Members Contribution:

We are a team of 3 members. Sreenath has gathered all the requirements of the project, created the database with the fields First name, last name, Gender, Age, Weight, Height, Health History and implemented main.py and user authentication. Rohitha has drawn architecture diagram and implemented the basic access control mechanism, basic query integrity protection security features given in the requirements. Vennela has implemented basic confidentiality protection and order preserving encryption scheme to weight attribute and documented the final report. We also provided the GitHub link with the commit history.

Conclusion

The Secure DB Project successfully demonstrates a secure database-as-a-service system with a focus on healthcare information. It incorporates essential security features such as user authentication, access control, data confidentiality, and query integrity. While there are limitations in terms of encryption overhead, trust assumptions, and scalability, the project serves as a robust starting point for building secure database systems. Future work could involve addressing these limitations and adding more features to make the system more versatile and scalable.