Secure Database Report

there are a few steps necessary to make a secure database:

- passwords should be hashed
- parameterize and validate database query to reduce the risk of SQL injection

procedure for hashing passwords:

1. when user registers, ask user to set a password
2. ensure password contains: a special character, a capital letter, at least 8 characters
3. hash password with SHA256
4. store hashed password in database
5. when user tries to log in, hash password and check if it matches hashed password

this way, the user's plaintext password is never saved, and since SHA256 is one way, no one can decrypt it even if they gained access to the website

parameterizing database queries:

- our code CANNOT concatenate and build raw SQL queries
- use DB driver's parameter placeholders (?, %s, $1, etc.) when building queries
- validate input before building query
- for python: sqlite3 is a good library, here's how parameterizing would look:
  ```
  cur.execute("INSERT INTO items (name, price) VALUES (?, ?)", ("Chair", 9.99))
  ```
- flow for how we update database:
  - user enters data into front end (react)
  - javascript sends data to backend via custom API we will build
  - python backend will parse JSON body
  - python converts data to parameterized code
  - python inserts parameterized query into database

extra considerations:

- when we launch our database, we can consider setting up a certificate and adding SSL requirements
- we can also configure an OAuth layer between front and back end after penetration testing if parameterizing is still unsafe
- AWS has 12 free months of database (we pay for any access beyond the free amount) so we can use that and add any native database protection they offer for our app