How have you implemented SQL queries:

In mijn project heb ik SQL-query's gebruikt om met de database te communiceren en specifieke acties uit te voeren, zoals het ophalen van gebruikersgegevens en het controleren van gebruikersnamen. Hieronder is een voorbeeld van hoe ik SQL-query's heb geïmplementeerd:

1. **Gebruikerslijst ophalen:**

cursor.execute("SELECT id, username, role\_level FROM Users")

user\_data = cursor.fetchall()

Hier gebruik ik een parameterized query om te controleren of een ingevoerde gebruikersnaam al in de database bestaat. Dit helpt om SQL-injecties te voorkomen en zorgt ervoor dat gebruikers geen dubbele gebruikersnaam kunnen kiezen.

1. Controle op bestaande gebruikersnaam:

cursor.execute("SELECT username FROM Users WHERE username =?", (userName,))

if cursor.fetchone():

print("Username already exists. Please choose another username.")

Is your application secured against SQL injection attacks and why?:

Ja, mijn applicatie is beveiligd tegen SQL-injectie-aanvallen. Dit kan je uit de manier waarop ik SQL-query's heb geïmplementeerd in mijn code zien. In het specifieke voorbeeld dat ik eerder heb laten zien, maak ik gebruik van parameterized queries, wat een belangrijke beveiligingsmaatregel is tegen SQL-injectie.

cursor.execute("SELECT username FROM Users WHERE username =?", (userName,))

Where have you put the validation layer?

De validatielaag in mijn applicatie is geïmplementeerd in een aparte module genaamd Validation. Deze module bevat verschillende functies die specifiek zijn ontworpen om de invoer van de gebruiker te controleren en te valideren. Voor elk type invoer, zoals voornamen, achternamen, leeftijd, geslacht, gewicht, straatnamen, huisnummers, postcodes, steden, landen, e-mailadressen, telefoonnummers, gebruikersnamen en wachtwoorden, heb ik validatiefuncties geschreven die controleren of de ingevoerde gegevens aan bepaalde criteria voldoen.

Which mechanism is used in the application to implement input validation?

In onze applicatie wordt de input validation gebruikt door middel van regex en logische controles binnen specifieke validatiefuncties, deze worden uit de Validation module gehaald.

Voorbeeld 1 regex:

**E-mailvalidatie:**

Zorgt ervoor dat het e-mailadres een standaardformaat volgt, zoals [gebruikersnaam@domein.extensie](mailto:gebruikersnaam@domein.extensie).

def validate\_email(email):

pattern = r"^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$" # email regex

if len(email) > 50 or not re.match(pattern, email):

print("Invalid email")

logger.warning("User entered an invalid email")

return True

return False

Logische controles typevalidatie:

**Leeftijdsvalidatie:**

Controleert of de leeftijd een geldig geheel getal is en binnen een realistisch bereik valt (0-120 jaar).

def validate\_age(age):

try:

age = int(age)

except ValueError:

print("Age must be a number")

logger.warning("User entered a wrong character for age")

return True

if age < 0 or age > 120:

print("Invalid age")

logger.warning("User entered an invalid age")

return True

return False