# *Abbottabad University of Science And Technology*

## *Department Of Computer Science*

***Semester Project Report***

***Group Members:** __Fahim Iqbal, Shayan Turk, Abdullah Asif__*

***Roll #:** __2023117, 2023139, 2023108__*

***Department:** __BSCS 2nd__*

***Subject:** __Object Oriented Programming__*

# *Title: Cybersecurity Application Development*

*****Personal Cybersecurity Assistant:*****

## *Objective:*

The objective of this lab is to create a basic cybersecurity application in Python that provides user registration, login, and password generation functionality. The application aims to demonstrate fundamental principles of user authentication and secure password management.

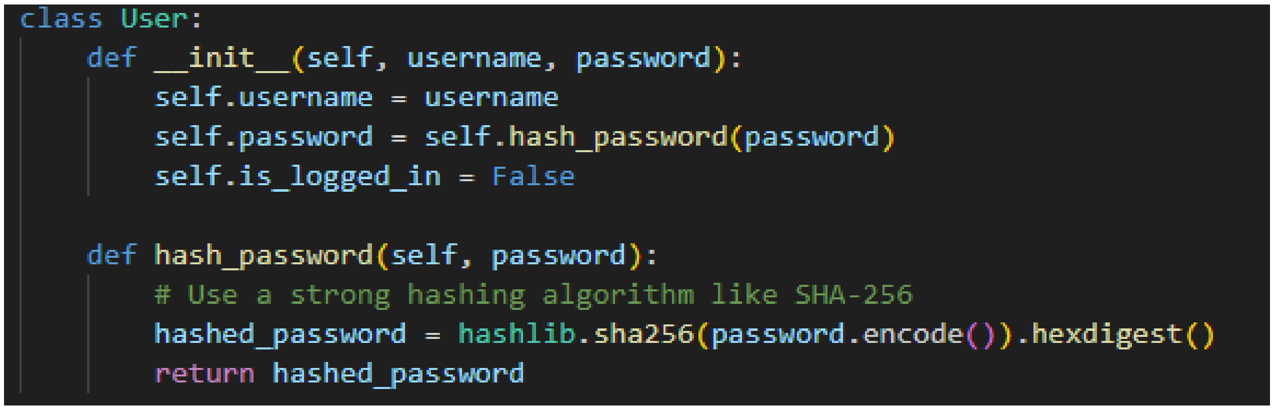
## *Materials and Methods:*

- Programming language: Python
- Integrated Development Environment (IDE): Any Python-compatible IDE
- Libraries: hashlib, random

## *Methodology*

**1. User Class**

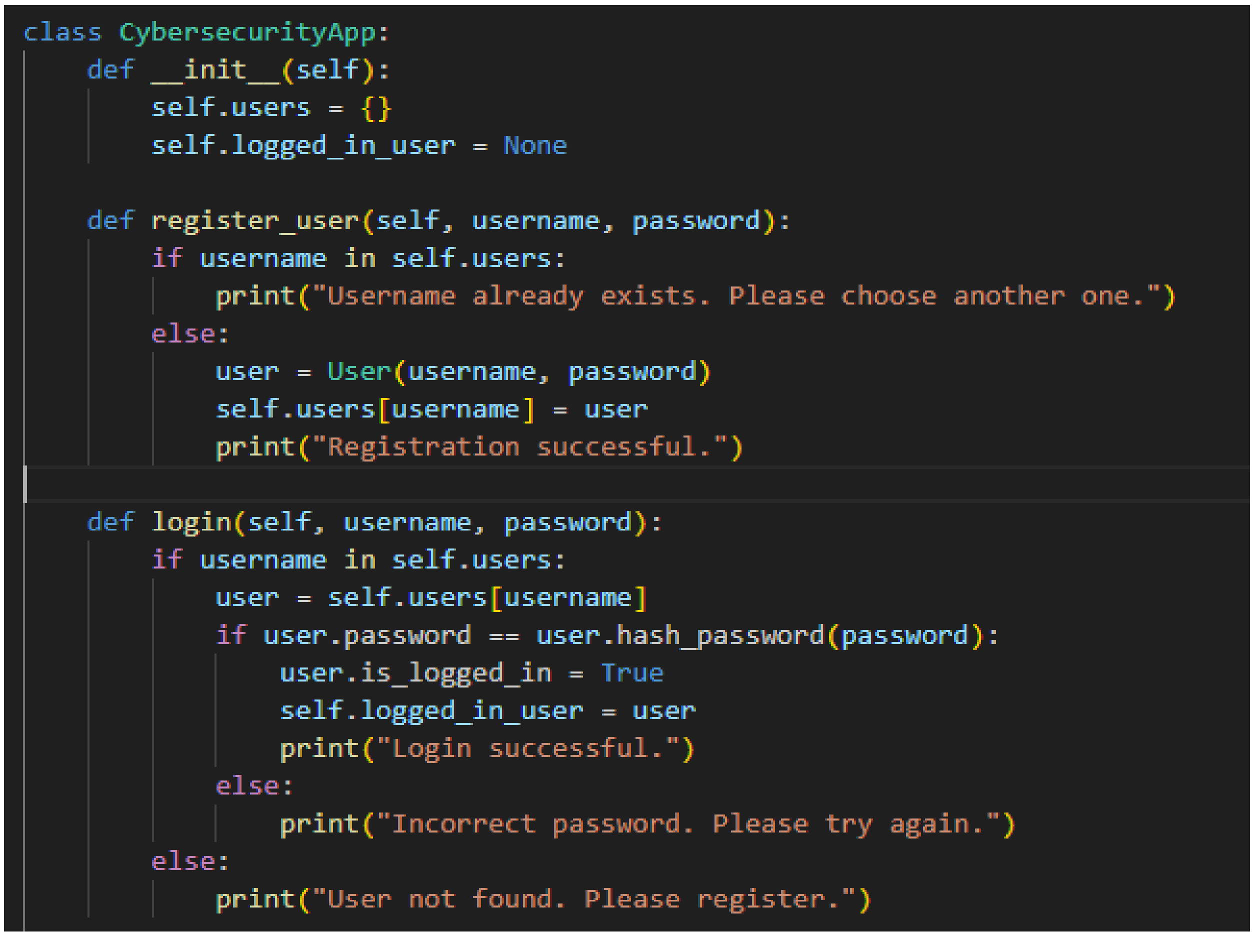
The **`User`** class is designed to represent a user in the cybersecurity application. It includes attributes such as username, hashed password, and a flag indicating whether the user is logged in. The **`hash_password`** method employs the SHA-256 hashing algorithm to securely store passwords.



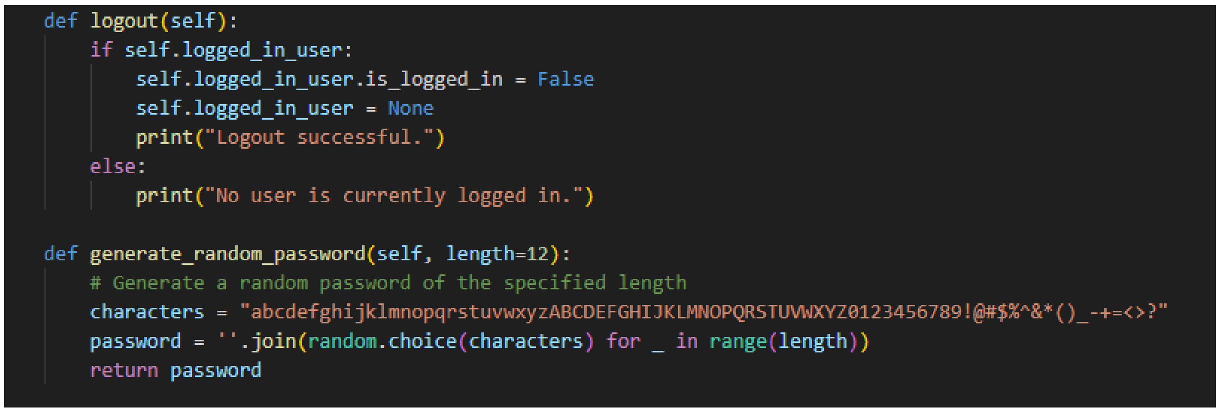
```python
class User:
    def __init__(self, username, password):
        self.username = username
        self.password = self.hash_password(password)
        self.is_logged_in = False

    def hash_password(self, password):
        # Use a strong hashing algorithm like SHA-256
        hashed_password = hashlib.sha256(password.encode()).hexdigest()
        return hashed_password
```

## 2. CybersecurityApp Class

The **`CybersecurityApp`** class acts as the main application container. It manages user registration, login, and password generation. It also maintains a list of registered users and tracks the currently logged-in user.



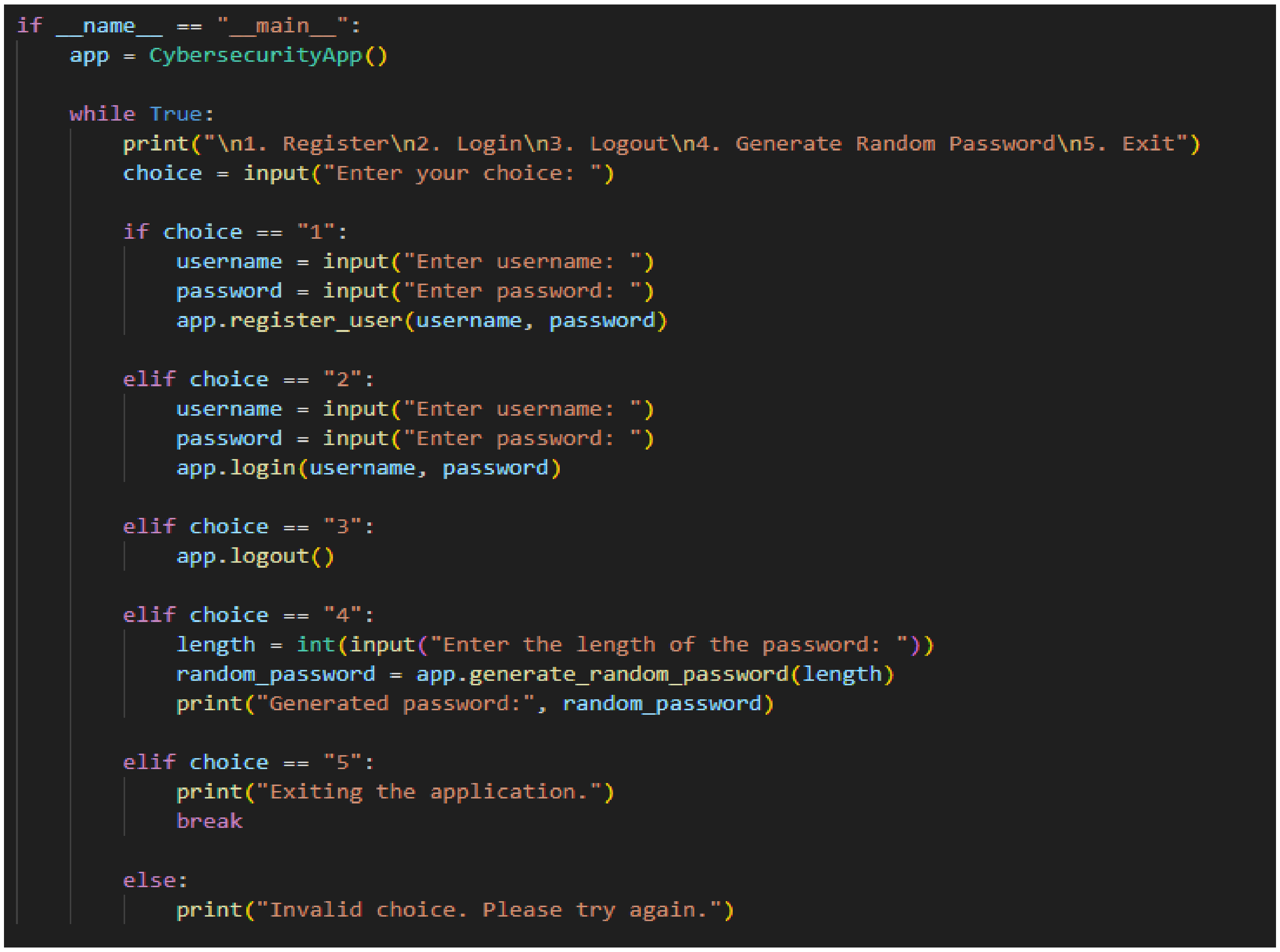
```python
class CybersecurityApp:
    def __init__(self):
        self.users = {}
        self.logged_in_user = None

    def register_user(self, username, password):
        if username in self.users:
            print("Username already exists. Please choose another one.")
        else:
            user = User(username, password)
            self.users[username] = user
            print("Registration successful.")

    def login(self, username, password):
        if username in self.users:
            user = self.users[username]
            if user.password == user.hash_password(password):
                user.is_logged_in = True
                self.logged_in_user = user
                print("Login successful.")
            else:
                print("Incorrect password. Please try again.")
        else:
            print("User not found. Please register.")
```



```python
def logout(self):
    if self.logged_in_user:
        self.logged_in_user.is_logged_in = False
        self.logged_in_user = None
        print("Logout successful.")
    else:
        print("No user is currently logged in.")

def generate_random_password(self, length=12):
    # Generate a random password of the specified length
    characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()_-+=<>?"
    password = ''.join(random.choice(characters) for _ in range(length))
    return password
```

## 3. Application Workflow

The application's workflow is structured around a menu-driven console interface. Users can register, log in, log out, generate random passwords, and exit the application. The loop continues until the user chooses to exit.

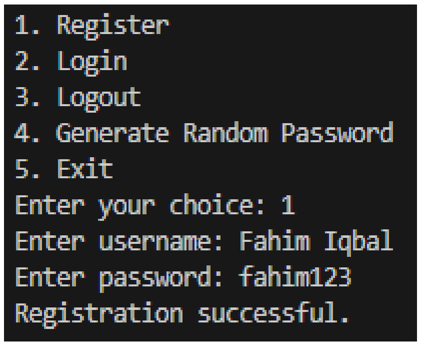


```
if __name__ == "__main__":
    app = CybersecurityApp()

    while True:
        print("\n1. Register\n2. Login\n3. Logout\n4. Generate Random Password\n5. Exit")
        choice = input("Enter your choice: ")

        if choice == "1":
            username = input("Enter username: ")
            password = input("Enter password: ")
            app.register_user(username, password)

        elif choice == "2":
            username = input("Enter username: ")
            password = input("Enter password: ")
            app.login(username, password)

        elif choice == "3":
            app.logout()

        elif choice == "4":
            length = int(input("Enter the length of the password: "))
            random_password = app.generate_random_password(length)
            print("Generated password:", random_password)

        elif choice == "5":
            print("Exiting the application.")
            break

        else:
            print("Invalid choice. Please try again.")
```

## *Results*

### 1. User Registration

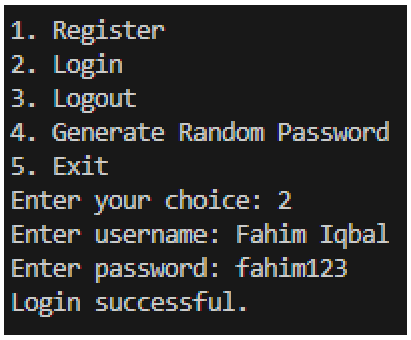
The application allows users to register by providing a username and password. The system checks for duplicate usernames and ensures that each user has a unique registration.



```
1. Register
2. Login
3. Logout
4. Generate Random Password
5. Exit
Enter your choice: 1
Enter username: Fahim Iqbal
Enter password: fahim123
Registration successful.
```

### 2. User Login

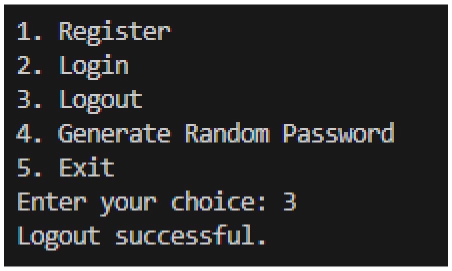
Users can log in using their registered username and password. The application verifies the entered password against the stored hashed password.



```
1. Register
2. Login
3. Logout
4. Generate Random Password
5. Exit
Enter your choice: 2
Enter username: Fahim Iqbal
Enter password: fahim123
Login successful.
```

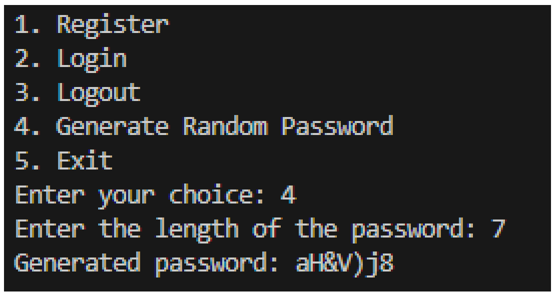
### 3. User Logout

The application supports user logout functionality, which deactivates the logged-in status for the current user.



```
1. Register
2. Login
3. Logout
4. Generate Random Password
5. Exit
Enter your choice: 3
Logout successful.
```

**4. Password Generation**
Users can generate random passwords of variable lengths using the application. The generated passwords include a mix of lowercase and uppercase letters, digits, and special characters.



```
1. Register
2. Login
3. Logout
4. Generate Random Password
5. Exit
Enter your choice: 4
Enter the length of the password: 7
Generated password: aH&V)j8
```

## *Discussion*

The cybersecurity application introduces basic concepts of user authentication and password security. The use of SHA-256 hashing for password storage enhances security by preventing plaintext password exposure. However, this lab focuses on fundamental features, and a real-world application would require additional security measures.

## *Future Work*

Future improvements to the cybersecurity application may include:

- Integration of more robust password hashing algorithms (e.g., bcrypt).
- Implementation of secure communication protocols.
- Addition of advanced cybersecurity features such as intrusion detection and vulnerability scanning.

## *Conclusion*

This lab successfully implemented a basic cybersecurity application in Python, providing essential features for user registration, login, and password generation. The application lays the foundation for more advanced cybersecurity practices and can serve as a starting point for future enhancements.