Task 3: Secure Coding Review

1. Selected Programming Language and Application

Language: Python
Application: Simple user login system with username & password validation.

2. Insecure Code Example

```
# Insecure login system (vulnerable code)
users = {
    "admin": "admin123",  # Password stored in plain text ■
    "user": "password"    # Weak password ■
}

def login(username, password):
    if username in users and users[username] == password:  # Direct string comparison ■
        print("Login successful!")
    else:
        print("Invalid credentials")

# Example login
login("admin", "admin123")
```

3. Identified Security Vulnerabilities

1. Passwords stored in plain text.
2. Weak/guessable passwords.
3. No account lockout mechanism (vulnerable to brute-force attacks).
4. Direct comparison without hashing.
5. No logging/auditing of failed login attempts.

4. Recommendations and Best Practices

1. Store hashed + salted passwords using bcrypt.
2. Enforce strong password policies.
3. Implement account lockout after repeated failures.
4. Use logging and monitoring for suspicious login attempts.
5. Avoid hardcoding credentials in the source code.

5. Secure Code (Remediated Version)

```
import bcrypt

# Store hashed passwords instead of plain text ■
users = {
    "admin": bcrypt.hashpw(b"Admin@123", bcrypt.gensalt()),
    "user": bcrypt.hashpw(b"User@456", bcrypt.gensalt())
}

def login(username, password):
    if username in users:
        # Verify password using bcrypt ■
        if bcrypt.checkpw(password.encode(), users[username]):
            print("Login successful!")
            return
    print("Invalid credentials")

# Example login
```

```
login("admin", "Admin@123")
```

6. Documented Findings and Suggested Remediation Steps

Findings:

1. Passwords stored in plain text.
2. Weak password policy.
3. Missing brute-force protection.
4. No secure authentication mechanism.

Remediation Steps:

1. Hash + salt passwords using bcrypt.
2. Enforce strong password rules (min length, complexity).
3. Add account lockout after 3–5 failed attempts.
4. Implement secure logging of failed attempts.
5. Store credentials in a secure database, not in code.