Question 3

An online banking application requires users to log in securely and protect their financial data during transactions. The system employs an authentication algorithm that uses hashing for password storage and encryption for transmitting sensitive transaction details.

a) Develop a pseudocode algorithm for a login_user function that takes username and password as inputs (5 marks)

b) Analyse the security of the application by evaluating how the CIA Triad is addressed or potentially compromised in the context of user authentication and data transmission. For each aspect of the CIA Triad, provide a specific example relevant to the online banking application. (6 marks)

c) Recommend **two distinct strategies** to further enhance the data security of this online banking application, beyond just hashing and basic encryption. Justify each recommendation by explaining how it would mitigate a specific data security threat and improve the application's overall security posture. (4 marks)

Question 3 Marking Guide

Part	Marks	Criteria	Evidence of achievement
A	1	Input handling	login_user function defined with username and password as inputs.
	1	Password hashing	Applies HASH(password) before any credential check.
	1	Modularisation	Uses a separate check_login function to perform stored credential verification.
	1	Credential verification	check_login loops through stored records and checks both username and hashed_password.
	1	Return values	login_user returns TRUE for success and FALSE for failure.
B	1	Confidentiality example	Explains how encryption protects sensitive data in transit (e.g., TLS securing transactions).
	1	Confidentiality risk	Identifies a risk, such as man-in-the-middle attacks if encryption is weak.
	1	Integrity example	Explains use of hashing or digital signatures to prevent tampering with login data or transactions.
	1	Integrity risk	Identifies a risk, e.g., if hashing algorithm is outdated (MD5 collisions) leading to compromised integrity.
	1	Availability example	Explains how redundancy or uptime policies ensure users can access banking services.
	1	Availability risk	Identifies a risk, e.g., denial-of-service (DoS) attack reducing service accessibility.
C	1	Strategy 1 identified	Suggests a specific improvement (e.g., multi-factor authentication).
	1	Strategy 1 justified	Explains how it mitigates a security threat (e.g., stolen passwords) and strengthens login security.
	1	Strategy 2 identified	Suggests a second distinct improvement (e.g., intrusion detection system, role-based access control, end-to-end encryption).
	1	Strategy 2 justified	Explains how it mitigates a security threat (e.g., insider misuse, network attack) and improves overall posture.

Sample Responses

Part A

```
FUNCTION login_user(username AS STRING, password AS STRING) RETURNS BOOLEAN
  DECLARE hashed_input AS STRING
  SET hashed_input = HASH(password)

  IF check_login(username, hashed_input) THEN
    RETURN TRUE
  ELSE
    RETURN FALSE
  END IF
END FUNCTION


FUNCTION check_login(user AS STRING, hashed_pass AS STRING) RETURNS BOOLEAN
  // stored_users is a list of records {username, hashed_password}

  FOR EACH record IN stored_users DO
    IF record.username = user AND record.hashed_password = hashed_pass THEN
      RETURN TRUE
    END IF
  END FOR

  RETURN FALSE
END FUNCTION
```

Part B

- Confidentiality
  - Addressed: TLS 1.3 encrypts login and transaction data in transit. Example: card number and transfer amount are encrypted between app and server.
  - Risk: if certificate pinning is not enforced, a man-in-the-middle could terminate TLS with a rogue certificate on a compromised network.
- Integrity
  - Addressed: HMAC or digital signature over transaction payloads prevents tampering. Example: server verifies HMAC before executing a funds transfer.
  - Risk: if nonces or timestamps are absent, an attacker could replay a previously signed request and duplicate a payment.
- Availability
  - Addressed: rate limiting, WAF, auto-scaling and failover keep login and payments responsive under load. Example: burst of login attempts is throttled without taking down the service.
  - Risk: volumetric DDoS or credential-stuffing storms could still exhaust resources and block legitimate users if upstream protection is weak.

Part C

- WebAuthn (passkeys) multi-factor authentication
  - Mitigates: phishing, credential stuffing, reused or stolen passwords.
  - Why it helps: private keys never leave the device, authentication is origin-bound, and a server-side breach of hashed passwords alone is insufficient to log in.
- Hardware-backed key management with envelope encryption
  - Mitigates: database exfiltration and insider key misuse.
  - Why it helps: store master keys in an HSM or cloud KMS, rotate keys regularly, and encrypt sensitive fields (e.g., account numbers, payee details) with data-encryption keys wrapped by the HSM-held master. Stolen database records remain unintelligible without access to the HSM-protected keys.