User Authentication and JWT

Uploaded image
answer these according to the marks beside them

Here are the answers structured according to the mark distribution:

3(a) Five Factors Used for User Authentication in Web Applications (10 marks)

The five common factors for user authentication in web applications are:

1. **Knowledge Factor** (Something you know): Examples include passwords, PINs, or answers to security questions. This is one of the most widely used methods in web applications.

2. **Possession Factor** (Something you have): Examples include OTP (One-Time Password) sent to a registered mobile phone, smart cards, or a security token device.

3. **Inherence Factor** (Something you are): Examples include biometric data such as fingerprints, facial recognition, or retinal scans.

4. **Location Factor** (Somewhere you are): This involves using the user's geographic location to authenticate, like IP address or GPS location, and is often used as a secondary check.

5. **Time Factor** (When you are): Authentication is sometimes restricted to a specific time range, such as business hours or within a limited time window after requesting access.

If allowed only one-factor authentication:
If I were to choose only one factor, I would choose the **Knowledge Factor** (e.g., password) because it’s the most commonly implemented and simplest to deploy in web applications. Although it is less secure than multi-factor authentication, it provides a balance of security and usability for general applications.

3(b) Structure of JSON Web Token (JWT) and Its Use for Verification (9 marks)

A JSON Web Token (JWT) is a compact, URL-safe token used for token-based authentication. It consists of three parts:

1. **Header**: The header contains metadata about the token, typically specifying the token type (JWT) and the signing algorithm (e.g., HMAC SHA256).

2. **Payload**: This is the body of the token, which includes claims. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims:

- **Registered claims** (e.g., `` `iss` `` for issuer, `` `exp` `` for expiration time)
- **Public claims** (custom claims defined in your application)
- **Private claims** (agreed upon between parties sharing the token)

3. **Signature**: The signature is created by encoding the header and payload, then signing it with a secret key or a public/private key pair (if using RSA). The signature helps to verify the integrity and authenticity of the token.

How JWT is Used for Verification: When a server receives a JWT, it verifies the token's signature using the same key used to sign it (for HMAC) or the public key (for RSA). If the signature matches, the token is valid, and the server can trust the claims in the payload for authentication. This enables secure stateless authentication without needing to store user session data on the server.