Authentication, JWT & Password Hashing - Theory Guide

Authentication vs Authorization

- **Authentication**: Confirms a user's identity (e.g., login with email/password)
- **Authorization**: Confirms what a user is allowed to do (e.g., access admin routes)

What is a Token?

- A token is a string that identifies a user once they are authenticated
- It is typically sent in headers to allow secure API access
- Tokens help build stateless authentication (no server session storage)

What is JWT (JSON Web Token)?

- JWT is a compact, self-contained token format
- Contains three parts: Header, Payload, and Signature
- Commonly used for authenticating users in modern web APIs

JWT Structure

- **Header**: contains algorithm and type (e.g., HS256)
- **Payload**: contains user data (e.g., user ID, email)
- **Signature**: verification hash using secret key

How JWT Works

1. User logs in and receives a JWT signed with a secret key
2. Client stores the token (e.g., localStorage or cookies)
3. Client sends token in `Authorization: Bearer <token>` header
4. Server verifies the token using the secret key

JWT Example

```js
jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: '1d' });
```

What is Bcrypt?

- Bcrypt is a password hashing algorithm
- It converts plain-text passwords into secure hashes

- Hashes are irreversible (cannot be decrypted)
- Uses 'salting' to make hashes more secure

How Bcrypt Works in Practice

- Before saving a password:

```js
user.password = await bcrypt.hash(user.password, 10);
```

- During login:

```js
bcrypt.compare(inputPassword, storedHash);
```

Why Hashing Passwords is Important

- Prevents plain-text password storage
- Even if the DB is hacked, real passwords stay hidden
- Adds a layer of cryptographic protection

Storing and Verifying Passwords

- Store only hashed versions (never raw passwords)
- Use `comparePassword` method to check user input

Best Practices for JWT + Bcrypt

- Always use environment variables for secrets
- Set token expiration times (e.g., 1h, 1d)
- Use HTTPS in production to protect tokens
- Never log or expose raw password data
- Hash passwords before storing
- Use `bcrypt.compare()` instead of manual matching

Recap: Key Concepts

- Token = string for identifying user
- JWT = signed, verifiable token containing user data
- Bcrypt = secure password hashing tool
- Authentication = 'Who are you?'
- Authorization = 'What are you allowed to do?'