Tirth Shah
Fullstack Developer

Authentication and Authorization

So, in this section, you learned that:

- **Authentication** is the process of determining if the user is who he/she claims to be. It involves validating their email/password.

- **Authorization** is the process of determining if the user has permission to perform a given operation.

- To hash passwords, use **bcrypt**:

```
// Hashing passwords
const salt = await bcrypt.genSalt(10);
const hashed = await bcrypt.hash('123456789', salt);

// Validating passwords
const isValid = await bcrypt.compare('123456789', hashed);
```

- A **JSON Web Token (JWT)** is a JSON object encoded as a long string. We use them to identify users. It's similar to a passport or driver's license. It includes a few public properties about a user in its payload. These properties cannot be tampered because doing so requires re-generating the digital signature.

- When the user logs in, we generate a JWT on the server and return it to the client. We store this token on the client and send it to the server every time we need to call an API endpoint that is only accessible to authenticated users.

- To generate JSON Web Tokens in an Express app use **jsonwebtoken** package.

```
// Generating a JWT
const jwt = require('jsonwebtoken');
const token = jwt.sign({ _id: user._id}, 'privateKey');
```

- Never store private keys and other secrets in your codebase. Store them in environment variables. Use the **config** package to read application settings stored in environment variables.

- When appropriate, encapsulate logic in Mongoose models:

```
// Adding a method to a Mongoose model
userSchema.methods.generateAuthToken = function() {
}
const token = user.generateAuthToken();
```

- Implement authorization using a middleware function. Return a 401 error (unauthorized) if the client doesn't send a valid token. Return 403 (forbidden) if the user provided a valid token but is not allowed to perform the given operation.

- You don't need to implement logging out on the server. Implement it on the client by simply removing the JWT from the client.

- Do not store a JWT in plain text in a database. This is similar to storing users' passports or drivers license in a room. Anyone who has access to that room can steal these passports. Store JWTs on the client. If you have a strong reason for storing them on the server, make sure to encrypt them before storing them in a database.