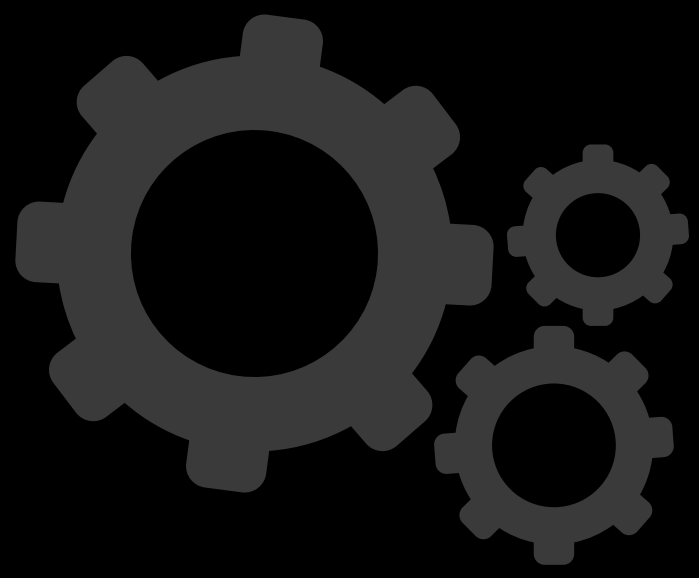
Lucky Jain

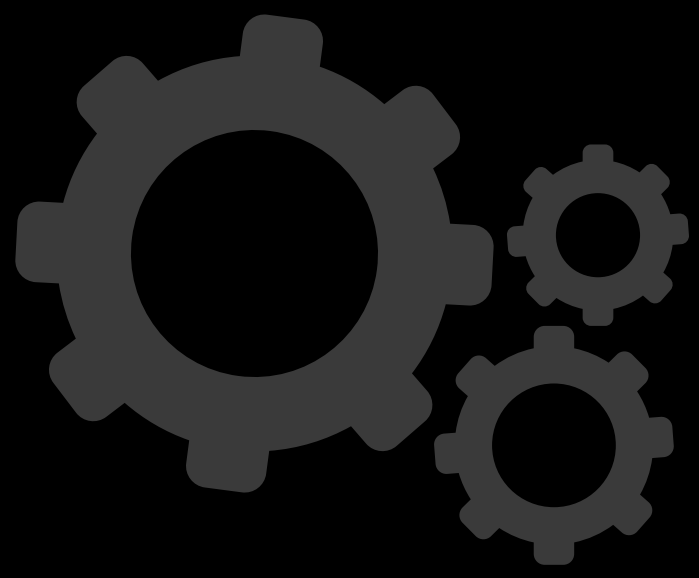


S.O.L.I.D

Open - Closed Principle

Swipe →

Lucky Jain



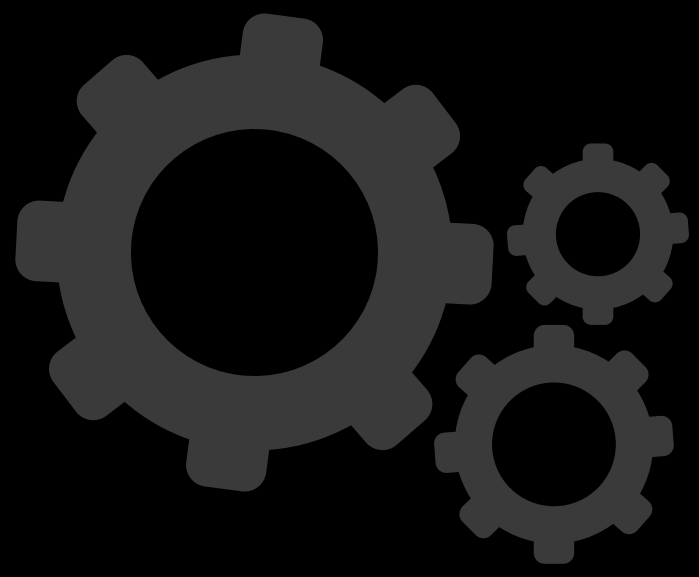
Open - Closed Principle

Software entities should be open for extension but closed for modification.

The open-closed principle states that when you define software entities, you should be able to extend their functionality, but you should not be able to modify the existing entity. Instead, you should just add a new mapping or configuration that allows the right strategy to be applied when running it.

Swipe 

Lucky Jain



Consider a User model

```typescript
class User {
    private username: string;
    private email: string;
    private password: string;

    constructor(username: string, email: string, password: string) {
        this.email = email;
        this.password = password;
        this.username = username;
    }
```

Swipe →

Lucky Jain

Suppose we need to add a new field called Account Type

```
private accountType: Account = "Admin";
```

But what if you had to add another user type?

```
isHR(): boolean {
    return this.accountType === "HR"; //❌
}
```

Swipe →

Lucky Jain

Instead of that, do the following

```
getAccountType(): Account {
    return this.accountType;
}
```

Swipe →

Lucky Jain

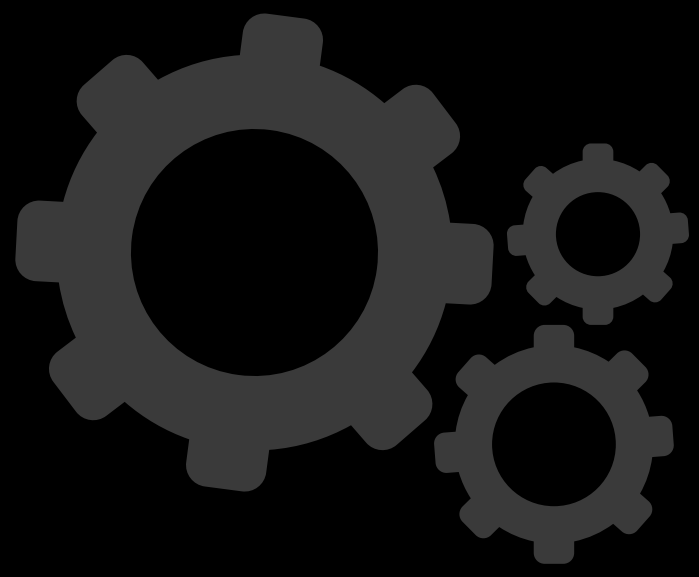
Create a roles record and Role service

```ts
const Roles: Record<Account, string> = {
    "Admin": "/dashboard",
    "Employee": "/",
    "Manager": "/employee-list",
    "HR": "/salaries"
}

class RoleService {
    getRole(user: User): string {
        return Roles[user.getAccountType()];
    }
}
```

Swipe →

Lucky Jain

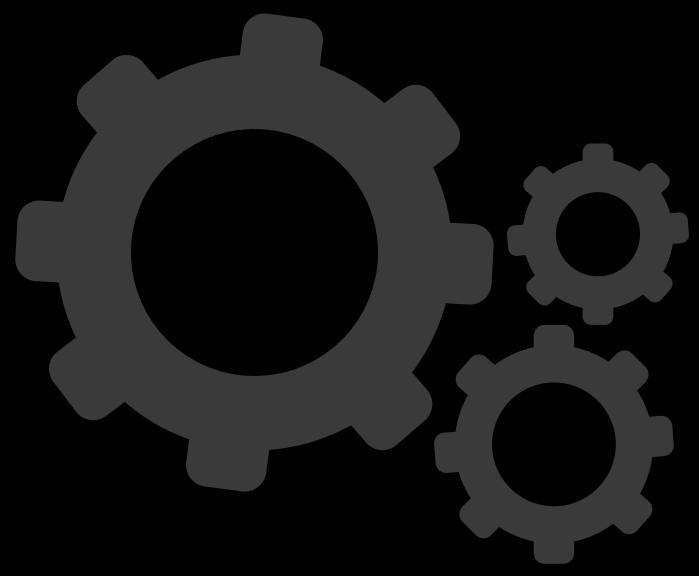


Benefits

The main benefit here is that for any similar future changes, you will have to change the code in few places in a more predictable way.

Swipe →

Lucky Jain



HAVE ANY VIEWS ON THIS?

COMMENT BELOW