## Introduction to The System

This system has been designed for the company called ‘E-shift’ which mainly focuses on shifting household items from point A to point B

## Main Users of the system

Mainly there are two types of users which mainly uses the system as follows

1. Admin User
2. Customer User

Roles of the users can be simply explained by a use case diagram as follows

As mentioned above mainly admin user is having the highest authority to make changes in the system. An admin is capable of accepting or rejecting any sort of a job and also product item CRUD operations can be conducted by the admin.

Customer responsibility is really simple. Which is to create jobs based on the his/her requirement.

## Architecture used in the system

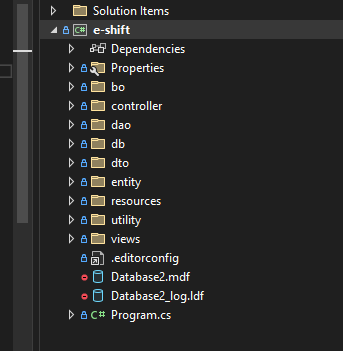
As this involced numeral number of transactions and specially repetitive crud operations I’ve decided to use the Layered/Multitier Architecture.

Mainly the layered architecture can be represented be the following diagram



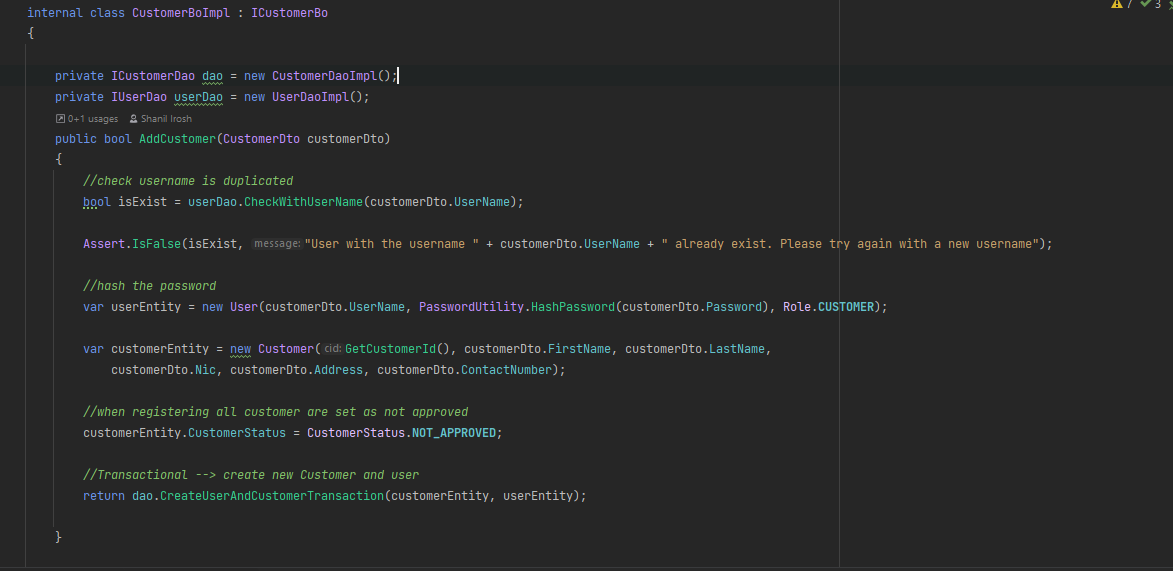
As mentioned above mainly 4 layers are used in the architecture. I have segregate the project mainly 3 parts as mentioned above

1. View (Presentation Layer)
2. Bo (Business Layer)
3. Dao (Database / Persistence Layer)



# Layers with Code Examples

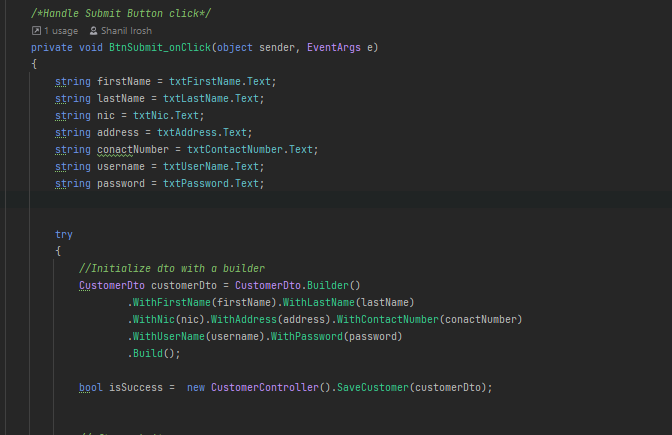
**Business Layer (BO Layer)** – The business logic of the business is conducted here. Validation such as checking of duplicates, Generating unique ID’s are conducted here. The only purpose of this layer is to manipulate the business logic. Following is an example of how I used this layer.

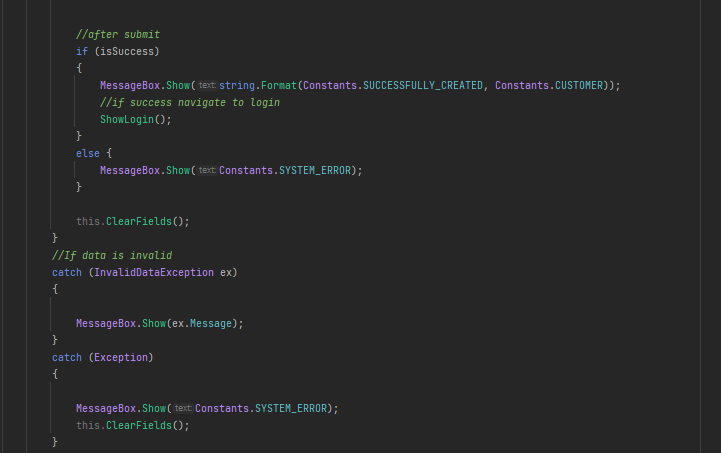


As shows above all the related logics such as checking whether username is already there in the system, hashing of passwords, approval statuses are added in this layer. And if there are any mismatches then exceptions are thrown from this layer to the presentation layer and displayed.

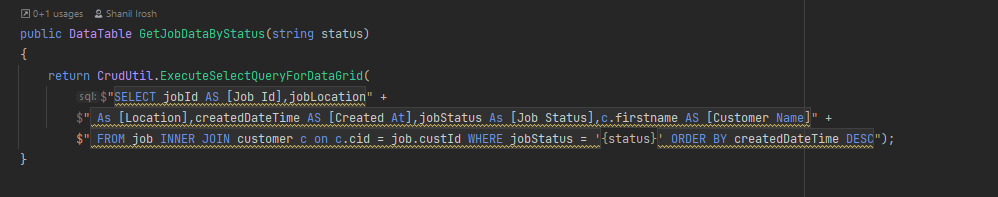
**Presentation Layer** – Mainly responsible for retrieving the dataset from the client side. And the UI validations such as Empty textbox, submitting text values for numerical inputs is conducted by the layer. In this project the views sections and the controller section represents the presentation layer. This layer has no idea about the business data. And all the exceptions thrown from the other layers are captured here and displayed to the user.

Exception handing is mentioned in the below Exception handling section





**Database Layer (Dao Layer)** – Mainly the communication with the database happens here. This layer has no idea about the business logic. This layer just executes the database queries and retrieves the output. And transaction management activities are also conducted here



## Data Flow Diagram

## Benefits of using the Layered Architecture

Really easy to maintain as all layers are segregated separately

Testing is very easy as separate components are maintained

Dependency is reduced as each layer is functioned separately. Eg. A front end expert can simple just have to keep on working the presentation layer.

UI changes or UI framework switches can be done very easily as all the business login and data layer is segregated. Eg. If the business want to develop a mobile application it can be done very simple as only the presentation layer is there to be changed.

## ER diagram of the business

Mainly I have used following tables in order to conduct persistence activities

* UserData Table – Responsible to keep the username,roles and passwords of an user. If the user is a customer the the primary key of that customer is kept in the user table as a foreign key in order to avoid duplication
* Customer Table – This table keeps the master records of a customer
* Job Table – This keeps the master record of a job such as location,created date etc. The primary key of the customer is maintained as a foreign key in this table in order to trace the customer who placed the job