# CS307 Project 2

**Contributor**:

- 戴郭轶 12011211
- 周凡卜 12012519

# Task 1: Overall introduction

We use springboot as our blackened to connect to the database and offer http/https serve. The optimization of `Thread Pool` and `DataSource Connection Pool` enables us to serve over `70k` requests in `one minite`. Also, we provides `User-friendly interface` to react with our users and `perfect authority management system` to control the permission of each user.

# Task 2: API Design

For all data, we just receive a parameter from `dto`, so all the parameters can be null for any opration unless required.

## 1. Basic Api

| Api (All operations) | Required parameters(DTO object) |
| --- | --- |
| Center | Center center |
| Staff | Staff staff |
| Model | Model model |
| Enterprise | Enterprise |

We support all basic table select, update, insert and delete operations. We set a while list for the parameters in case of SQL injection.

All operations are based on the `dto` objects, so the can set the parameters with `js` functions and pass the `json` object to the backened and auto transformed them into the required format. Before every operation, we do checks to the data and ensure them be valid.

We return a `json` containing all the operation result and the total `cost` time.

## 2. Stock In

All we need is a `json` object and it will be decoded as follows:

| Requirement parameters | Required type | Is Must |
| --- | --- | --- |
| supplyCenter | String | Yes |
| productModel | String | Yes |
| supplyStaff | String | Yes |
| date | String | Yes |
| price | String | Yes |
| quantity | String | Yes |

We allow our users to pass different types of date to our servers and we can dealt with it successfully.

In this part, we firstly check whether all data exsits and then we check whtther the staff is valid whether the staff is supply staff and belongs to the supply center in data. Then we use a procedure to add the stock and add a record for it.

We return the `json` about thether the insert is successful, if not , we return the reason.

## 3. Place Order

All we need is a `json` object and it will be decoded as follows:

| Requirement parameters | Required type | Is Must |
| --- | --- | --- |
| contractNum | String | Yes |
| enterprise | String | Yes |
| productModel | String | Yes |
| quantity | String | Yes |
| contractManager | String | Yes |
| contractDate | String | Yes |
| estimatedDeliveryDate | String | Yes |
| lodgementDate | String | Yes |
| salesmanNum | String | Yes |

In this part, we just need to check the stock of the center when we place order and whther the staff has the right to sell things.

When we insert the contract, we first check if the contact exsit, if not, we will insert a new contract first and then insert the content.

We return the `json` about thether the insert is successful, if not , we return the reason.

## 4. Update Order

All we need is a `json` object and it will be decoded as follows:

| Requirement parameters | Required type | Is Must |
|---|---|---|
| contractnum | String | Yes |
| productmodel | String | No |
| salesman | String | Yes |
| quantity | String | No |
| estimatedeliverydate | String | No |
| lodgementdate | String | No |

We will update all orders meet the requirement.

We check the order whther belongs to the salesman and update stock, as a result of involving triggers in this part,updating stock becomes really easy.

We return the `json` about thether the insert is successful, if not , we return the reason.

## 5. Delete Order

All we need is a `json` object and it will be decoded as follows:

| Requirement parameters | Required type | Is Must |
|---|---|---|
| contractnum | String | Yes |
| salesman | String | Yes |
| sequence | String | Yes |

When selecting the orders of a saleman, we order it by the required order. Then we can check the validity of the data. The same reason as update order, update stock number becomes a really work in this part.

We return the `json` about thether the insert is successful, if not , we return the reason.

## 6. 6 only selection api

We just did as what mention in the api document using view function.

We return a `json` object containing the result and decode it in frotened.

## 7. Get Product By Number And Get Contract Info

All we need is a `json` object and it will be decoded as follows:

| Requirement parameters | Required type | Is Must |
|---|---|---|
| contractnum | String | Yes |

We use a view function that return all we need to help us do join selections in 3 different tables.

We will display all result in frotened.

# Task 3: Advance Part

## 1. Connection to DataBase

Thanks to spring boot, we only add the connection information in `application.yaml` and spring boot initialize the connection source for us.

## 2. Enhance the Section2

All operations involves multiple parameters we allow our users to input requirements flexible. And all our tables allows users to selction if he/she has been authorithed. Also, we allow admins to check all operation logs to all table.

## 3. Http/Restful Web Service && Gui

We provides our users a human-friendly frotened and connect with the backened with http connections.

## 4. User Privilege

We can easily set user's privilege to different operations using cookies to specify the user's identification and ban some usages of them.

## 5. DataBase User && Procedures && Views && Functions && Triggers

We set different users in our database and has set the `secienaroes tests`. We use `procesures` in `stock in` and `place order` functions to simplify the operations. We use `views` to simplify the `selection` operation and reduce the bandwidth between server and fatabase to serve more users at one time. The `log` funtion was realized by `triggers` and `functions` to record and stored in a new table.

## 6. Muiti-Thread

We set a `thread pool` to deal with the high load. And in the stress test, we have served over `70k` requests in one minute.

## 7. One Key Import For All Apis

All we apis allow us to import test data when upload a `csv` file to realize one key import. All file process uses multi-thread to accelerate the speed.

In this part, we will return the numbers of successes and failures, the operation cost time. The failure infomation is listed in the console.