<CAU Plate>  
Design Specific

Web programming

Hee Chan Jang 20132180

Min Ji Kim 20153029

Yoon Ju Lee 20150798

Seok Ho Shin 20144051

# Module

Module is a piece of program that specifies how other pieces are related on other pieces and which functionality it provides for other modules to use (its interface). By restricting the ways in which modules interact with each other, the system becomes more like LEGO, where pieces interact through well-defined connectors, and less like mud, where everything mixes with everything.

Our Project has 3 Module

At the top, Status bar

Search bar

Insulted map from existing websites

Our project is to make a web page which is to serve good restaurant. We did not want to create a one-way internet web page. We want interactive path. We make a home page. Then if you want go to other page, just click upper Statusbar. At the top of the page, we put a Statusbar. We have created a status bar where you can go to another page whenever you want. And every page have that statusbar.

So we make the statusbar to Module. That module provides statusbar for every page. That bar has always link to other page. The bar is static. Not changeable. The bar provides link to main page, ‘Category’ page, ‘Location’ page, ‘Recommend’page. So In any page, you will go to page that bar provides.

And that bar has other function. That is search bar. If you want to go to concrete restaurant page, you will use the search bar. That bar is included by status bar. So simple if you want go other page, just put that name. It will go directly to that page.

Example code)

<!-- #region 네비게이션 메뉴 -->

<div class="top\_nav\_wrapper">

<ul class="top\_nav">

<li id="mnu\_best">

<a href="/best/best\_list.aspx">Category</a>

</li>

Finally, a map of Naver and other search engines is put on each restaurant page. To do this, we need to get api from Naver. On the next page after api, mark Naver map with markers. That way you can easily put the map of the restaurant you want, rather than simply putting the map image. And it is easy to use map by linking with Naver map...

Example code): <

div id="map" class="map">

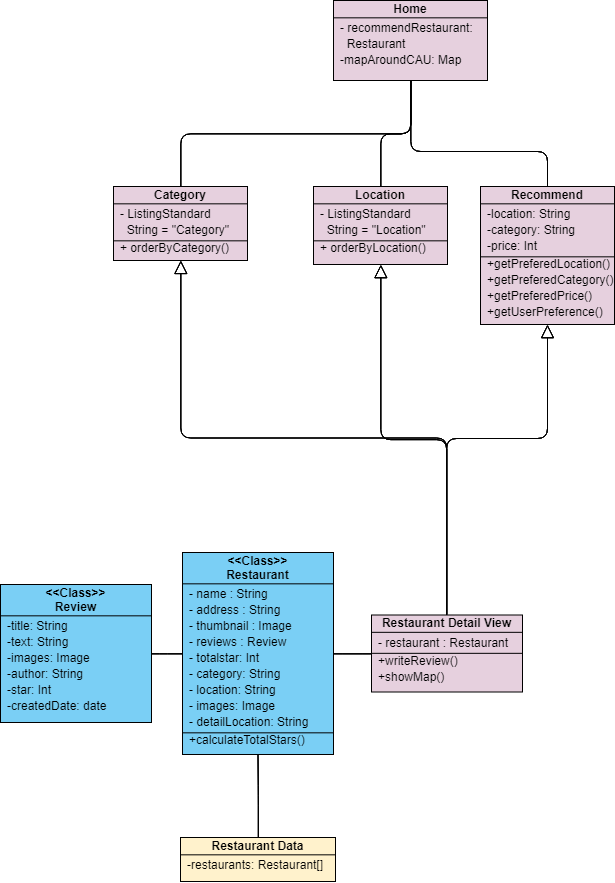
</div>

<script type="text/javascript" src="https://openapi.map.naver.com/openapi/v3/maps.js?clientId=자동으로 생성되는코드&submodules=geocoder"></script>

<script>

var CustomOverlay = function(options) {

# Classes



## Classes explanation

Our project’s class diagram looks like the image below. Since none of our team members learned how to draw class diagrams, our team tried best but the result is quite simple.

Purple classes mean pages and blue classes mean data class. Our project, CAUPLATE has three main page and one detail page.

First main page is home page. We will show large map using google maps api around CAU that contains markers about restaurants. If user click the marker, it will show the restaurant’s name and simple information. We will also show recommending restaurants in our database randomly in home page.

Second main page is Listed page. There are two listed pages based on different standard, category and location. In category, there are “한식”, “일식”, “양식”, “기타”. In location, there are “정문”, “중문”, “후문”. We will show listed page of restaurants that categorized by their features.

Third main page is recommended page. This helps user to choose restaurant. It collects data of user’s preference by asking them questions. Through those data, CAUPLATE will show recommending restaurants.

Restaurant detail view page is showed when users click certain restaurant in listed page or recommended page. People can view about restaurant’s detail, write reviews and see its location through map. Restaurant’s information comes from the data of restaurants, which is collected by team members. It will be organized as class Restaurant.

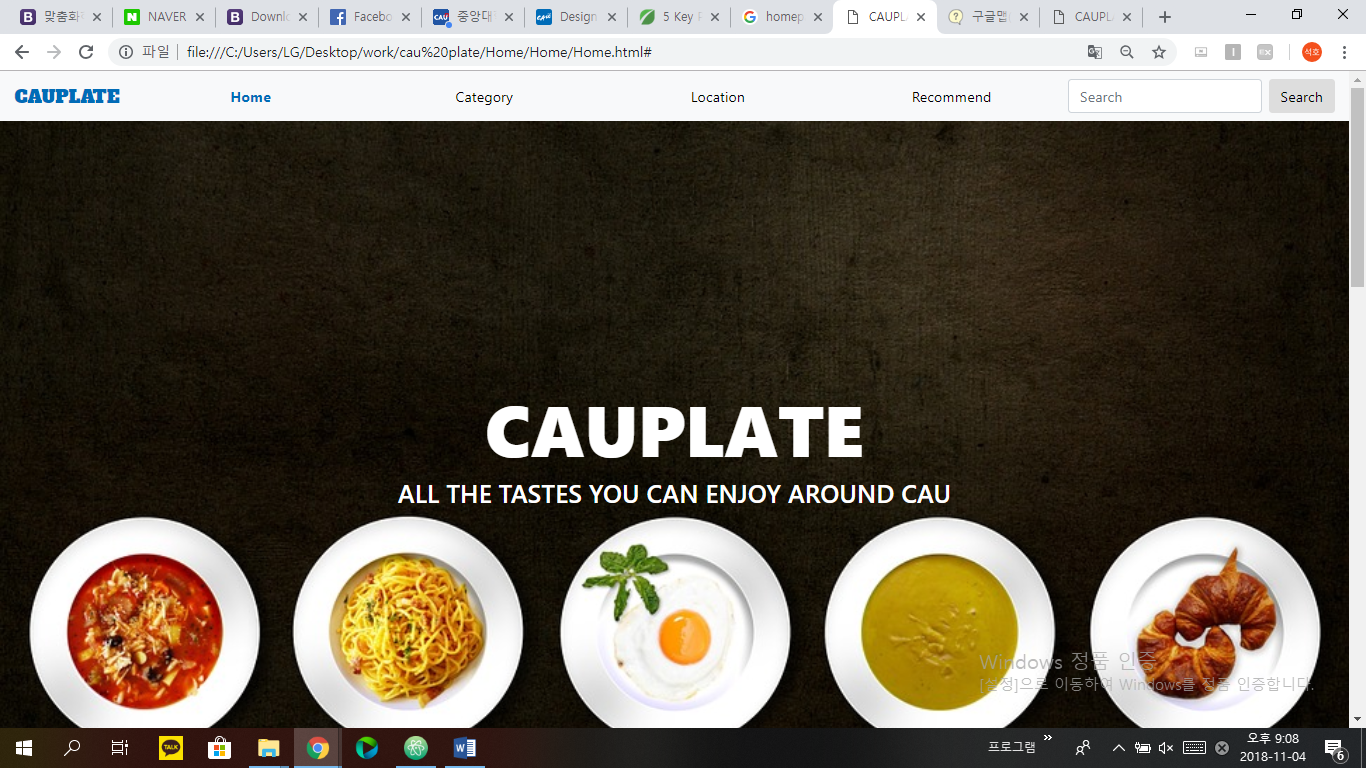
There are two data classes, Restaurant class and Review class. Restaurant class contains information of restaurant that are used for listing and detail pages. Review class contains basic information of reviews and is contained in Restaurant class.

# USEr Interface

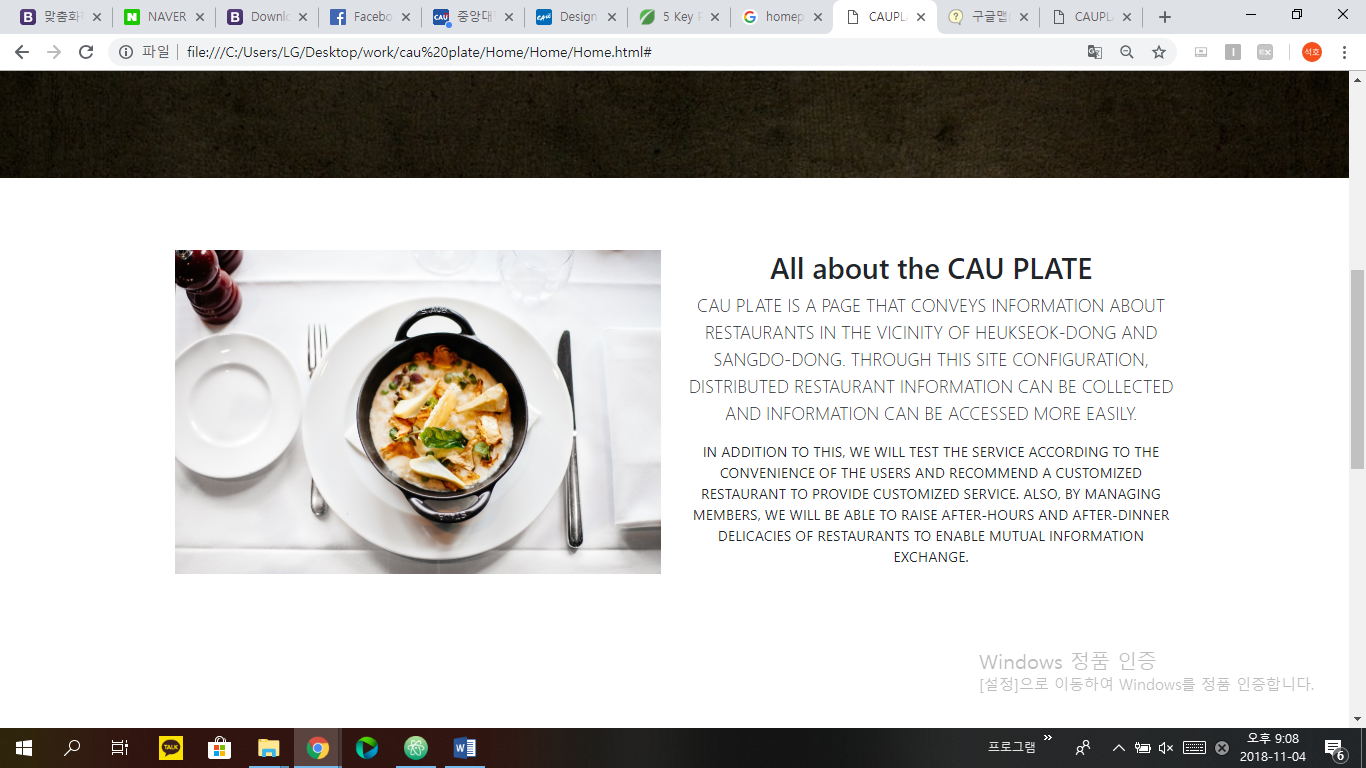
## UI Requirement

|  |  |
| --- | --- |
| **Requirements** | **Specific** |
| User Experiences | * **Important on top**: Put the information that users want on the top. * **Live page** : Make page so that user can feel like they are experiencing the restaurant it self |
| Simplicity | * **Picture**: Use as much picture as possible to describe. * **Word**: Use least words but if you have to use word, it must be key word. |
| Scan able | * **Lengthening the pages and including less information to digest at once**: As the user scrolls, they are taking in more supplementary information. |

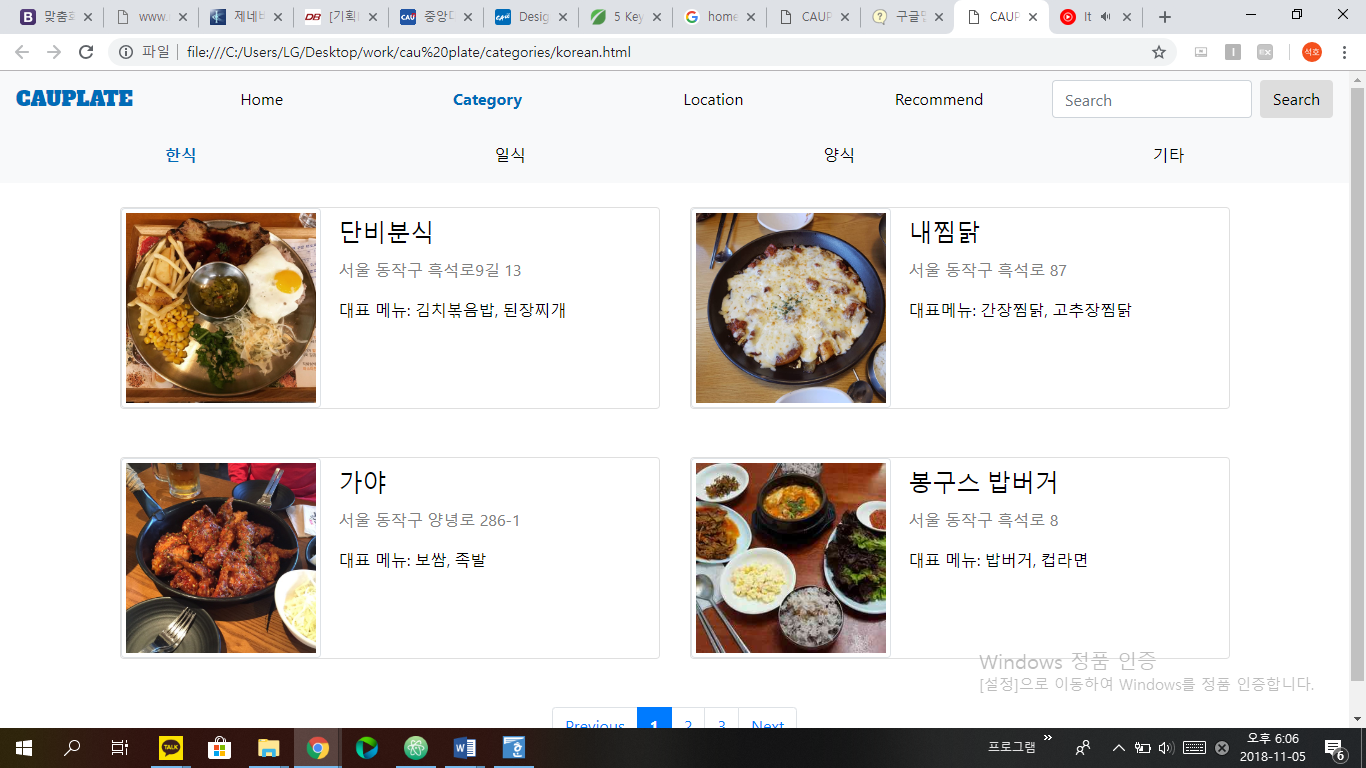
* The Home page



This is the homepage of CAU PLATE. First, the home logo and each menu are implemented through simple navigation at the top. Next to that, we put the search module and focused all the functions that users wanted at the top of the page. This will make it easy for users to experience the page and it is simple to see at a glance. It also makes you feel confident that when users enter this page, users will be able to realize that they have come to the right site through a visible logo on a large photo. Furthermore this black-and-food photo can fascinate customers who come in accidentally or incidentally, and prolongs their stay on the page. In particular, the CAU PLATE page name and simple catch phrases are used to narrate the precise and accurate sense of purpose of this page.



# At the same time, I tried to keep the simplicity by limiting the cuts that can be seen in one field of view as much as possible. Instead, the insufficient information was replaced by making the scrolling relatively long. Along with this, it makes it easy to notice what it means to convey intuitive pictures in every article or word, without reading the information in detail.



# On the page that shows information of specific restaurants, 4 restaurants were arranged on one page, making it easy and intuitive for users to view and explore. The restaurant's name, location, and representative menu are created in the sequence so that information can be viewed at a glance. On top of that, the navigation bar is continuously positioned so that the USER can select and browse the desired information, and the number of pages is set below to make it possible to view the following information. The photos were set to thumbnail so that they could display food at the right size.

# DATA SCHEMa

## DATA SCHEMA EPLANATION

CAU PLATE is a web page that provides the following services:

- Search restaurants

- Show restaurants by categories, locations

- Post reviews

- Match recommended restaurants

CAU PLATE Database consists of 5 relations to provide that services: *restaurant, menu, user, review, recommend*. *Restaurant* relation has 4 attributes. ID attribute is a primary key of this relation. The others represent name, location, category of restaurants. *Menu* relation consists of ID, dish, price of restaurant. ID is a foreign key referencing *restaurant* relation and (ID, dish) is a primary key of this relation. *User* relation consists of ID and name attributes. Primary key of this relation is ID attribute. *Review* relation consists of user\_ID, restaurant\_ID, restaurant\_name, review\_contents, star\_rating. User\_ID is a foreign key referencing *user* relation and restaurant\_ID is a foreign key referencing *restaurant* relation. (user\_ID, restaurant\_ID) is a primary key of this relation. Review contains text, images, and star rating. *Recommend* relation consists of ID, category, location, price attributes. When a user takes a test, result of test are recorded at this relation. Then, web page shows recommended restaurants. ID attribute is a primary key and a foreign key referencing *user* relation.

Data schema is as follows:

*restaurant* (ID, name, location, category)

*menu* (ID, dish, price)

*user* (ID, name)

*review* (user\_ID, restaurant\_ID, restaurant\_name, review\_contents, star\_rating)

*recommend* (ID, category, location, price)

