As you can see from the above section, MVC or Model-View-Controller is a suitable design pattern for our desired system. Here, we would like to point out some important features of MVC architecture and apply them to the system:

- **Model**: contains the following class and potential table in the database
    - FoodItem: this will have a unique ID, together with a description, information about the food such as the number of addon dishes, and the number of available dishes the restaurant could serve per day.
    - Cart: this together with the cookie of a new customer will contain many FoodItem together with their corresponding price and quantity.
    - Order: contain the date the order is made, together with the generated code and the status, which will require the clerk to confirm before notifying the kitchen. In addition, orders also record the payment method and the feedback or rating of the customer.
    - Transaction: by utilizing the bank API in the controller part, we will store the transaction detail information for analysis purposes.
    - Account: a new customer coming to the restaurant doesn't need to have an account. After enjoying the meal and rating it, he/she can sign up for an account so as to receive bonuses, discounts, priority, and convenience when ordering the same meal the next time.
- **View:** the main graphical user interface to interact with the system. Here we break the view into many different view components, each for a different user
    - Customer view: will be a QR code and the website interface for the user to skim the menu, select food put into the cart. Customers can provide feedback through the website, and sign in or sign up so as to receive recommending food of the day or a discount, bonus. In addition, based on the number of available dishes, the menu could display only the food that is not out of order.
    - Clerk view: this contains the POS system interface so that the clerk can confirm the new incoming order, check if the order's dish can be served by the restaurant or not, then record the errors if it exists, and allow the clerk to input the cash if the user pays in cash.
    - Staff view: allow the staff to easily input the number of dishes that can be served by the restaurant so as the system can provide a better user experience for the customer. In addition, the system, through staff view, could inform the staff to prepare food for the incoming confirmed order.
    - Manager view: allow the manager to view the report of the restaurant finance, some graphs of trending of customer towards the food and the service.
- **Controller:** the main role of the controller is to receive user action from the view and render the corresponding view, receive notify from the change of model and update the model according to the new view. The controller will contain all the business logic flow of the restaurant.
    - Customer controller: dedicated for the customer experience, which includes automatically generating the order id, allowing customers to interact with the food cart model by adding/removing/updating the food item. Provides the sign-in/signup/log-out option for users and also performs the bank API call in order to handle credit card transactions.
    - Clerk controller: allow the clerk's view to interact with the Order model such as confirm status, update the errors, and the payment method.

    - Staff controller: allow the logical way to update the number of dishes served and notify the staff of the new order.
    - Manager controller: in charge of generating reports, involving the analyze data API, forecast API, and recommend API.

In addition to the MainStream System which utilizes the MVC design pattern, we also enhance the business logic with the Intelligent Business Assistant System which follows the pipe and filters design pattern:

- The model of the MainStream System stores all of its data inside a database. Through the raw data collected from the database, the system can perform some cleaning data tasks such as removing missing values from rating, averaging the number of dishes, etc.
- On the analyzed data branch, the system will figure out some interesting important features and display them as a graph so as to communicate the structure of the data.
- On the forecast data branch, the system will identify the favorite dish of a day, when the restaurant serves customers, etc. This will generate a quality report for the manager.
- Combining both the important features and the trending of the service, the system constructs a recommender system, which will provide to the customer, through the MainStream System some related dishes and encourage users to try more dishes. Thus, increase the user experience and the revenue of the restaurant.