Online Food Ordering System

Project Design Doc

Team 8,

Deeksha Thakur

Siddharth Saklecha

Saurabh Jain

Rajesh Dansena

1. Introduction

The project is about creating a website as a solution to online food ordering system. The architecture of this software is designed considering modularity such that each part of this software provides an API interface which will be easily portable across different platforms and reusable for generic websites.

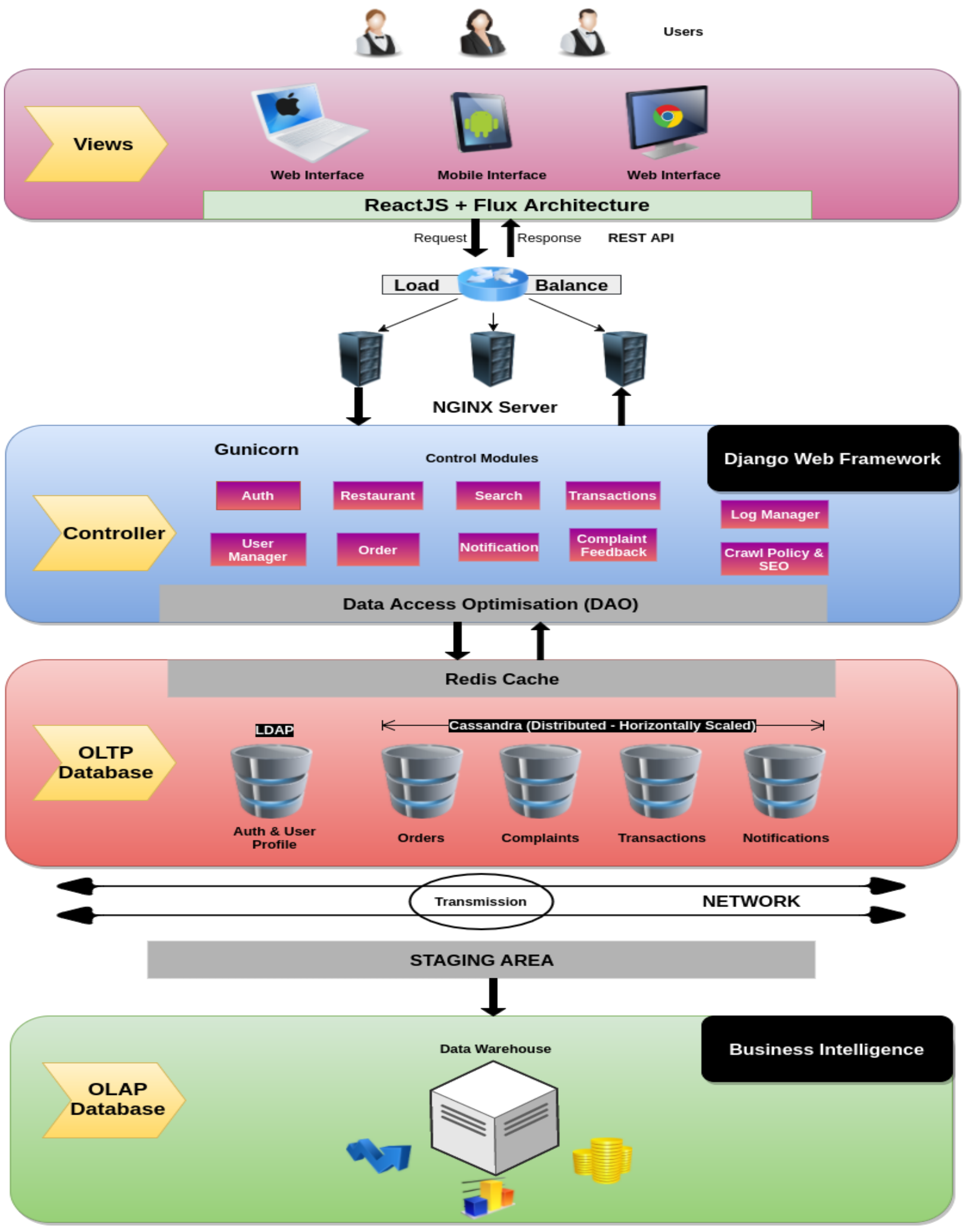
The objective of this document is to provide Architecture, High Level Design and Low Level Design of the system to give a clear image upon how the system is going to be implemented. This is the most crucial document for the developers as well as for anyone who wants to understand how this system works.

Architecture diagrams clarifies all the modules and the overall structure of the system and therefore it is the most crucial part of this document.
High level design describe each module upto a depth such that its implementation and interaction with other modules get clear.
Low level design is currently very flexible and one may expect it to change a lot as the project progresses towards implementation phase. To avoid analysis-paralysis, low level design is though of only upto a certain depth as of now.

After going through this document, one will have a clear image of this system and will be able to appreciate its architecture.

2. System Architecture

2.1 Architecture Diagram



2.2 Architecture Description

Model-View-Controller (MVC) architectural pattern is being used for this project. The pattern has been slightly modified so as to incorporate all the requirements as mentioned in SRS document. The pattern has been designed to incorporate following properties:

1. Optimised
2. Scalable
3. Modular
4. Simple

2.2.1 Views

Responsive and Reactive are two main properties which everyone expects to be present in UI/UX. We are using ReactJS (a JS library by Facebook) to create a reactive page along with Flux architecture to produce a clean data flow at client-side. It's more of a pattern rather than a formal framework.

2.2.2 Controller

We are using Django Web Framework to perform as the controller at server-side with Gunicorn as the interpreter. Django has in build DAO operations and a good library support for different kind of database protocols such as LDAP and Cassandra. Different modules have been created to fulfill a particular functionality keeping in mind high cohesion and low coupling between them. Each module provides an API interface to interact with it. Below is the description of the most important modules:

2.2.2.1 Search Module

The search module allows user to search for a specific content on the system. Customers can search for restaurants, cuisine, etc. whereas managers can search for users and orders as well.

Different types of users have different roles along with different search privileges. The advanced search will be incorporated with a number of useful filters such as search by cuisine, price, location, etc.

2.2.2.2 Notification Module

This module is responsible for generating and sending messages/emails with the help of the Notification Functionality (Push notification).

Using the activity information as the basis, system will trigger notifications that will be delivered to the customers, attendant, delivery person etc. There will be 3 types of notifications:

1. SMS
2. Email
3. App/Website notification

This module is designed in such a way that the module remains independent of the system and only serves system through APIs.

2.2.2.3 Complaint-Feedback Module

The Complaint-Feedback module is used to address complaints and resolve disputes of the users. It also allows user to give his/her valuable feedback about services and food of the restaurant.

This customer support will be served in a number of ways but will be implemented in iterations:

1. Website complaint Forum
2. Phone Service
3. Chat Bot for simple queries

2.2.2.4 Order Module

This module is responsible to create an order and generate an invoice once the order has been finalised by the customer. Once the potential customer confirms the estimate, the information is transferred to this module. This module will keep track of customer's previous order and will help in the personalisation.

2.2.2.5 Authentication Module

Central authentication system for all types of users. As the name suggests this modules authenticates a user and allows to use the system. Main functionalities:

1. User Registration
2. User Login
3. Change Password
4. Forgot Password

2.2.2.6 Offer-Discount Module

The sole purpose of this module is to allow customer to avail discounts by using the coupon code and allow manager to configure these offers.

Manager can configure any type of discount in the system in a truly intuitive manner where system allows different combinations of discount conditions and restrictions. Instead of having to memorize multiple discount types for all manner of situations, simply create a new set of discounts, or open up an existing discount and apply the required conditions to get it up and running.

This module also has a special type of calculator that allows managers to realize the impact of discounts on their margins immediately. These calculations are based upon the business intelligence sub-system.

2.2.2.7 Restaurant Module

This module basically allows restaurant managers to update menu, change price of an item and update the availability of a particular item. This module can be understood as Restaurant management module to set, get and update different restaurant related parameters.

2.2.3 OLTP Database

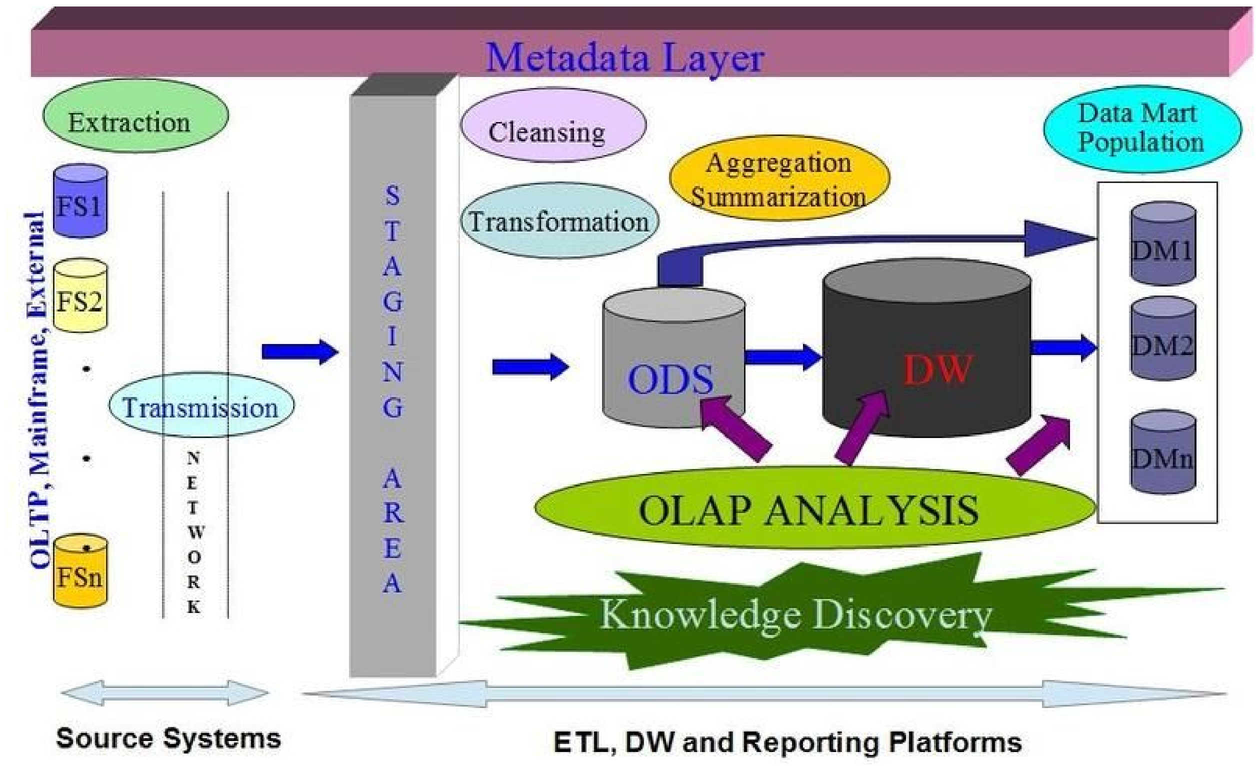
We are using 2 types of databases -

1. LDAP: For user authentication and profile related details which requires less write and more read accesses. This database protocol is really efficient as compared to RDBMS or NoSQL when data has more reads and very less writes.
2. Cassandra: It is a distributed NoSQL database. Cassandra helps to set up a distributed database environment with an ease. It automatically handles with the node failures and data degeneracy. This leads to almost zero downtime and efficient horizontal scaling.

We will be using Redis Cache to optimize the performance of the system.

2.2.4 OLAP Database

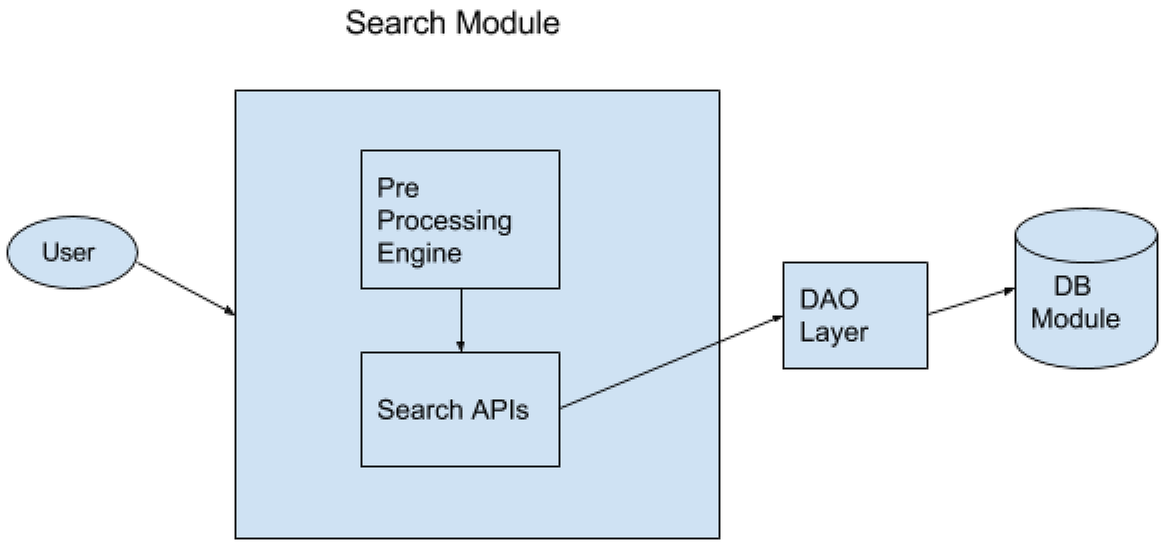
Here is the rough architecture of OLAP database system:



The exact technologies will be finalised at later stages as the project progresses. Refer glossary for better understanding of the above architectural diagram.

3. High Level Design

3.1 Search Module



➢ Search Module has 3 sub-module systems as described below:

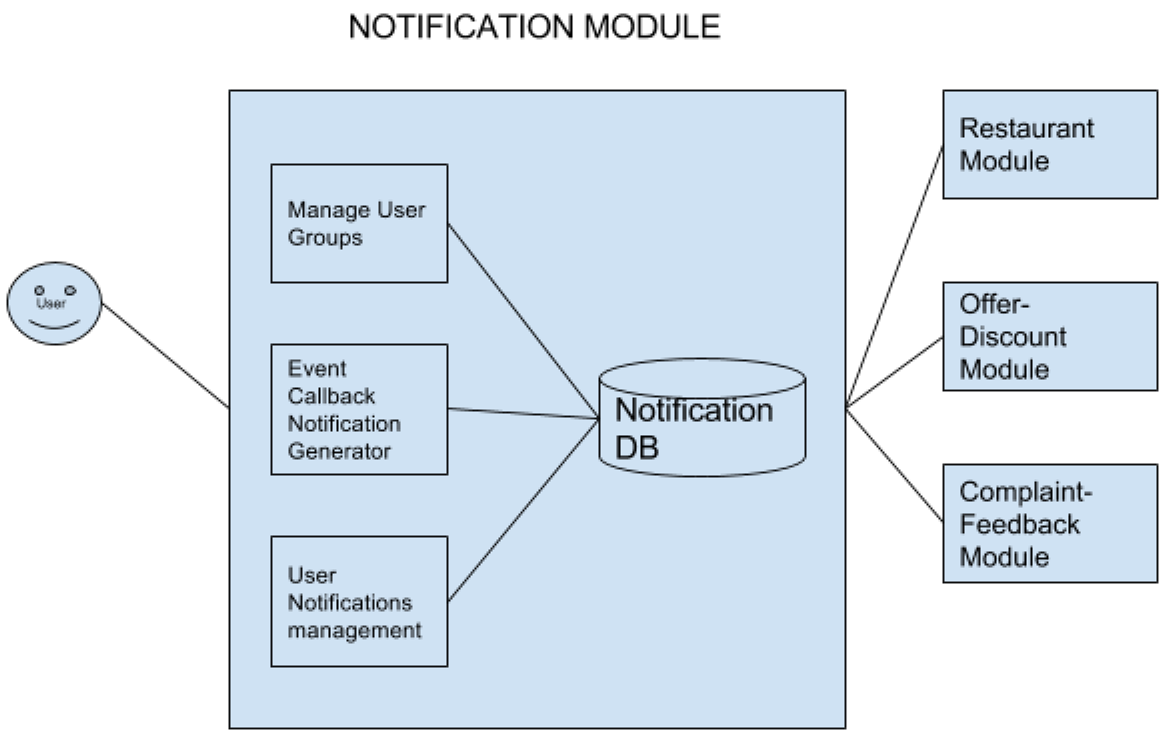
- Pre Processing Engine: This sub-modules takes care about preprocessing all the incoming queries and gives output in the specified way.
- Search APIs: This module has all the APIs or functions used for search. It is therefore responsible for searching items in database and displaying results accordingly.
- DAO Layer: This module provides access to an underlying database.

➢ Search Module mainly interacts with Users (Customers, Managers, Attendants) and Database module.

➢ Exposes following APIs -

- *searchARestuarant(String name)*: This API will help a user to search a restaurant by its name.
- *searchAllRestuarantsInArea(String area)*: This API will help a user to search all the restaurants in a particular area.
- *searchRestuarantInARange(int maximum,int minimum)*: This API will help user to search all the restaurants within given range of amount.
- *searchAnOrder(String OrderID)*: This API will help manager or an attendant to search an order by its unique identifier.

3.2 Notification Module

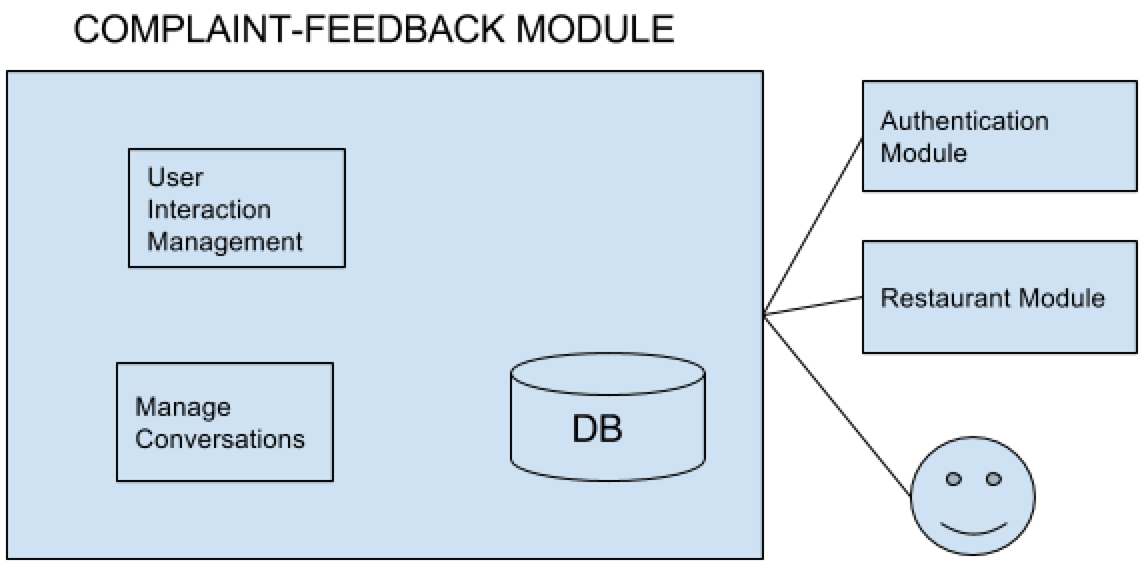


- ➢ Notification module interacts mainly with Users, Offer-Discount module and Complaint-Feedback module.
- ➢ Exposes the following APIs -

- *generate_notification(sender_Id,receiver_Id,message):* Generates notification for particular user or user group given sender id, receiver id and message of notification.

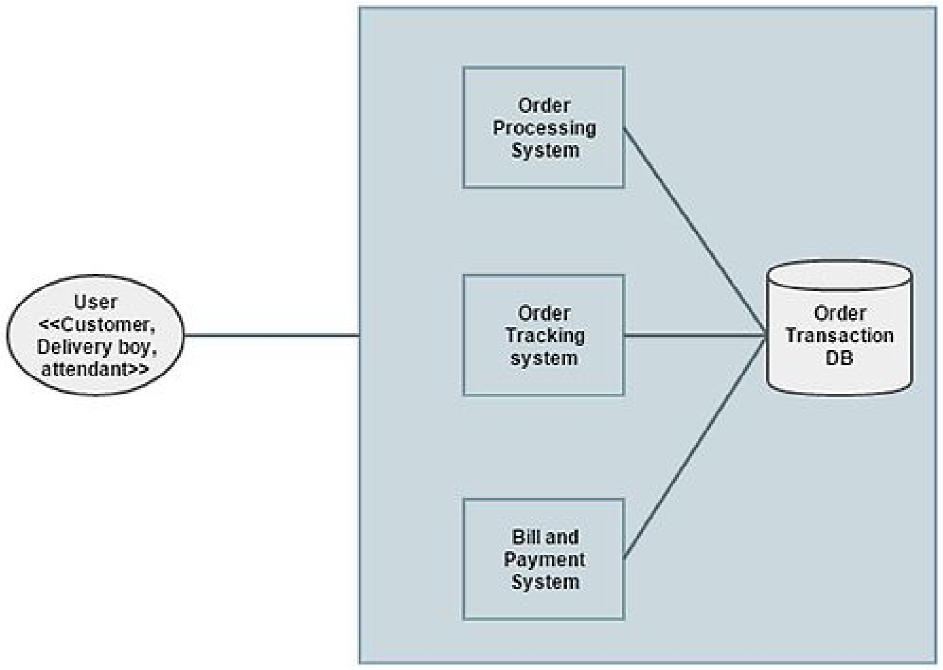
- *event_callback_notification(sender_Id,event_name,message):* Notifications can be generated by functions on occurrence of certain events.
  Events Notification can be preset on Notification Server - on occurrence of event, the event update notification will be sent to destined audience
- *get_notification(receiver_Id)*
  Receive notification from notification server.

3.3 Complaint-Feedback Module



- ➢ Users can register complaints for certain order or service, and its redirected to appropriate user.
- ➢ Following list of APIs are exposed for use -
- *register_complaint(userId, message)*
- *update_status(userId, status)*

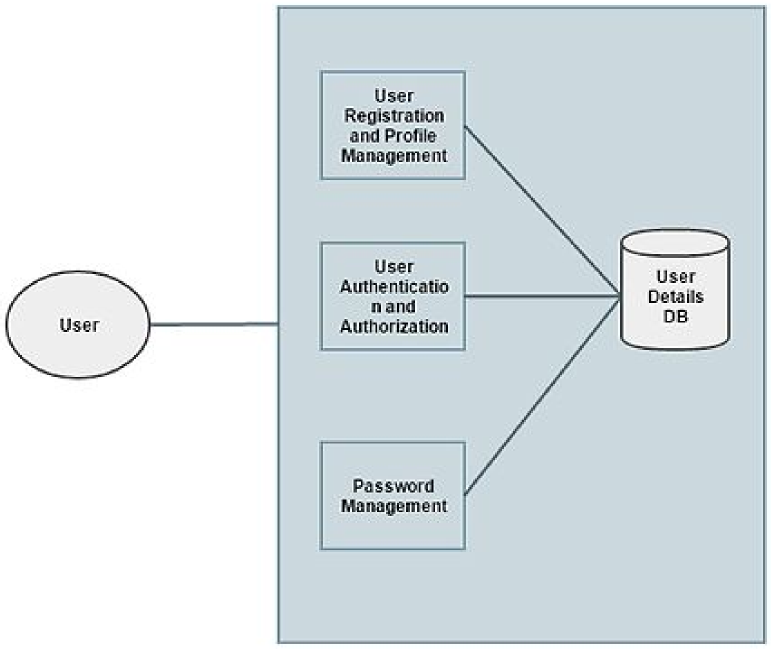
3.4 Order Module



Ordering Module has 3 sub-module systems as below:

- ➢ **Order Processing System:** This modules takes care about preparing order, final confirmation and checking out the order for final payment.
- ➢ **Order Tracking System:** This API handles the tracking system of the order and status update of the for the online ordered food. In case of local cafeteria it handles the token system based on FIFO(First Come First Serve) mechanism or on priority based on some criteria.
- ➢ **Bill and Payment System:** This API handles the 3rd party bill payment authentication and payment system in case of credit/debit or net banking. In case of cash on delivery also it handles the payment status update and tracking by different FOS user group. This API also takes care about the bill generation, bill history and all kind of bill management.

3.5 Authentication Module


User
User Registration and Profile Management
User Authenticatio n and Authorization
Password Management
User Details DB

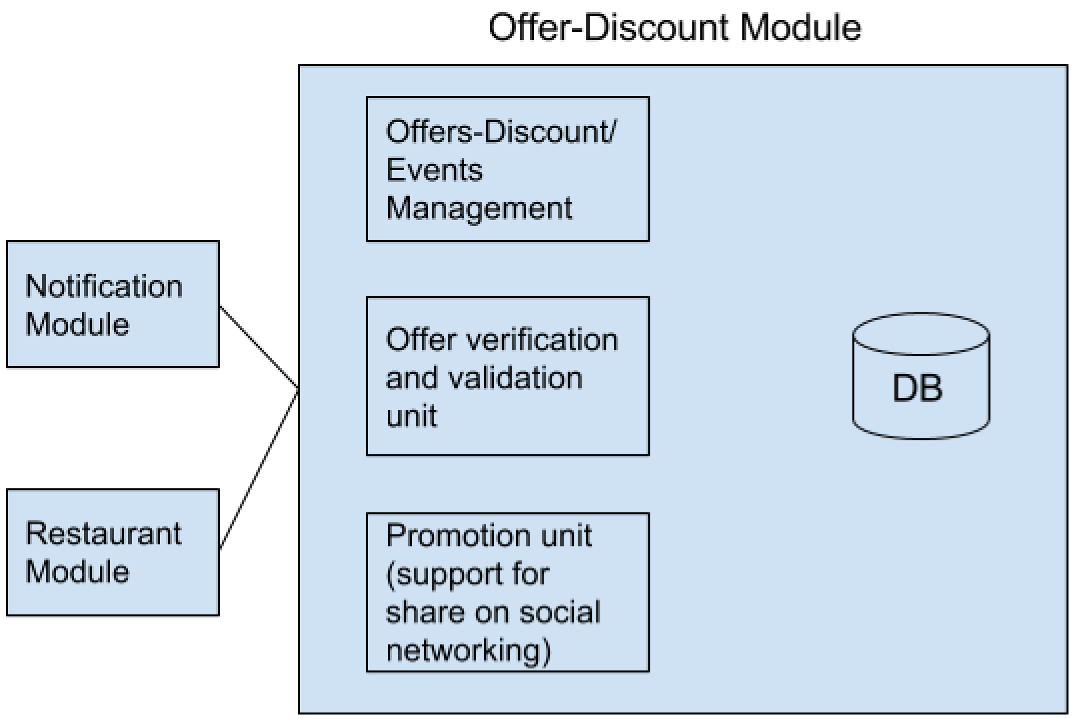
Authentication modules contains basic three sub-modules:

➢ **User Registration and Profile management:** User can register to use this app and further view update delete his information which is said as user profile management.

➢ **User Authentication and Authorization:** User can login into the FOS system by the credentials. User authentication API provides a provision to authenticate the user to access the application and authorization defines the user's accessibility area in the app based on the user's role and authorization details in the DB.

➢ **Password Management:** This API enables a provision to changes the password or reset the password in case user forgot it. This module is also used to activate the new user's account through email.

3.6 Offer-Discount Module


Offer-Discount Module
Offers-Discount/ Events Management
Offer verification and validation unit
Promotion unit (support for share on social networking)
DB
Notification Module
Restaurant Module

Interaction with other modules through APIs :

- *create_offer(TemplateType1 t):* Offer can be created by filling up the pre defined templates available. The creator of the offer has to supply values to fill the template and offer is created.
- *validate_offer(user_id, offer_id)*: To verify if a particular offer/discount is valid for a user.
- *share_on_network(userId, Object ob ,[message]):* User can share details of his order, or offer or promotion post from FOS on his social networks. This API will further use other social networking APIs to post on user's social network.

3.7 Restaurant Module

- *UpdateMenu(restaurant):* This API is used by restaurant manager or to add items, delete item from menu.
- *UpdateItem(restaurant):* This API is used by restaurant manager or to update item details.
- *checkAvailability(Item):* This API is used by restaurant manager or to check availability of an item.
- *AddNewResutaurant(restaurant):* This API is used by restaurant manager to new restaurant in the locality.

4. Low Level Design

➢ MVC Architectural Pattern: This application is going to be developed on Model-View-Controller architectural pattern.

➢ We are going to use ReactJS and Flux architecture for creating reactive page and handling data more precisely client-side.

➢ REST APIs: We mainly use RESTful APIs for communicating with the servers/modules.

➢ Nginx: The server used to host the application modules.

➢ Django web framework as controller of the application. Gunicorn as the interpreter.

➢ Mozilla Desktop Notification APIs: For handling notification publishing on web app.

➢ Redis Cache: We use redis cache for caching all the data from database.

➢ LDAP: For storing user data since more reads and very less writes.

➢ Cassandra: Distributed NoSQL database for storing application data.

➢ Singleton Design Pattern: This pattern is used in data access layer of the application. Basically used in getting the DB connection pool and to maintain the DB connection integrity.

5. Key Data Structures

DATABASE TABLES FOR FOS:

1) RESTAURANT

```
{
Id : string,
Name : string,
Address : string,
Employee Info : {
                        Manager : integer,
                        Receptionist : integer,
                        DeliveryPerson : integer,
                        }
}
```

2) USER

```
{
Id : string,
Name : string,
Permanent Address : string,
Temporary_Address : [ Address1 : {
                                                residence : string,
                                                locality :  string,
                                                city : string,
                                                pincode: string},
                                   Adress2 ....]
Contact : integer,
Gender : [M/F],
Age : integer,
Email : string,
}
```

3) NOTIFICATIONS

- **Notifications**

```
{
Id : string,
sender_userId : string,
receiver_userId : string,
Category : [Complaint/Offer/OrderStatusUpdate]
Message : string
}
```

(broadcast_userId : 0)

- **Notification_groups**

```
{
Group_id : string,
Group_name : string,
Members  : {
              userIds
             }
}
```

4) COMPLAINT-FEEDBACK

```
{
Complaint_id : string,
Status : string,
Complainant_Id : string,
Restaurent_Id : string,
Message : string,

}
```

6. Glossary

1. OFOS - Online Food Ordering System
2. SEO - Search Engine Optimization
3. DAO - Data Access Optimization
4. LDAP - Lightweight Directory Access Protocol
5. OLTP - Online Transaction Processing
6. OLAP - Online Analytical Processing
7. MVC - Model-View-Controller
8. API - Application Program Interface
9. ETL - Extract Transform Load
10. DW - Data Warehouse
11. ODS - Operational Data Store