

**<<CHATTER>>**

**Software Design Document**

– Hanoi, May 11 –

**Table of Contents**

[I. Overview 3](#_heading=h.gjdgxs)

[1. Code Packages 3](#_heading=h.30j0zll)

[2. Database Schema 3](#_heading=h.1fob9te)

[II. Code Designs 4](#_heading=h.3znysh7)

[1. Account Management Feature](#_heading=h.2et92p0) 5

[a. Class Diagram](#_heading=h.tyjcwt) 5

[b. Class Specifications](#_heading=h.3dy6vkm) 6

[c. Sequence Diagram(s)](#_heading=h.1t3h5sf) 6

[d. Database queries](#_heading=h.4d34og8) 6

[2. Customer Feature](#_heading=h.2s8eyo1) 7

[a. Class Diagram](#_heading=h.tyjcwt) 7

[b. Class Specifications](#_heading=h.3dy6vkm) 8

[c. Sequence Diagram(s)](#_heading=h.1t3h5sf) 9

[d. Database queries](#_heading=h.4d34og8) 9

[3. Message Management Feature](#_heading=h.ho04fngdm2r7) 9

[a. Class Diagram](#_heading=h.tyjcwt) 9

[b. Class Specifications](#_heading=h.3dy6vkm) 10

[c. Sequence Diagram(s)](#_heading=h.1t3h5sf) 10

[d. Database queries](#_heading=h.4d34og8) 11

[4. General Feature](#_heading=h.t5hagnufu1nq) 11

[a. Class Diagram](#_heading=h.tyjcwt) 11

[b. Class Specifications](#_heading=h.3dy6vkm) 11

[c. Sequence Diagram(s)](#_heading=h.1t3h5sf) 13

[d. Database queries](#_heading=h.4d34og8) 13

[5. Marketing Feature](#_heading=h.4iwevzkz87) 14

a. Class Diagram 14

[b. Class Specifications 14](#_heading=h.3dy6vkm)

[c. Sequence Diagram(s)](#_heading=h.1t3h5sf) 15

[d. Database queries](#_heading=h.4d34og8) 15

[6. Restaurant Management Feature](#_heading=h.ruqu04lu1l5h) 17

[a. Class Diagram](#_heading=h.tyjcwt) 17

[b. Class Specifications](#_heading=h.3dy6vkm) 17

[c. Sequence Diagram(s)](#_heading=h.1t3h5sf) 18

[d. Database queries](#_heading=h.4d34og8) 18

7[. Admin Feature](#_heading=h.ruqu04lu1l5h) 18

[a. Class Diagram](#_heading=h.tyjcwt) 18

[b. Class Specifications](#_heading=h.3dy6vkm) 19

[c. Sequence Diagram(s)](#_heading=h.1t3h5sf) 19

[d. Database queries](#_heading=h.4d34og8) 19

[III. Database](#_heading=h.17dp8vu) Design 20

[1. <Table name 1> 5](#_heading=h.3rdcrjn)

[2. <Table name 2…> 5](#_heading=h.26in1rg)

# I. Overview

## 1. Code Packages/Namespaces

*[Provide the package diagram for each sub-system. The content of this section including the overall package diagram, the explanation, package and class naming conventions in each package. Please see the sample and description table format below – following Java project naming convention]*



***Package descriptions & package class naming conventions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| *01* | *controller* | *Control flow into model and update view*  *Package name: Controller* |
| *02* | *model* | *A database, file, or a simple object*  *Package name: Model* |
| *03* | *view* | *Display data to screen* |

## 2. Coding Conventions

# II. Code Designs

## 1. Account Management Feature

### a. Class Diagram



### b. Class Specifications

#### LoginController Class

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *redirect to login page* |
| *02* | *doPost* | *search the database for username and password and return true if found* |

#### LogoutController Class

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest* | *end session* |

#### 

#### RegisterController Class

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *redirect to register page* |
| *02* | *doPost* | *create new account and insert into database* |

#### ResetPassword Class

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *redirect to reset password page* |
| *02* | *doPost* | *take action type to choose a method to execute* |
| *03* | *sendAGPassword* | *send an auto-generate password to user’s email, redirect to the confirm page.* |
| *04* | *confirmAGPassword* | *If user input correct password, send redirect to the update password page.* |
| *05* | *updatePassword* | *update user’s input password into database* |

#### ChangePassword Class

***Class Methods***

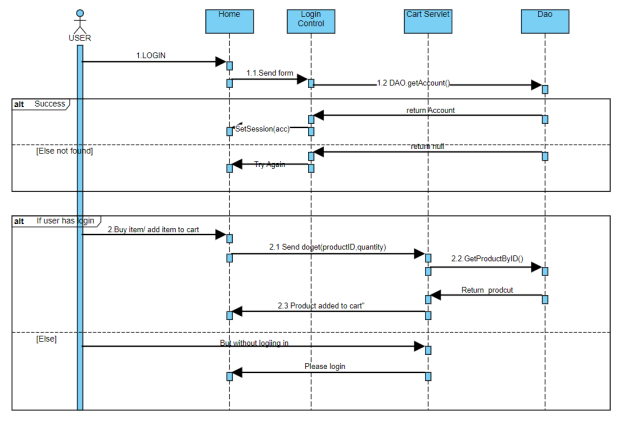
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processGet* | *redirect to change password page* |
| *02* | *processPost* | *if user input correct old password, update new password* |

#### EditUserProfile Class

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processGet* | *redirect to profile page* |
| *02* | *processPost* | *update user profile to the database* |

### c. Sequence Diagram(s)



### d. Database queries

*SELECT [username]*

*,[password]*

*,[displayname]*

*FROM [users]*

*WHERE [username] = ? AND [password] = ?*

*INSERT INTO [users]*

*([username]*

*,[password]*

*,[displayname])*

*VALUES*

*(?*

*,?*

*,?)*

*UPDATE [users]*

*SET [password] = ?*

*WHERE [username] = ?*

## 2. Customer Feature

### a. Class Diagram



### b. Class Specifications

#### MyOrders Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | load orders list from database and redirect to the order page |
| 02 | processPost | edit orders still in summited form |

#### BuyController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processPost | add customer select foods to customer’s cart in database |

#### MyCart Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | load cart and redirect to cart page |
| 02 | processPost | update cart is edited by customer to the database |

#### CommentController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | Customers can see a comment area at the end of food detail page or restaurant detail page |
| 02 | processPost | Add comment with vote to the database (if customer vote many times, only the last vote will count)1 |

### c. Sequence Diagram(s)

[Provide the sequence diagram(s) for the feature, see the sample below]



### d. Database queries

## 3. Message Management Feature

### a. Class Diagram



### b. Class Specifications

#### MesssageController Class

#### Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | load customer messages list from database |
| 02 | processPost | insert customer messages into database |

### 

#### RestaurantMesssageController Class

#### Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | load restaurant messages list from database |
| 02 | processPost | insert restaurant messages into database |

#### AdminMessageController Class

#### Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | load customer list with message from database and redirect to message page |
| 02 | processPost | insert customer messages into database |

### c. Sequence Diagram(s)

[Provide the sequence diagram(s) for the feature, see the sample below]



### d. Database queries

[Provide the detailed SQL (select, insert, update...) which are used in implementing the function/screen]

## 

## 4. General Feature

Features for guest who are not login.

### a. Class Diagram



### b. Class Specifications

#### FoodController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | doGet | load food list from database, send food list and redirect to the home page |
| 02 | doPost | when customer click on food image, redirect to the food’s detail page |

#### FoodDetailController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | doGet | redirect to a chosen food detail page |
| 02 | doPost | add numbers of order food to the cart |

#### CategoryController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | doGet | load category list from database, send category list and redirect to the home page |
|  | doPost | when customer click on category image, view foods from that category |

#### RestaurantController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | doGet | load Restaurants list from database, send Restaurant list and redirect to the home page |
|  | doPost | when customer click on Restaurants image, redirect to the restaurant detail page |

#### RestaurantDetailController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest | redirect to a chosen restaurant detail page with foods from the restaurant |

#### FeedbackController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest | redirect to the feedback page and insert feedback to the database |

### c. Sequence Diagram(s)

[Provide the sequence diagram(s) for the feature, see the sample below]



### d. Database queries

select \* from Product

select \*

from Product

where Product\_ID = ?

select \*

from Category

SELECT \*

FROM Feedback

## 

## 5. Marketing Feature

### a. Class Diagram



### b. Class Specifications

#### ListOrdersController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | doGet | load orderlist from database, send food list and redirect to the home page |
| 02 | doPost | update edited order into database |

#### ListCategoryController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | doGet | load category list from database, send category list and redirect to the home page |
|  | doPost | update edited categoryinto database |

#### ListCustomerController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | doGet | load customer list from database, send customer list and redirect to the home page |
|  | doPost | update edited customer into database |

#### ListFeedbackController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | doGet | load feedback list from database, send feedback list and redirect to the home page |
|  | doPost | update edited feedback into database |

### c. Sequence Diagram(s)

[Provide the sequence diagram(s) for the feature, see the sample below]



### d. Database queries

*select \* from Product*

*UPDATE [Product]  
 SET [Product\_Name] = ?  
 SET [Product\_Price] = ?  
 SET [Product\_image] = ?  
 WHERE [Product\_ID] = ?*

*select \**

*from Product*

*where Product\_ID = ?*

*SELECT [username]*

*,[password]*

*,[displayname]*

*FROM [users]*

*WHERE [role] = ‘Customer’*

*UPDATE [User]  
 SET [displayname] = ?  
 SET [email] = ?  
 SET [firstname] = ?  
 WHERE [username] = ?*

*select \**

*from Category*

*UPDATE [Category]  
 SET [Category\_Name] = ?  
 WHERE [Category\_ID] = ?*

*SELECT \**

*FROM Feedback*

## 6. Restaurant Management Feature

### a. Class Diagram



### b. Class Specifications

#### RestaurantStatsController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | view orders number, turnover, top customers,.. |

#### ListFoodController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | load restaurant’s foods list from database |
| 02 | processPost | updating edited foods into database |

#### EditRestaurantDetailController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | redirect to the restaurant edit detail page |
|  | processPost | update restaurant information into database |

### c. Sequence Diagram(s)



### d. Database queries

## 7. Admin Feature

### a. Class Diagram



### b. Class Specifications

#### ListUserController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | load users list from database, send users list and redirect to the admin home page |
| 02 | processPost | update edited users into database |

#### UserController Class

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | processGet | load chosen user from database, send and redirect to the edit page |
|  | processPost | update user information into database |

### c. Sequence Diagram(s)



### d. Database queries

*SELECT \**

*FROM [users]*

*INSERT INTO [users]*

*([username]*

*,[password]*

*,[displayname])*

*VALUES*

*(?*

*,?*

*,?)*

*UPDATE [users]*

*SET [role] = ?*

*SET [displayname] = ?*

*SET [email] = ?*

*SET [role] = ?*

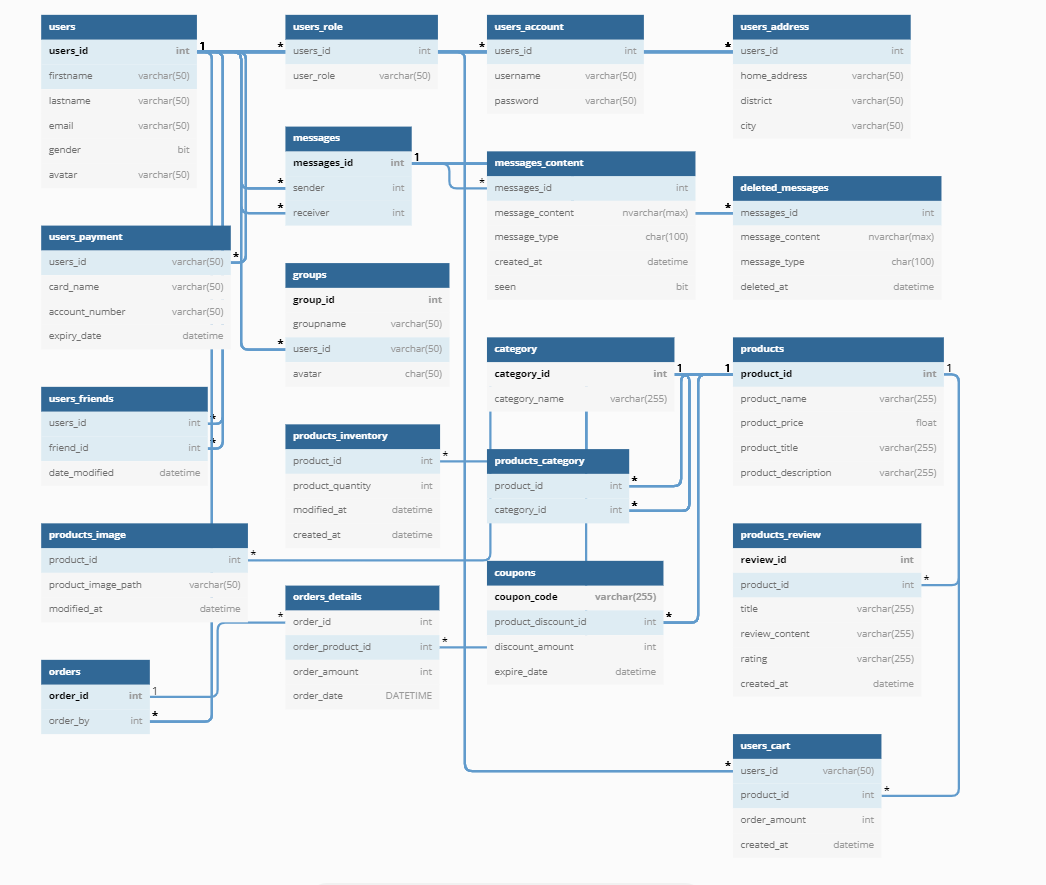
*WHERE [username] = ?*

## 

# III. Database Design

## 1. Database Schema

*[Provide the tables relationship like example below – following MySQL database naming convention]*

**

## 2. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| *01* | *users* | *<Description of the table>*  *- The users table shows the user's information including username, password, last name, first name, email, gender, role*  *- Primary keys: users\_id*  *- Foreign keys: <<list of foreign key fields>>* |
| *02* | *messages* | - message between customer and seller  - Primary keys: id  - Foreign key: username |
| *03* | *users\_friends* | - a table to save friends via messages with the date and time of the message  - Primary keys: user\_id, friend\_id |
| *04* | *group* | - group includes friends list  - Primary keys: id |
| *05* | *coupon* | - coupon, including discount code and expiration date  - Primary keys: CouponCode |
| *06* | *orders* | - table containing detailed orders for each order including order number, date, and amount  - Primary keys: order\_id |
| *07* | *products* | - The table contains detailed product information including name, photo, price, title, product type, product type code,  - Primary keys: Product\_ID |
| *08* | *category* | - The table contains the list of products  - Primary keys: Category\_ID |
| *09* | *users\_role* | -List of users and their respective roles  -FK: users\_id |
| *10* | *users\_account* | -List of users and their login account  -FK: users\_id |
| *11* | *users\_address* | -List of users address  -FK: users\_id |
| *12* | *users\_payment* | -List of users card info:cardname, account number, expiry date  -FK: users\_id |
| *13* | *products\_image* | -List of products image path  -FK: products\_id |
| *14* | *deleted\_messages* | -all deleted messages  -FK: users\_id |
| *15* | *products\_inventtory* | -show all detailed information of products including quangtity  -FK: product-id |
| *16* | *products\_category* | -list of all product’s category  -FK: product-id |
| *17* | *products\_review* | -list of all product’s review  -FK: product-id |
| *18* | *users\_cart* | -list of all item in user’s cart  -FK: users\_id |
| *19* | *orders\_details* | - The table contains detailed order information including id, amount  - FK: order-id |
| *20* | *messages\_details* | - show all detailed info of messages  -FK: messages\_id |

CREATE TABLE users(

users\_id int identity(1,1) primary key,

firstname varchar(50) NOT NULL,

lastname varchar(50) NOT NULL,

email varchar(50) NOT NULL,

gender bit NOT NULL,

avatar varchar(50)

);

create table users\_role(

users\_id int references users(users\_id) NOT NULL,

user\_role varchar(50) NOT NULL

);

create table users\_account(

users\_id int references users(users\_id) NOT NULL,

username varchar(50) NOT NULL,

password varchar(50) NOT NULL

);

Create TABLE users\_address(

users\_id int references users(users\_id),

home\_address varchar(50) NOT NULL,

district varchar(50) NOT NULL,

city varchar(50) NOT NULL

);

create table users\_payment(

users\_id varchar(50) references users(users\_id) NOT NULL,

card\_name varchar(50) NOT NULL,

account\_number varchar(50) NOT NULL,

expiry\_date datetime

);

CREATE TABLE messages (

messages\_id int identity(1,1) primary key,

sender int references users(users\_id) NOT NULL,

receiver int references users(users\_id) NOT NULL

);

CREATE TABLE messages\_content(

messages\_id int references messages(messages\_id) NOT NULL,

message\_content nvarchar(max) NOT NULL,

message\_type char(100) NOT NULL,

created\_at datetime default current\_timestamp,

seen bit NOT NULL

);

CREATE TABLE deleted\_messages(

messages\_id int references messages(messages\_id) NOT NULL,

message\_content nvarchar(max) NOT NULL,

message\_type char(100) NOT NULL,

deleted\_at datetime default current\_timestamp

);

CREATE TABLE users\_friends (

users\_id int references users(users\_id),

friend\_id int references users(users\_id),

date\_modified datetime default current\_timestamp

) ;

create table groups(

group\_id int identity(1,1) primary key,

groupname varchar(50) NOT NULL,

users\_id varchar(50) references users(users\_id) NOT NULL,

avatar char(50)

);

create table category(

category\_id int PRIMARY KEY IDENTITY(1,1),

category\_name varchar(255) NOT NULL

);

create table products(

product\_id int NOT NULL PRIMARY KEY IDENTITY(1,1),

product\_name varchar(255) NOT NULL,

product\_price float NOT NULL,

product\_title varchar(255) NOT NULL,

product\_description varchar(255) NOT NULL

);

create table products\_image(

product\_id int references products(product\_id),

product\_image\_path varchar(50) NOT NULL,

modified\_at datetime default current\_timestamp

);

create table products\_inventory(

product\_id int references products(product\_id),

product\_quantity int NOT NULL,

modified\_at datetime,

created\_at datetime default current\_timestamp

);

create table products\_category(

product\_id int NOT NULL references products(product\_id),

category\_id int references category(category\_id)

);

create table products\_review (

review\_id int NOT NULL PRIMARY KEY IDENTITY(1,1),

product\_id int NOT NULL references products(product\_id),

title varchar(255) NOT NULL,

review\_content varchar(255) NOT NULL,

rating varchar(255) NOT NULL,

created\_at datetime default current\_timestamp

);

Create table orders(

order\_id int PRIMARY KEY IDENTITY(1,1),

order\_by int REFERENCES users(users\_id)

);

Create table orders\_details(

order\_id int references orders(order\_id),

order\_product\_id int REFERENCES products(product\_id),

order\_amount int NOT NULL,

order\_date DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP

);

Create table coupons(

coupon\_code varchar(255) NOT NULL PRIMARY KEY,

product\_discount\_id int REFERENCES products(product\_id),

discount\_amount int NOT NULL,

expire\_date datetime NOT NULL

);

create table users\_cart(

users\_id varchar(50) references users(users\_id) NOT NULL,

product\_id int NOT NULL references products(product\_id),

order\_amount int NOT NULL,

created\_at datetime default current\_timestamp

);