

**BAKERY SHOP**

**Software Design Document**

– Hanoi, August 2019 –

**Table of Contents**

[I. Overview 3](#_gjdgxs)

[1. Code Packages 3](#_30j0zll)

[2. Database Schema 3](#_1fob9te)

[II. Code Designs 4](#_3znysh7)

[1. <Feature/Function Name1> 4](#_2et92p0)

[a. Class Diagram 4](#_tyjcwt)

[b. Class Specifications 4](#_3dy6vkm)

[c. Sequence Diagram(s) 4](#_1t3h5sf)

[d. Database queries 5](#_4d34og8)

[2. <Feature/Function Name2> 5](#_2s8eyo1)

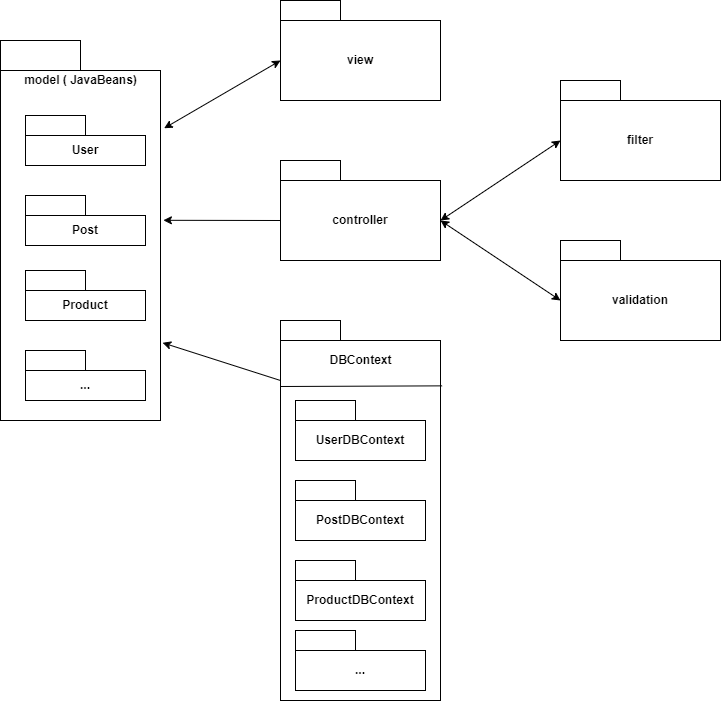
[III. Database Tables 5](#_17dp8vu)

[1. <Table name 1> 5](#_3rdcrjn)

[2. <Table name 2…> 5](#_26in1rg)

# I. Overview

## 1. Code Packages/Namespaces



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

| **No** | **Package** | **Description** |
| --- | --- | --- |
| *01* | *Model* | * *The Model is the part of MVC which implements the domain logic. In simple terms, this logic is used to handle the data passed between the database and the user interface (UI).* * *The domain objects are stored under the Models package* * *Class names should be nouns, in mixed case with the first letter of each internal word capitalized* * *Model name (Singular name) corresponding to the database table should be similar to the database table name* * *Try to keep your class names simple and descriptive* |
| *02* | *DA0* | * *DAO pattern offers a logic to structure your persistence mechanism (the glue between your database and the model of your MVC).* * *DAO is an abbreviation for Data Access Object, so it should encapsulate the logic for retrieving, saving and updating data in your data storage (a database, a file-system, whatever).* * *Class names should be nouns appending “DAO”, in mixed case with first letter of each noun word capitalized and “DAO” all capitalized* * *Try to keep your class names simple and descriptive* |
| *03* | *Controller* | * *The Controller package contains classes which are responsible for controlling the application logic and acts as the coordinator between the View and the Model. The Controller receives an input from the users via the View, then processes the user's data with the help of Model and passes the results back to the View.* * *It controls the data flow into model object and updates the view whenever data changes* * *Class names must be nouns ending with “Controller”, in mixed case with first letter of each noun and “controller” word capitalized* * *Try to keep your class names simple and descriptive* |
| *04* | *View* | * *View package has classes which represent the visualization of the data that model contains* * *Class names should be nouns, in mixed case with the first letter of each internal word capitalized* * *Try to keep your class names simple and descriptive* |
| *05* | *Validation* | * *Validation Package contains classes used to check input data and ensure that information is valid* * *Class names should be nouns, in mixed case with the first letter of each internal word capitalized* * *Try to keep your class names simple and descriptive* |
| *06* | *Filter* | * *This package contains filter classes that are responsible for intercepting requests from the client before they are passed to the backend and for manipulating responses from the servlet before they are sent back to the client.* * *Class names should be nouns, in mixed case with the first letter of each internal word capitalized* * *Try to keep your class names simple and descriptive* |

## 2. Coding Conventions

* Follow proper naming convention;
* Add comments;
* Identifier means a symbolic name that refers to the name of classes, packages, methods and variables in a Java program;
* The variable name should be related to its purpose;
* Name of the method should relate to the method's functionality;
* There should be no Duplicate code in the same class or other class;
* Declare global variables only if necessary to use in the other methods;
* Double check creation of static variables inside a class;
* Avoid accessing variables directly from other classes instead use getter and setter methods;
* All business logic should be handled in the service class only;
* All DB related code should be in the DAO classes only;
* Use getters and setters;
* Declare instance Variable as private;

# II. Code Designs

## 1. <Feature/Function Name1>

*[Provide the detailed design for the function <Feature/Function Name1>. It include Class Diagram, Class Specifications, and Sequence Diagram(s)]*

### a. Class Diagram

*[This part presents the class diagram for the relevant feature]*



### b. Class Specifications

*[Provide the description for each class and the methods in each class, following the table format as below]*

### c. Sequence Diagram(s)

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



### d. Database queries

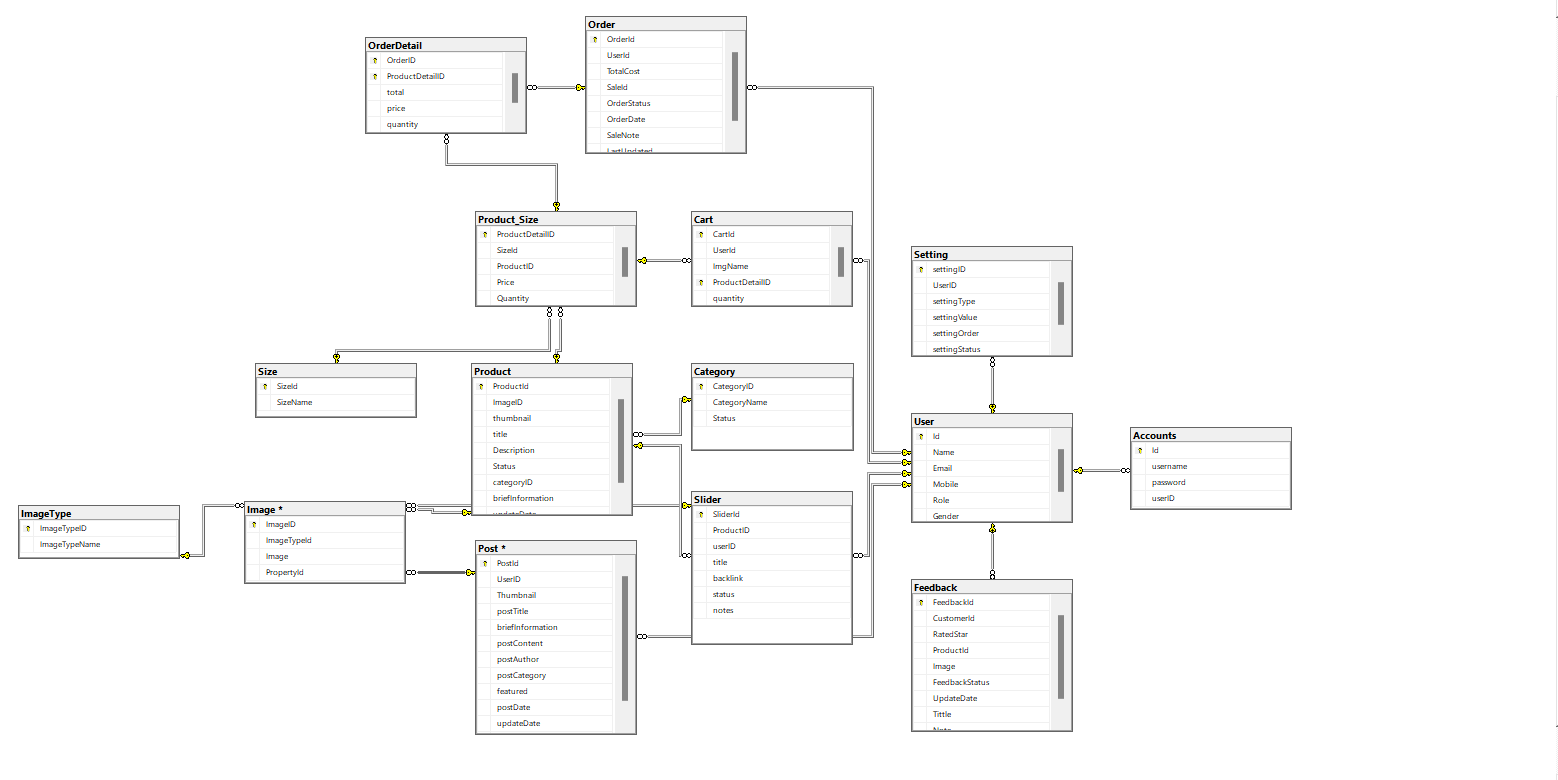
*[Provide the detailed SQL (select, insert, update...) which are used in implementing the function/screen]*

## 2. <Feature/Function Name2>

…

# III. Database Design

## 1. Database Schema



## 2. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| *01* | *User* | *Stores a list of user information (name, mobile,email,gender,...)*  *- Primary keys: UserID*  *- Foreign keys: none* |
| *02* | *Product* | *Store information list of products sold on the web with product attributes*  *- Primary keys: ProductID*  *- Foreign keys: categoryID* |
| *03* | *Category* | *Store information list of product categories appearing in the website*  *- Primary keys: CategoryID*  *- Foreign keys: none* |
| *04* | *Account* | *Store the list of accounts, login information (username, password) to the website*  *- Primary keys: AccountID*  *- Foreign keys: userID* |
| *05* | *Order* | *Store information including ordering person, product-related information such as order date, total price, ...*  *- Primary keys: OrderID*  *- Foreign keys: UserID* |
| *06* | *OrderDetails* | *Deeper storage of order information such as quantity, category, size, product id,...*  *- Primary keys: ProductID*  *- Foreign keys: Orderid, ProductID* |
| *07* | *Product\_Size* | *Store information about price, quantity corresponding to each product id and product size*  *- Primary keys: ProductID*  *- Foreign keys: SizeId, ProductID* |
| *08* | *Size* | *Store the size of each product in the web*  *- Primary keys: SizeId*  *- Foreign keys: none* |
| *09* | *Post* | *Store information about the posts made by the marketing team including specifically: function, posting date, post content, category,...*  *- Primary keys: PostID*  *- Foreign keys: UserID* |
| *10* | *Slider* | *Contains images, titles, and links of products that are alternately run in the website*  *- Primary keys:SliderID*  *- Foreign keys: UserID* |
| *11* | *Setting* | *Contains information before and after being updated by the user, applicable to the user or the website.*  *- Primary keys: SettingID*  *- Foreign keys: UserID* |
| *12* | *Feedback* | *Includes feedback on product quality, service quality, comments, reviews from users sent to the website for each product.*  *- Primary keys: FeedbackID*  *- Foreign keys: UserID* |