Rentek

ONLINE SHOPPING PORTAL

Software Requirements Specification

Version 1.1

Team Guide:	V. B. Nikam
Members:	Rohit Joshi Amol Patil Ritesh Bhatkar Navneet Kambo
College Name:	Veermata Jijabai Technological Institute
Department:	Information Technology
State:	Maharashtra

Revision History

Date	Version	Description	Author
12/12/2010	1.0	Initial SRS	Rentek
12/2/2010	1.1	Final SRS	Rentek

Table of Contents

1 Introduction 1
1.1 Purpose 1
1.2 Scope 2
1.3 Definitions, acronyms and abbreviations 3
1.4 References 4
1.5 Technologies to be used 5
1.6 Overview 6
2 Overall description 7
2.1 Product Perspective 7
2.2 Software Interface 8
2.3 Hardware Interface 9
2.4 Product Function 10
2.5 User Characteristics 11
2.6 Constraints 12
2.7 Architecture Design 13
2.8 Use Case Model Description 14
2.9 Class Diagram 16
2.10 Sequence Diagrams 27
2.11 Database Design 39
2.11.1 ER Diagram 39
2.11.2 Schema 40
2.12 Assumptions and Dependencies 41
3 Specific Requirements 42
3.1 Use case reports 42
3.2 Supplementary Requirements 51

Software Requirements Specification

1 Introduction

1.1 Purpose

To provide a one-stop shopping assistant to all users (novice/expert) alike. The shopping portal would reduce users' burden of sifting through a large number of various products in the market. Instead, the user will specify the product specifications and the system will give a list of products matching the criteria and redirect users to the appropriate web-sites that offer to sell the product.

1.2 Scope

- ✓ **Unlimited extensibility:** The system provides support for cataloging any type of product. Once the product's various aspects have been defined (using the predefined data-types provided by the system), the users can search on any detail of the product. There is no limit to the number of aspects that can be added to a product, nor is there a limit on the number of product types that can be defined or on the total number of products catalogued itself.

- ✓ **Highly scalable:** The system has been designed so that even with a large number of products catalogued the performance does not suffer. Even if million products are catalogued in 100 categories, the system (including the search function) still is as efficient as having 10,000 products in a single category.

- ✓ **Highly customizable:** While defining the product types, it is possible to set ranges for numeric fields, and it is possible to set options for string fields. There is no limit to the ranges that the user can set, and neither is there any limit for the number of options for a string field. The system stores all this information as an XML document. Further, there are many settings stored in different files which allow the system to be modified without recompiling. Even permissions can be edited this way.

- ✓ **Multiple database support:** The system has been designed so that with little modifications, any database engine can be used. Currently, only IBM DB2 is supported.

- ✓ **Captcha support:** The system uses captchas for multiple tasks which can be performed by users who have not logged in to provide bot protection.

- ✓ **Mobile web support:** The system not only supports a standard UI, but also a light-weight UI for mobile devices. The system adopts a pattern (MVC) that allows multiple UIs with minimal work. Thus, additional UIs can be added without affecting the system core itself.

- ✓ **Multi-language support:** The system has support for multiple languages. The user can select any language of preference. Additionally, more languages can be added without modifying\recompiling the code.

- ✓ **Changeable themes support:** The system allows users to change visual theme of the UI. Additionally, more themes can be added without modifying\recompiling the code.

- ✓ **Easy installation:** The system has support for an installer that provides an easy UI for configuring various options (like the database engine used, the view being used, etc.)

1.3 Definitions, acronyms and abbreviations

- HTML (Hyper Text Markup Language): It is used to create static web pages.
- JSP (Java Server Pages): It is used to create dynamic web content.
- J2EE (Java 2 Enterprise Edition): It is a programming platform, belonging to the Java platform, which is used for developing and running distributed java applications.
- WASCE (WebSphere Application Server Community Edition): It is an application server that runs and supports the J2EE and the web service applications.
- WSAD (WebSphere Studio Application Developer): It is a designer toolkit which is designed to develop more complex projects by providing a complete dynamic web service.
- DB2 (IBM Database 2): It is a database management system that provides a flexible and efficient database platform to raise a strong "on demand" business applications.
- HTTP (Hyper Text Transfer Protocol): It is a transaction oriented client/ server protocol between a web browser and a web server.
- XML (Extensible Markup Language): It is a markup language that was designed to transport and store data.
- AJAX (Asynchronous JavaScript and XML): It is a technique used in java script to create dynamic web pages.
- Web 2.0: It is commonly associated with web applications which facilitate interactive information sharing, interoperability, user-centered design and collaboration on the World Wide Web.

1.4 References

- Project specification requirement (provided by IBM)
- IEEE SRS format

1.5 Technologies to be used

Technologies used:

- J2EE (Servlet, JSP, Java Beans)
- JAVA
- AJAX (Asynchronous Java Script and XML)
- XML (Extensible Markup Language)

IBM software used:

- Rational Rose v7.0.0
- InfoSphere Data Architect v7.5.3.0
- Eclipse v3.3
- DB2 - Express 'C' v9.1 (also tested on v9.7.2)
- WebSphere Application Server Community Edition (WASCE) v2.0.0.1 (also tested on v2.1.1.5)

1.6 Overview

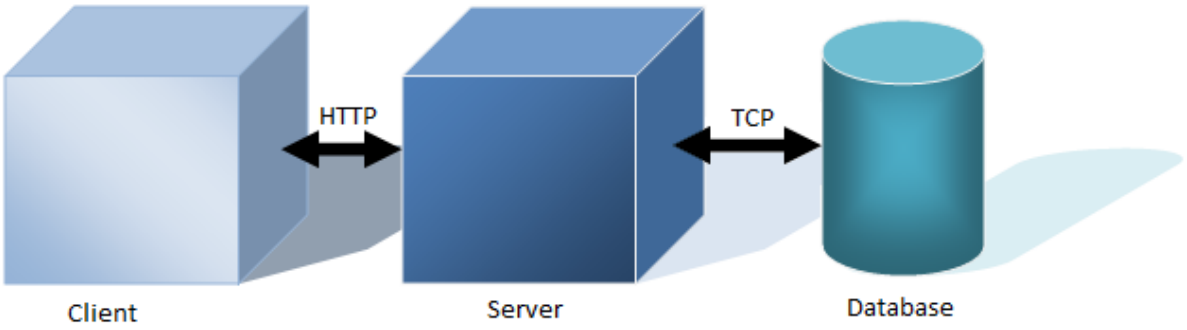
The SRS will include two sections, namely:

- **Overall Description:** This section will describe major components of the system, interconnections, and external interfaces.
- **Specific Requirements:** This section will describe the functions of actors, their roles in the system and the constraints faced by the system.

2 Overall description

2.1 Product Perspective

- The web pages provide the UI on the client side. Communication between client and server is through HTTP.
- The client side software consists of the web browser and the operating system itself.
- The server side consists of the web server (for the web application) and the database server (for storing data for the web application).



The database may reside on the same server on which the web server resides, or on a different server.

2.2 Software Interface

- **Client:** Web Browser, Operating System (any).
- **Web Server:** WASCE, Operating System (any).
- **Database Server:** DB2, Operating System (any).

2.3 Hardware Interface

❖ **Client Side:**

	Processor	RAM	Disk space
Internet Explorer 6 (or equivalent)	Intel Pentium III (1 GHz) or equivalent	256 MB	1 GB

❖ **Server Side:**

	Processor	RAM	Disk space
WebSphere Application Server Community Edition v2.0.0.1	Intel Pentium III (1 GHz) or equivalent	512 MB	256 MB
DB2 - Express 'C' v9.1	Intel Pentium III (1 GHz) or equivalent	512 MB	1 GB (excluding data size)

2.4 Product Function

The sole function of the system is to help users buy products.

The system allows users to comment on products. Thus the users can provide reviews, benefits and disadvantages of products. For the novice users, the system provides a "basic search" interface that allows users to specify what they want. Novice users may benefit from other users' experience by viewing their comments. For expert users, the system provides an "advanced search" interface that allows users to precisely specify what they want.

A voting system allows users to gauge the popularity of a particular product.

2.5 User Characteristics

The users must be comfortable in using computers, particularly in browsing the web.

2.6 Constraints

- The data-types used for defining product fields are hard-coded and if extra data-type is needed, or if it is desired to alter functionality of a data-type, modifying and recompiling the code is required.
- Not everything is localized – examples are product names, product type names, and other things stored in the database.

2.7 Architecture Design

The system is based on the MVC (model view controller) design pattern.

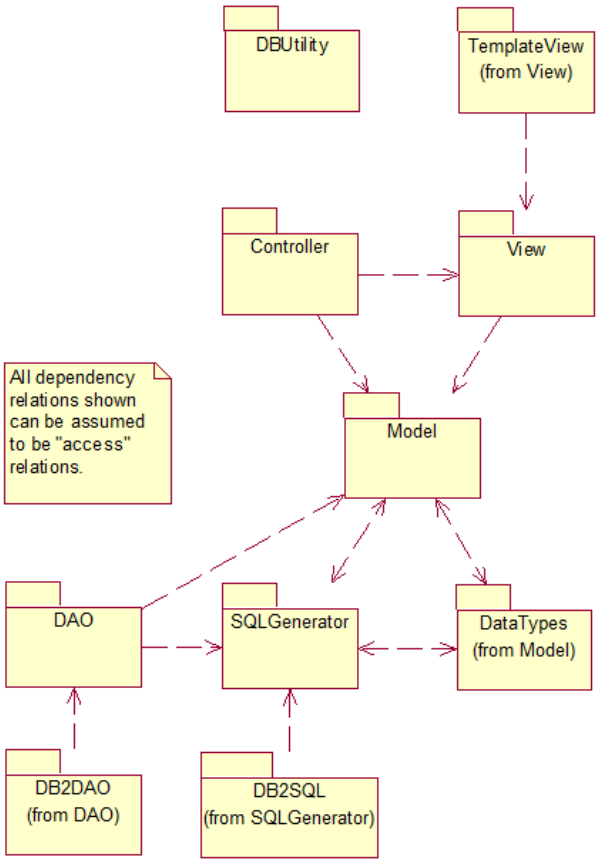
The model consists of the classes that represent objects or users, like product, product definition, etc.

The view consists of what the user sees. The system has two views, a standard view (for PCs) and a lightweight view (for mobile devices). The view depends on the model.

The controller consists of actions. These actions may be initiated by the user, or by the system. The controller depends on the model as well as the view.

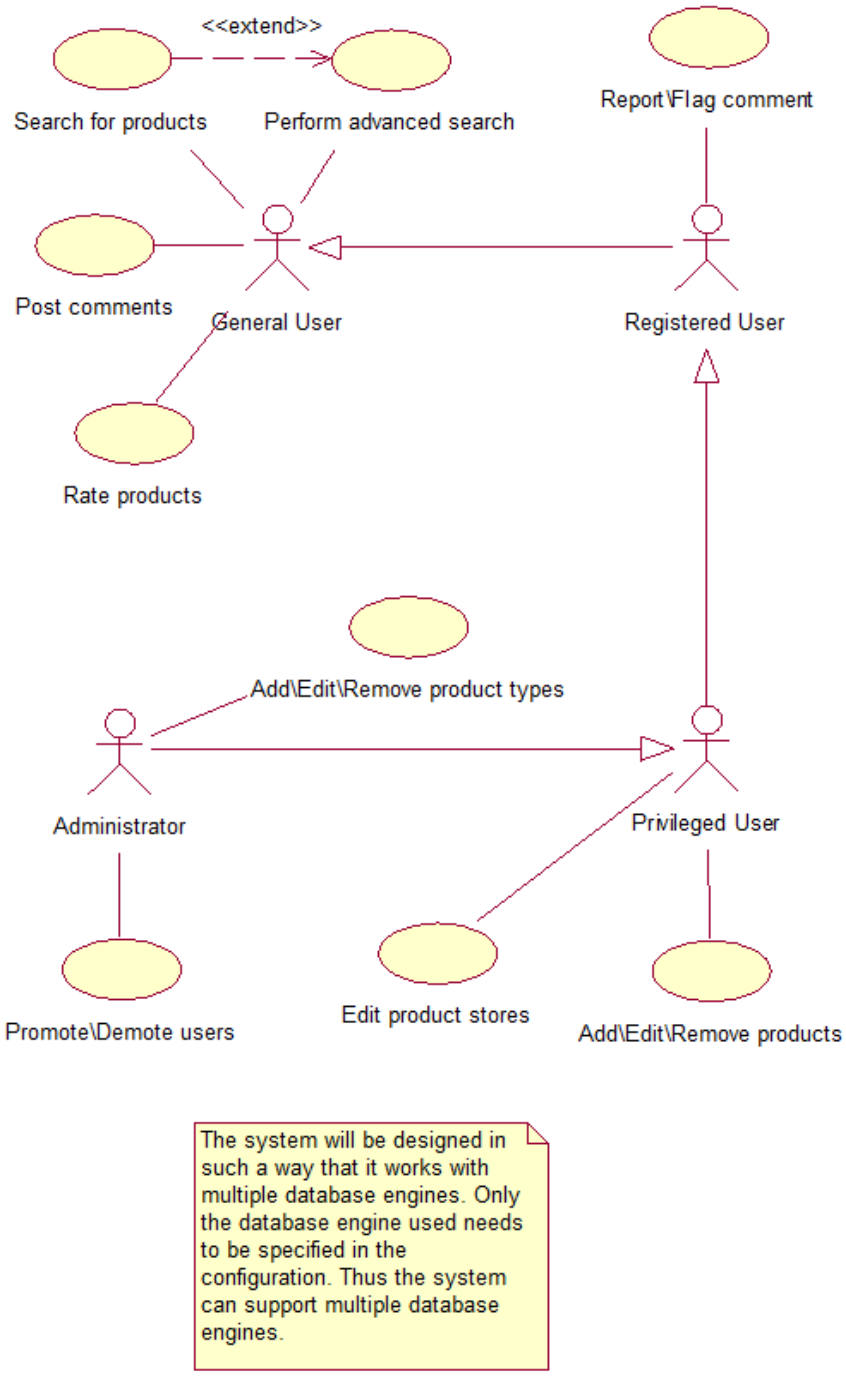
There is a fourth layer -- the persistence layer -- that deals with getting the objects of the model to permanent storage so that they can be retrieved layer. The system supports multiple persistence layers so that multiple database engines are supported.

The UML package diagram describes the architecture of the system:



2.8 Use Case Model Description

This section describes the use case view of the system.



The actors in the system are:

1) **General User:** Represents any person that uses the system, including those who have not registered.
2) **Registered User:** Represents persons that have registered.
3) **Privileged User:** Represents registered users that have elevated privileges.
4) **Administrator:** Represents the most powerful class of users.

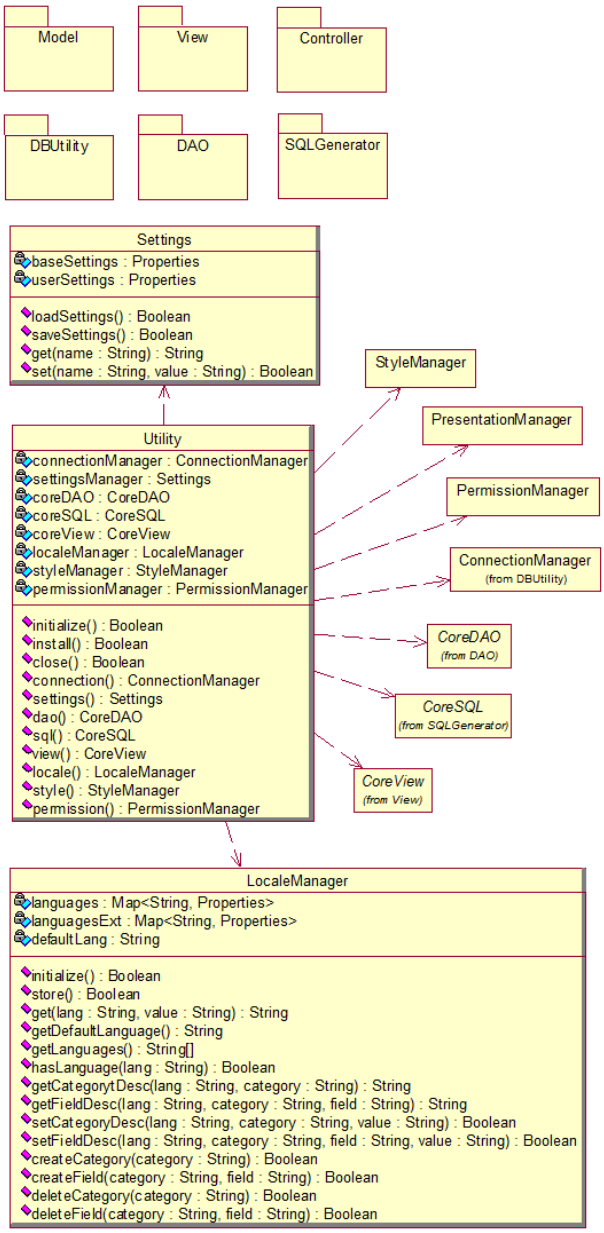
The use cases are described in detail in section 3.1.

2.9 Class Diagram

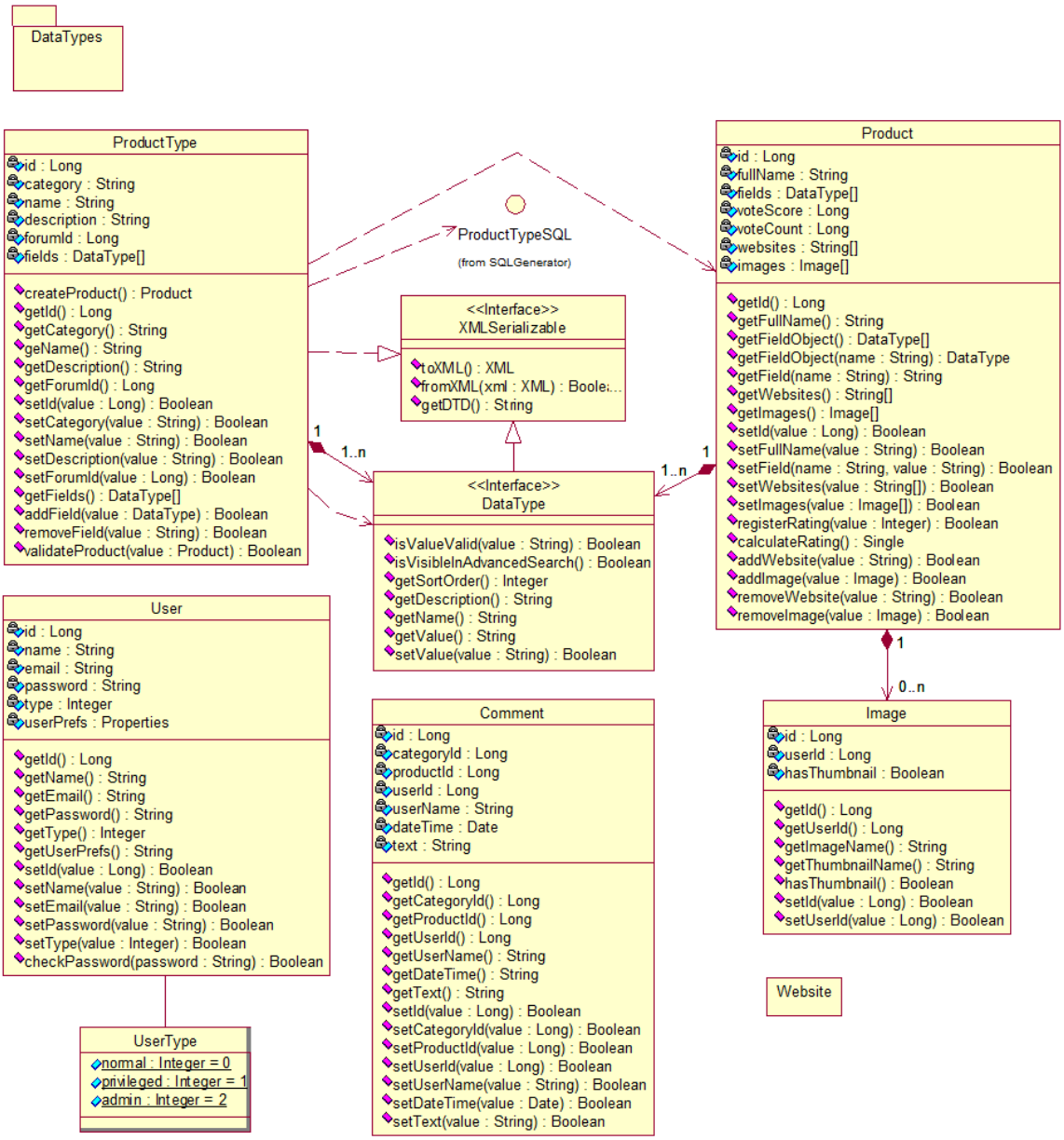
This section describes the logical view of the system.

- **The root package:**

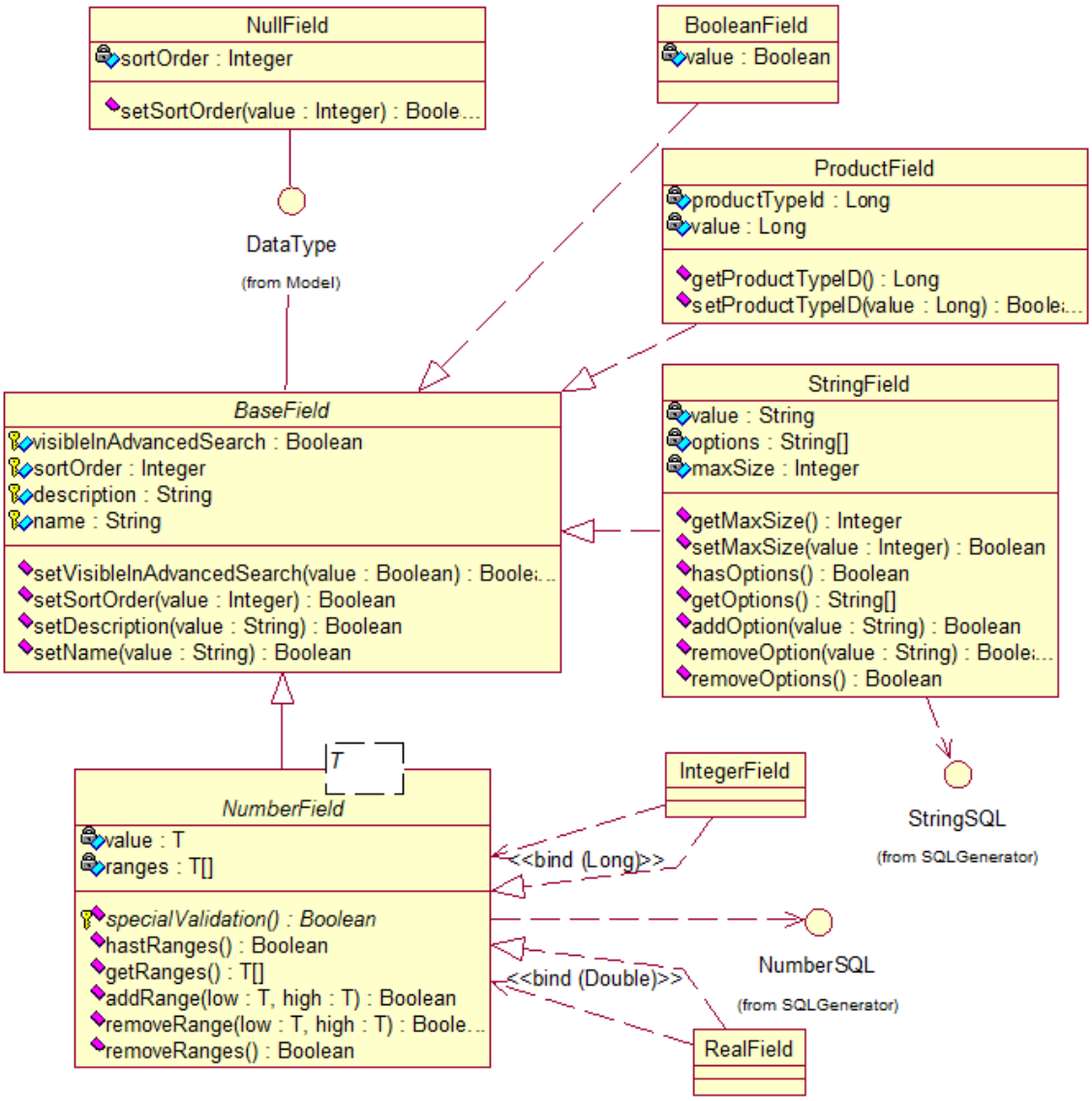
(for detailed documentation, see the accompanying Rational Rose model)



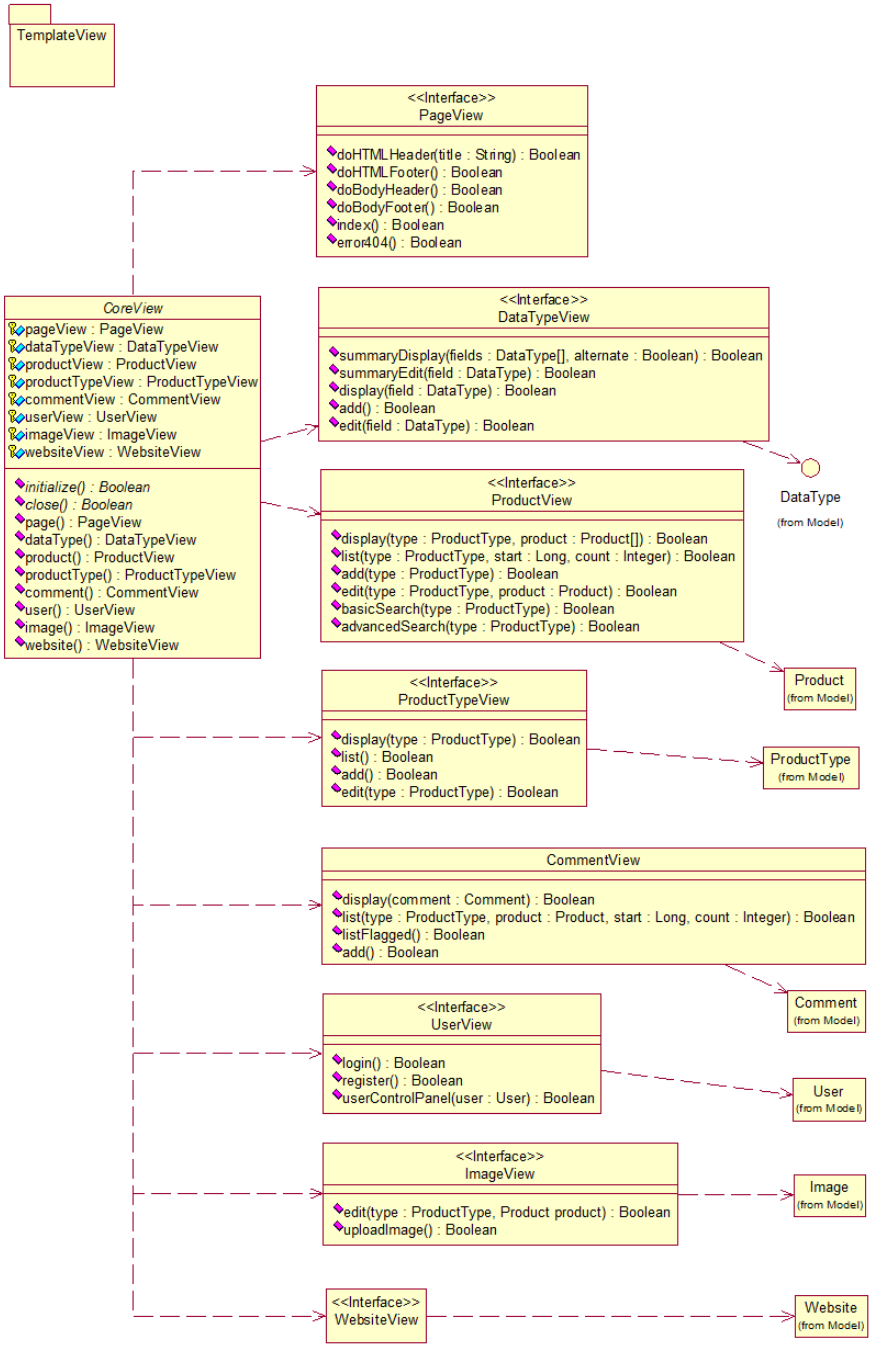
- **"Model" package:** This package contains the classes which are the "model" of the system. It contains classes representing the product, it's various fields, the various data-types fields may be of, the comments that can be made by users and the users itself.



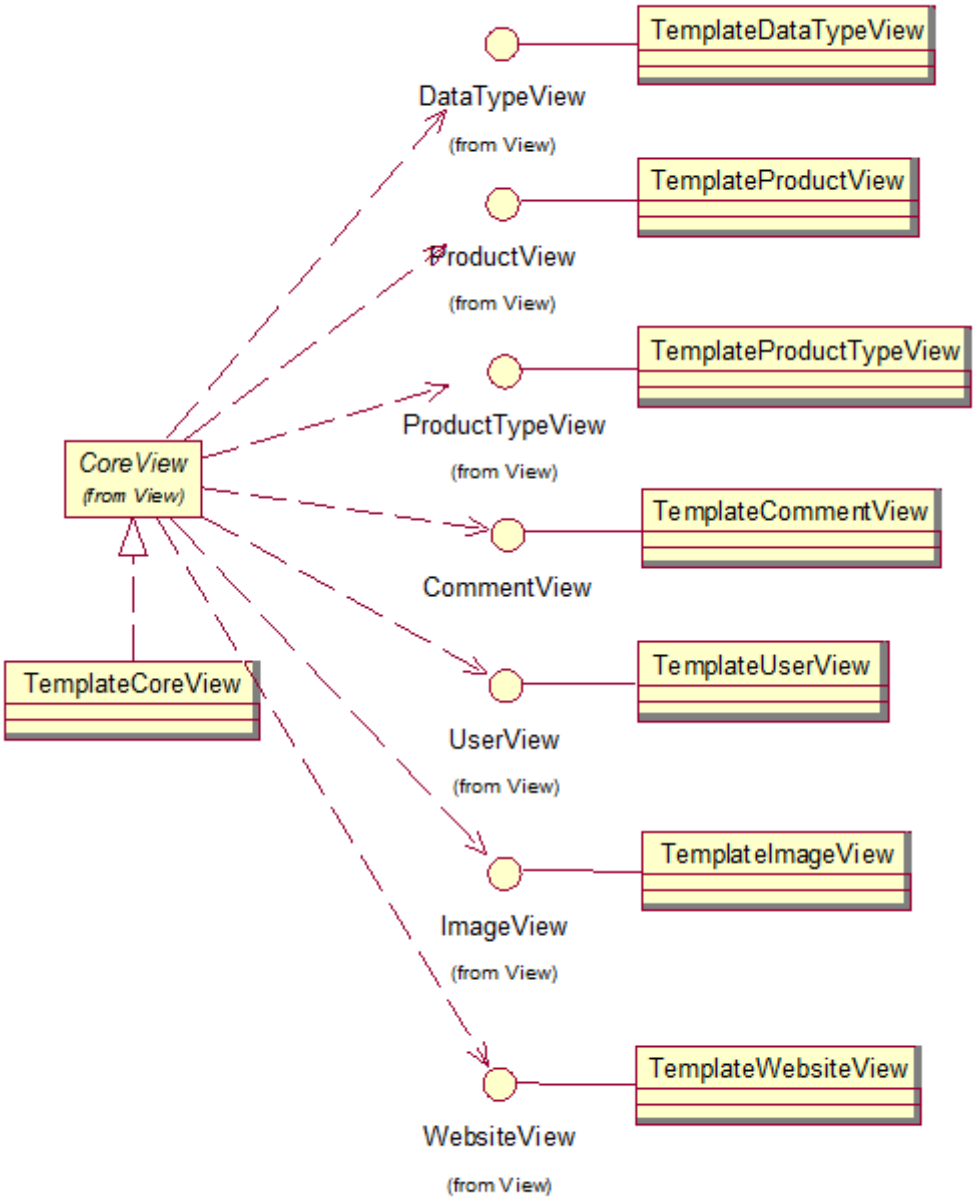
- **"DataTypes" package:** This package contains classes representing various data-types like Boolean, Integer (i.e. long), Real (i.e. Double) and String (which can also double up as one-of-many option field).



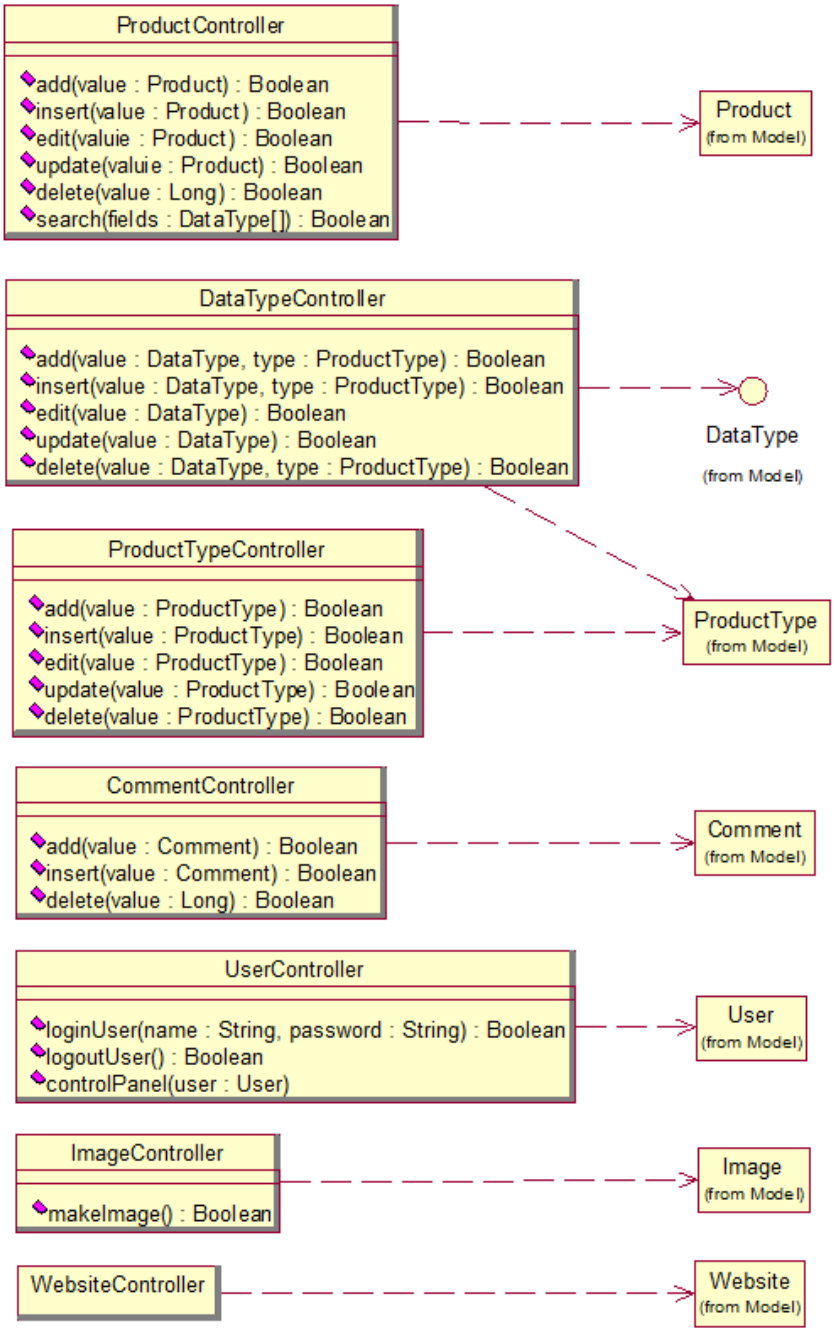
- **"View" package:** This package contains the classes which form the "view" of the system. It contains classes for generating HTML pages, for both PCs and mobile devices.



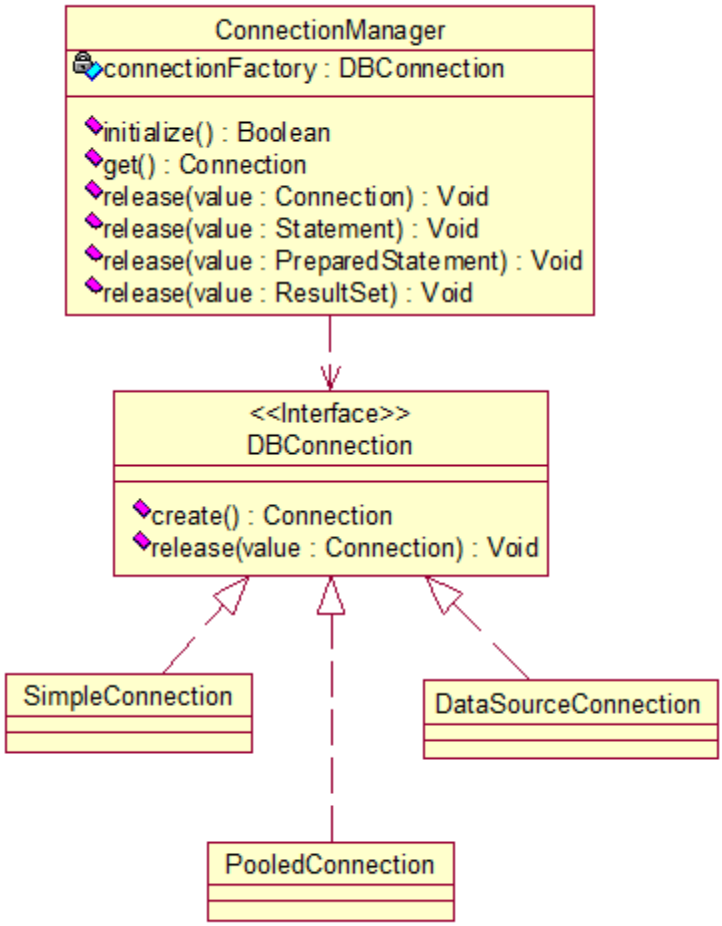
- **"TemplateView" package:** Freemarker Template View.



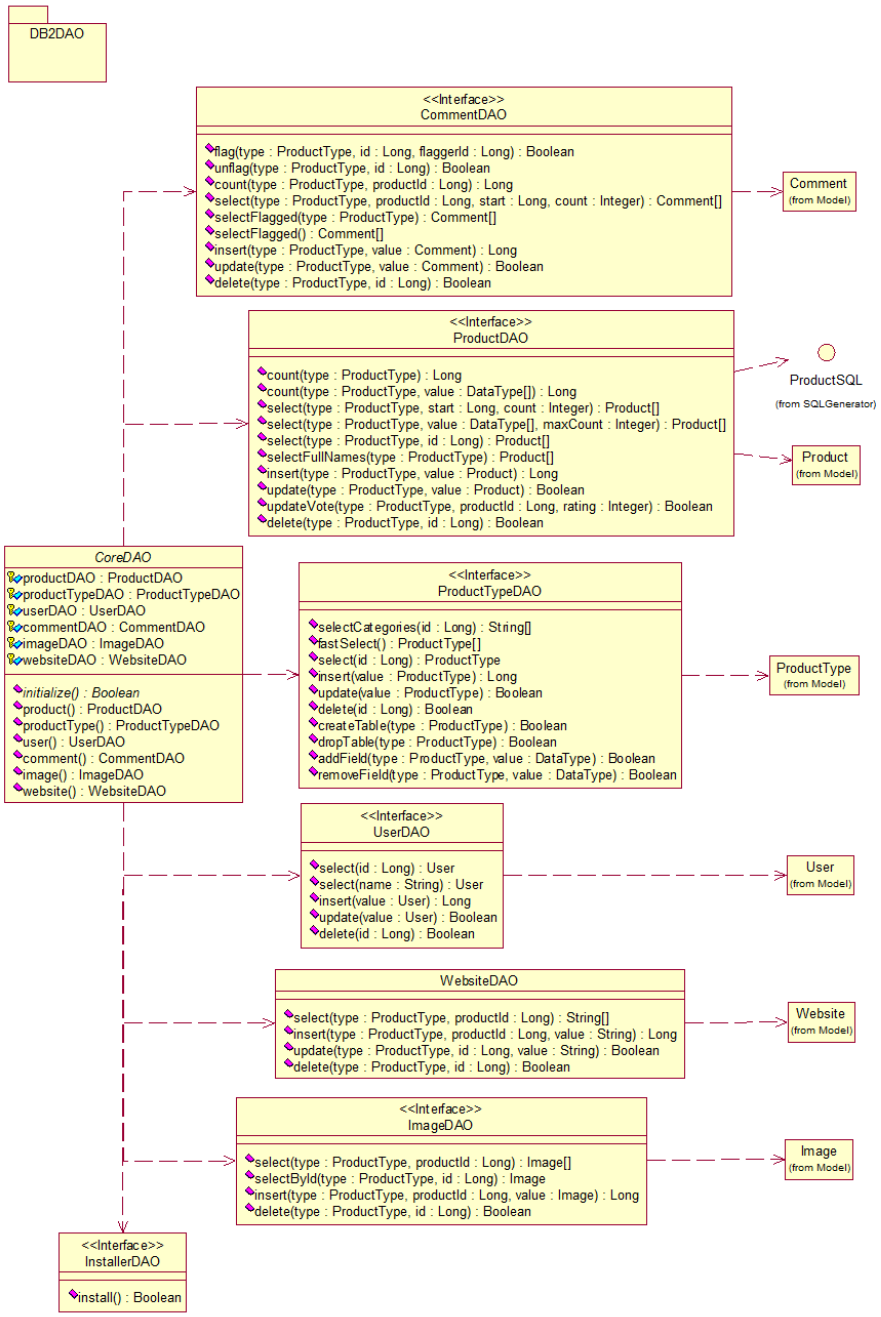
- **"Controller" package:** This package contains classes which form the "controller" of the system. It contains classes for co-ordinating the model and the view.



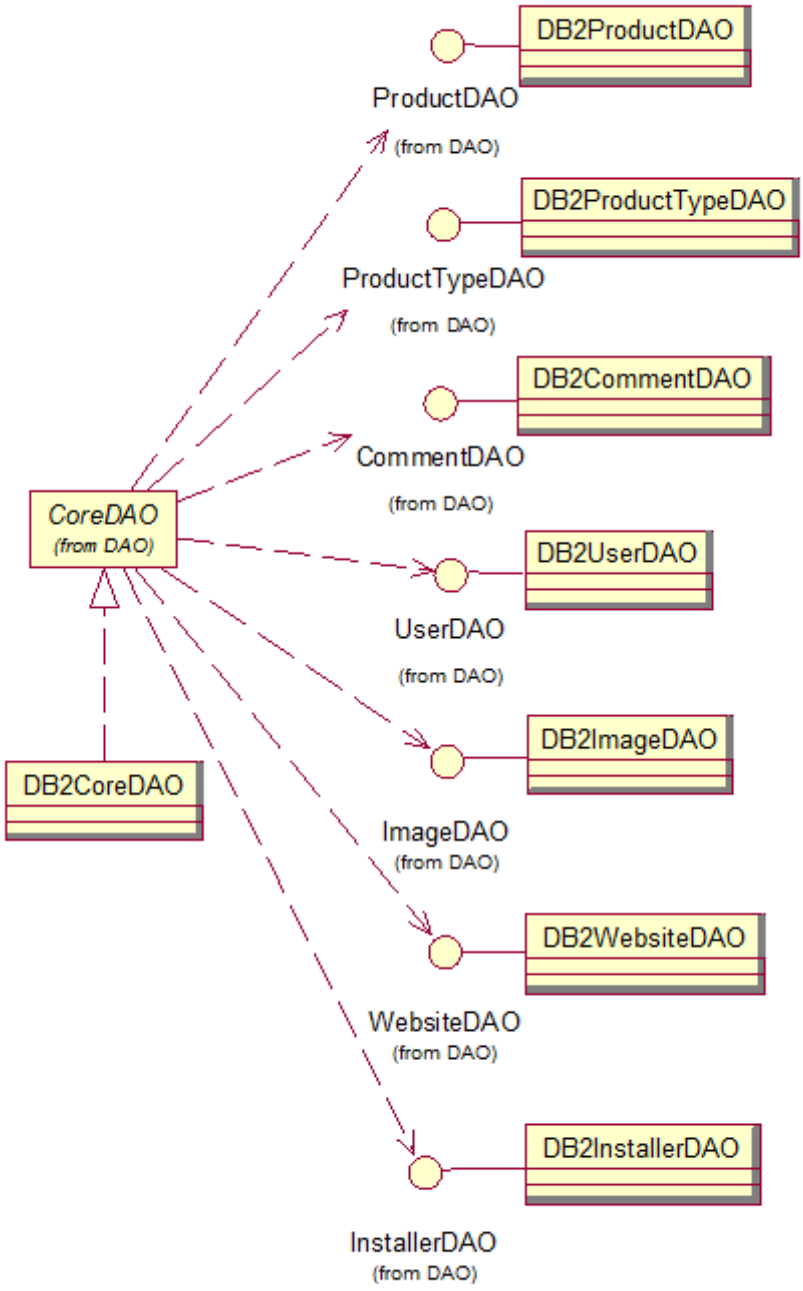
- **"DBUtility" package:** Package containing classes for various types of connection factories (simple and pooled) and utility database code.



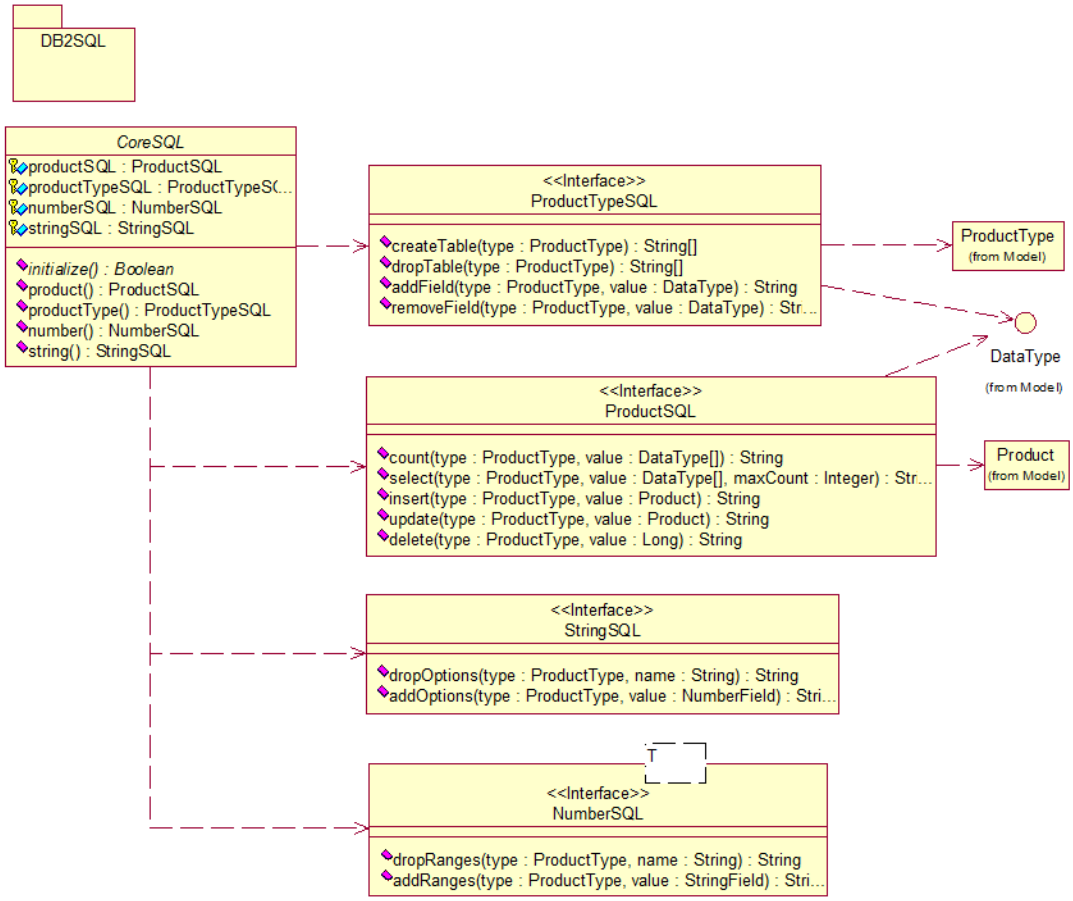
- **"DAO" package:** This package contains interfaces, classes and packages for providing uniform access to various database engine.



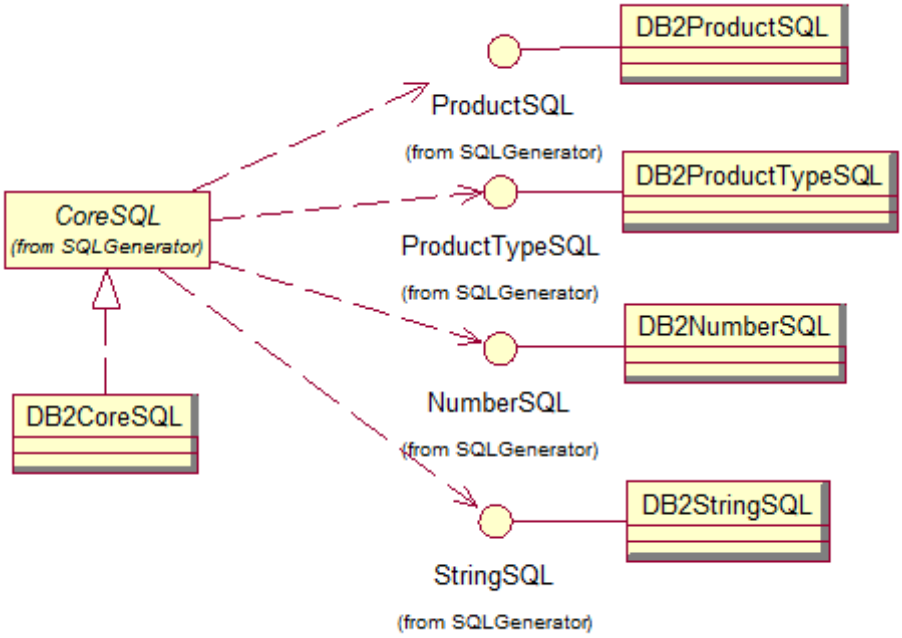
- **"DB2DAO" package:** DAO for IBM DB2.



- **"SQLGenerator" package:** This package contains interfaces, classes and packages for providing uniform access to SQL generators for various database engines.

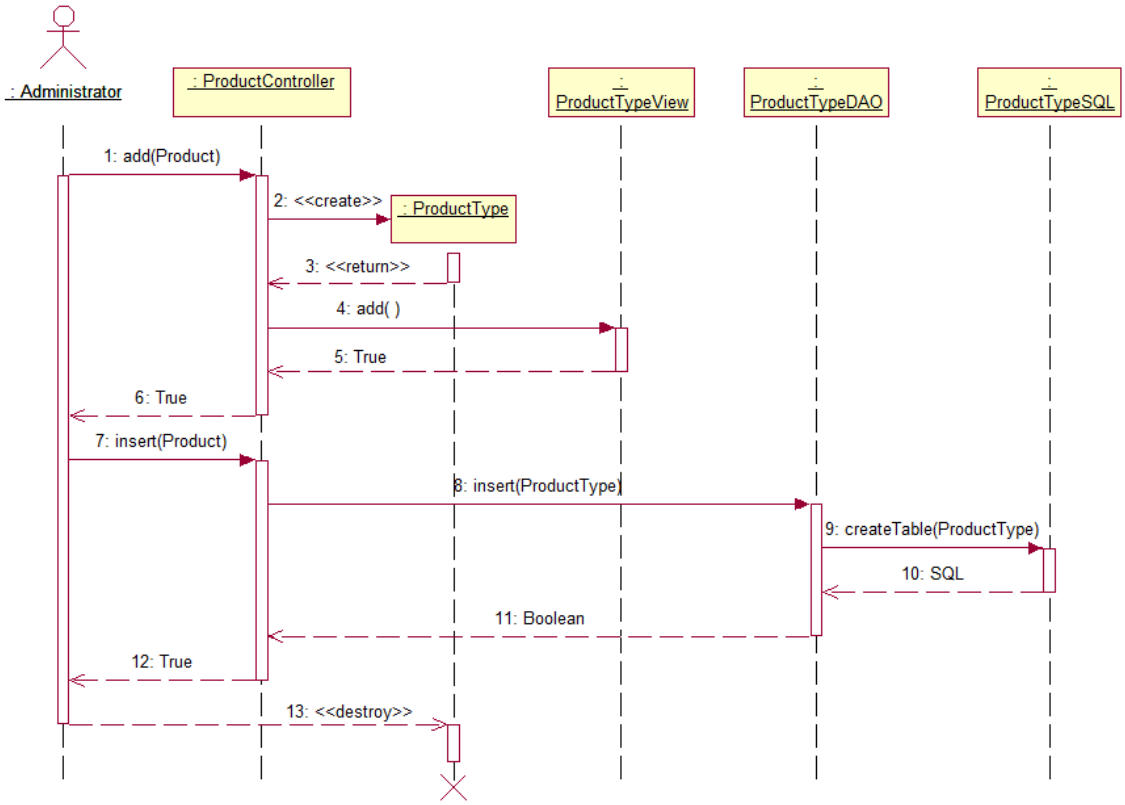


- **"DB2SQL" package:** SQL Generator for IBM DB2.

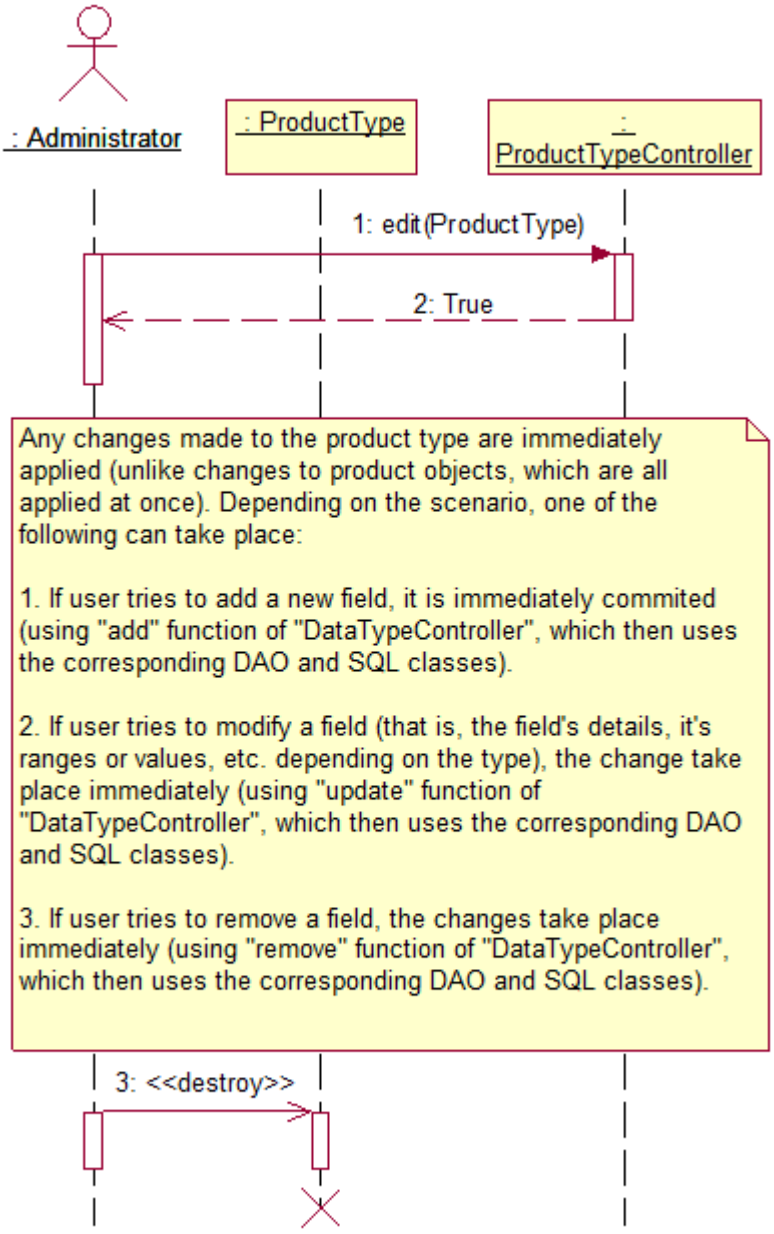


2.10 Sequence Diagrams

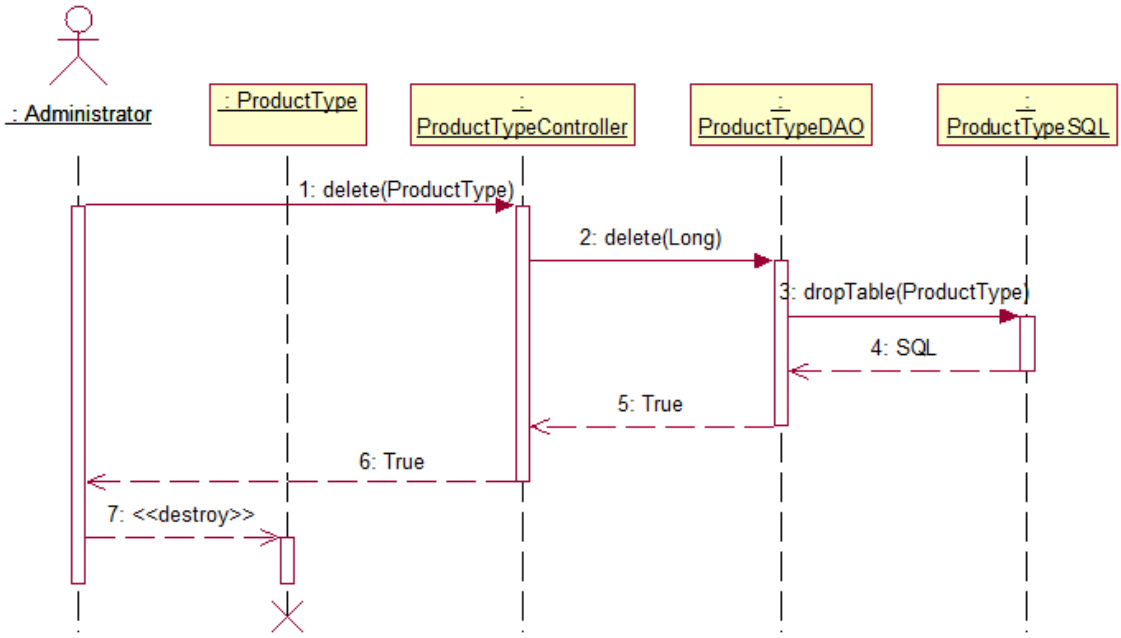
- **Add product type:**



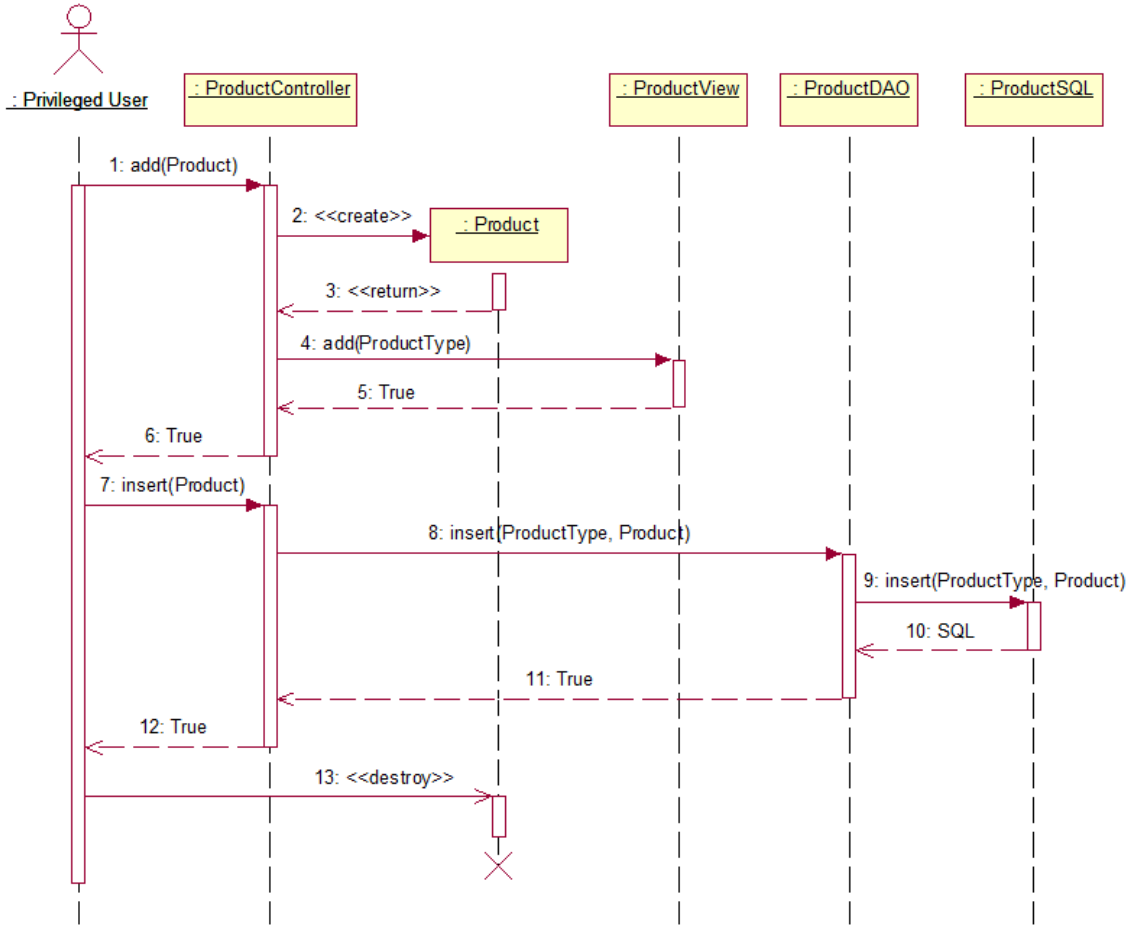
- **Edit product type:**



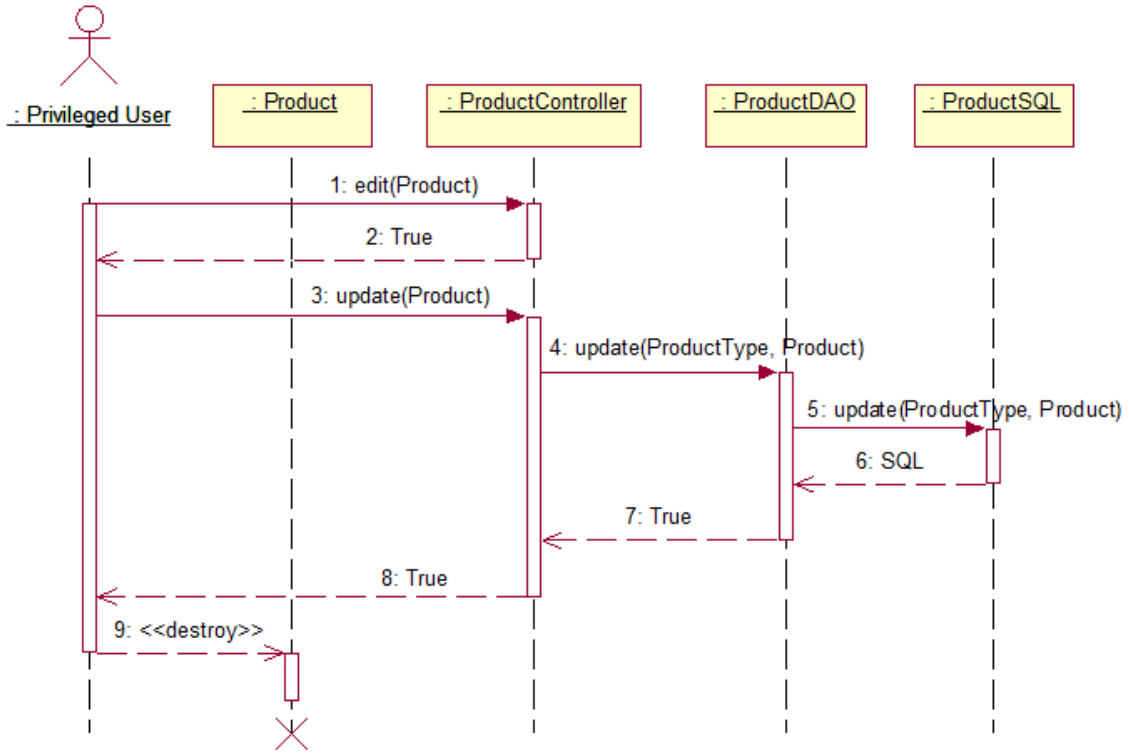
- **Remove product type:**



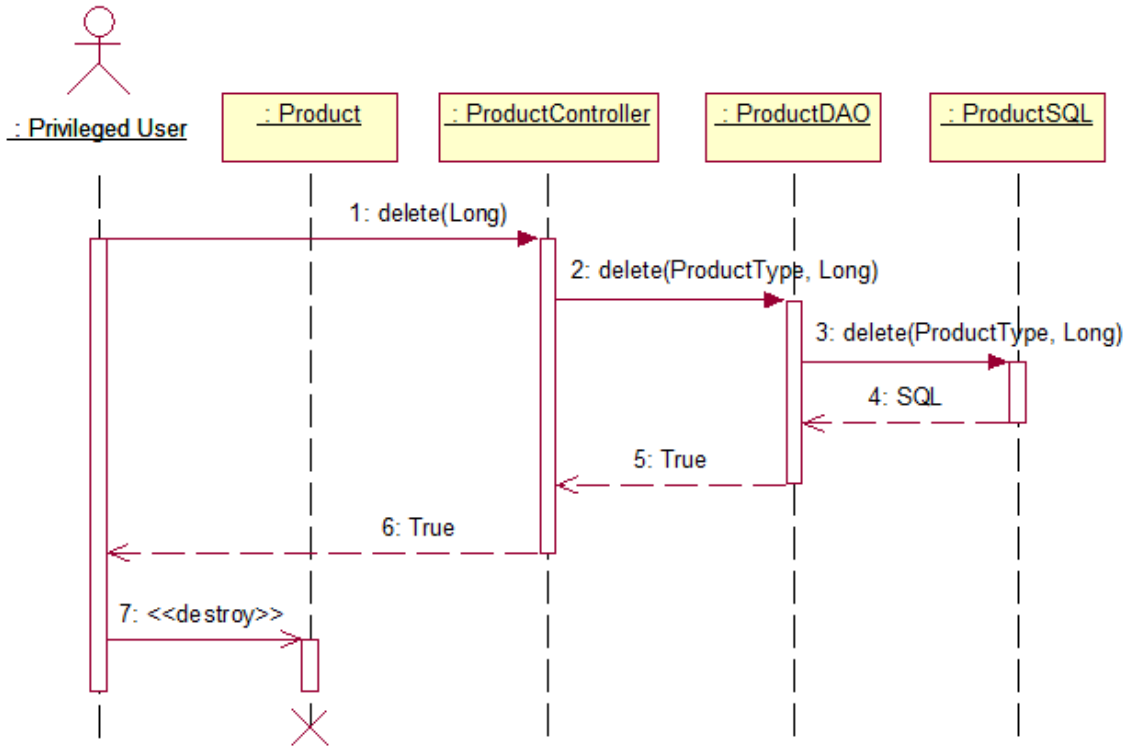
- **Add product:**


: Privileged User
: ProductController
: ProductView
: ProductDAO
: ProductSQL
1: add(Product)
2: <<create>>
: Product
3: <<return>>
4: add(ProductType)
5: True
6: True
7: insert(Product)
8: insert(ProductType, Product)
9: insert(ProductType, Product)
10: SQL
11: True
12: True
13: <<destroy>>

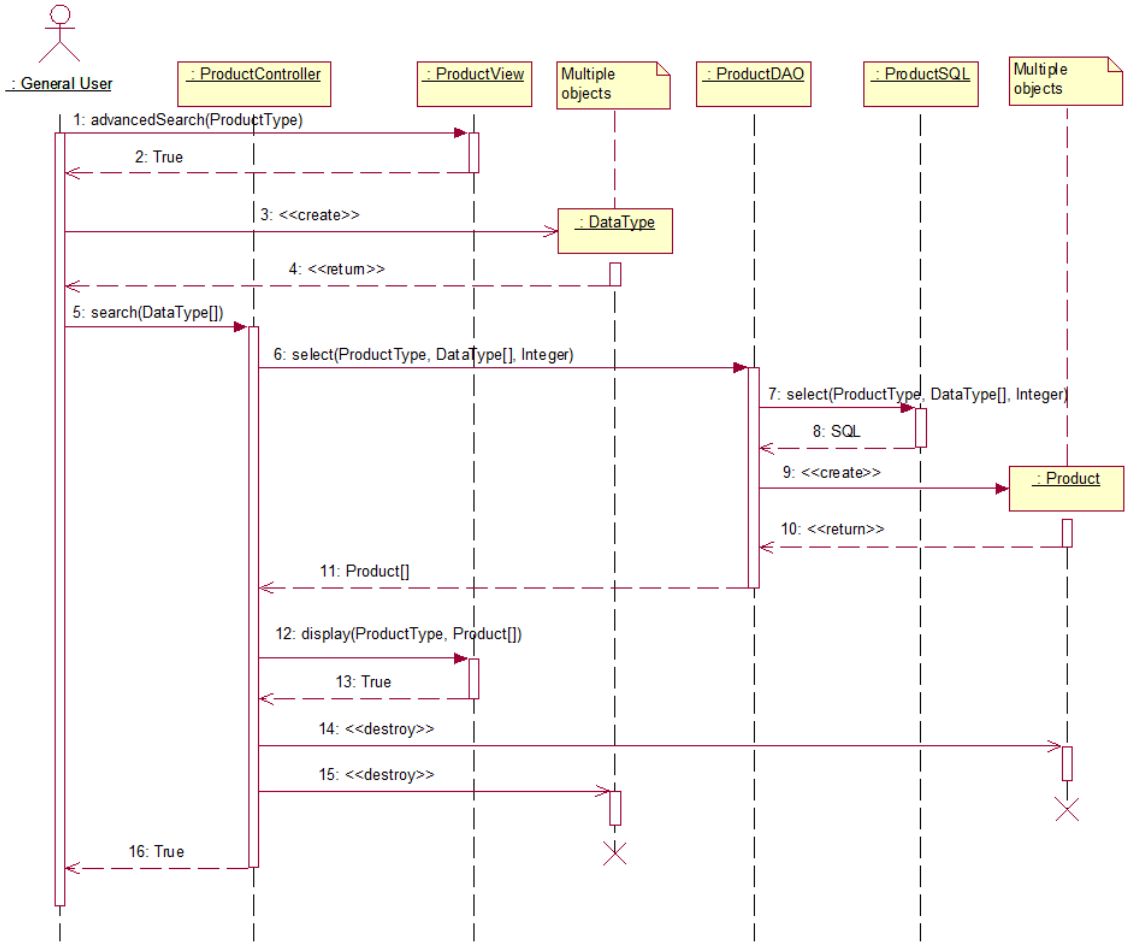
- **Edit product:**



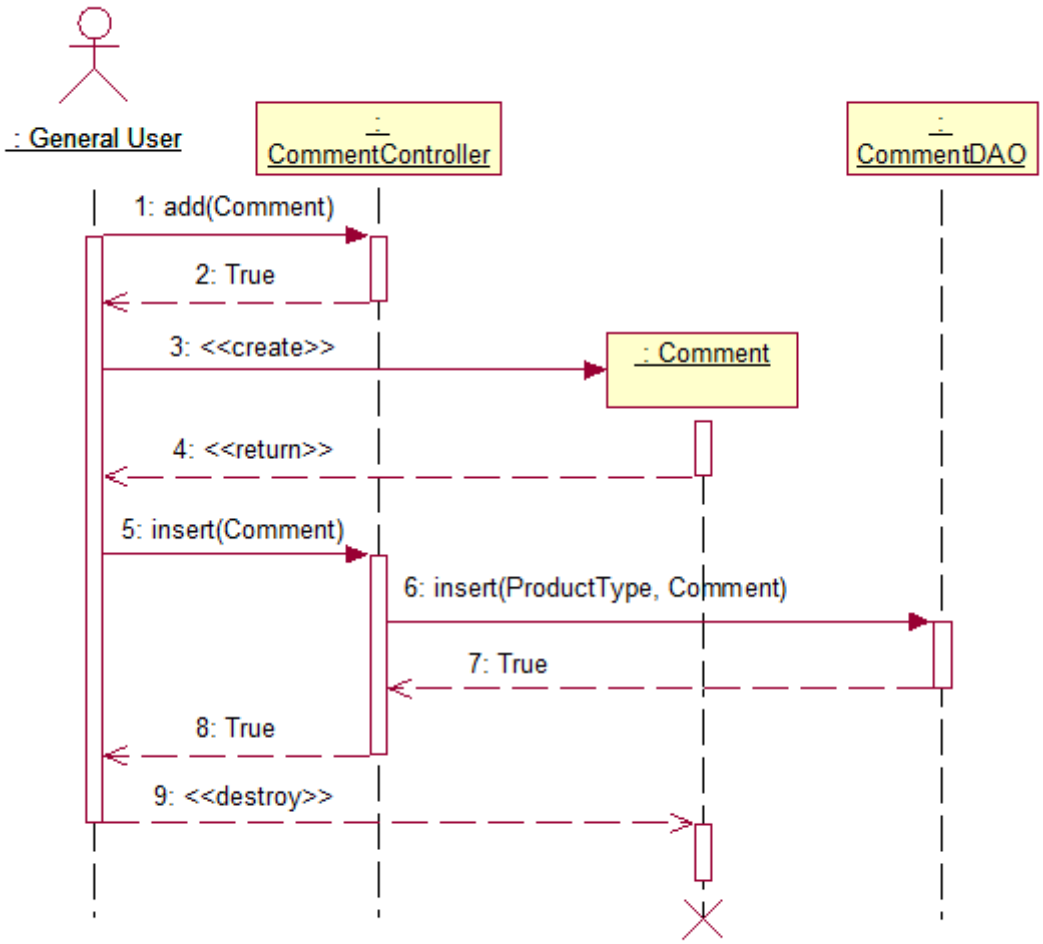
- **Remove product:**



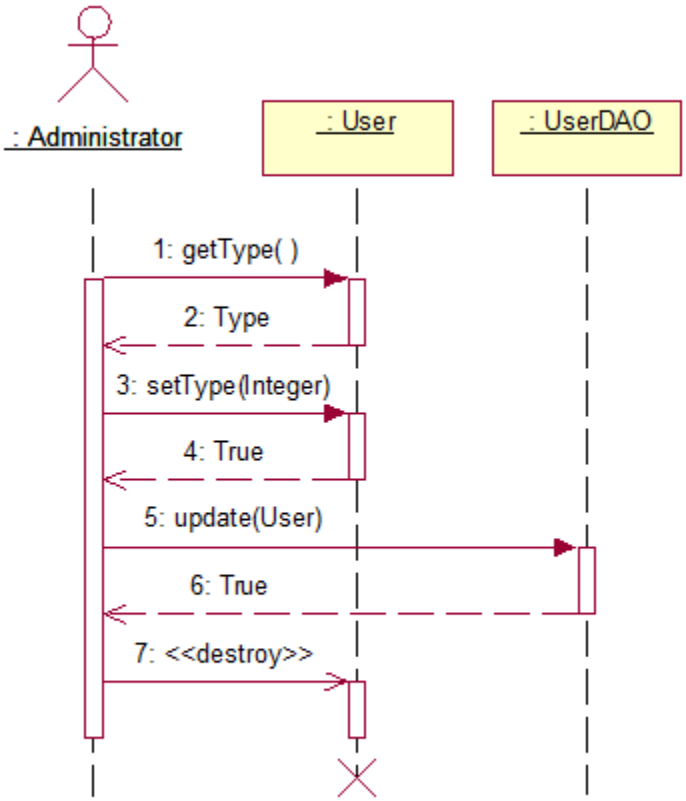
- **Perform advanced search:**



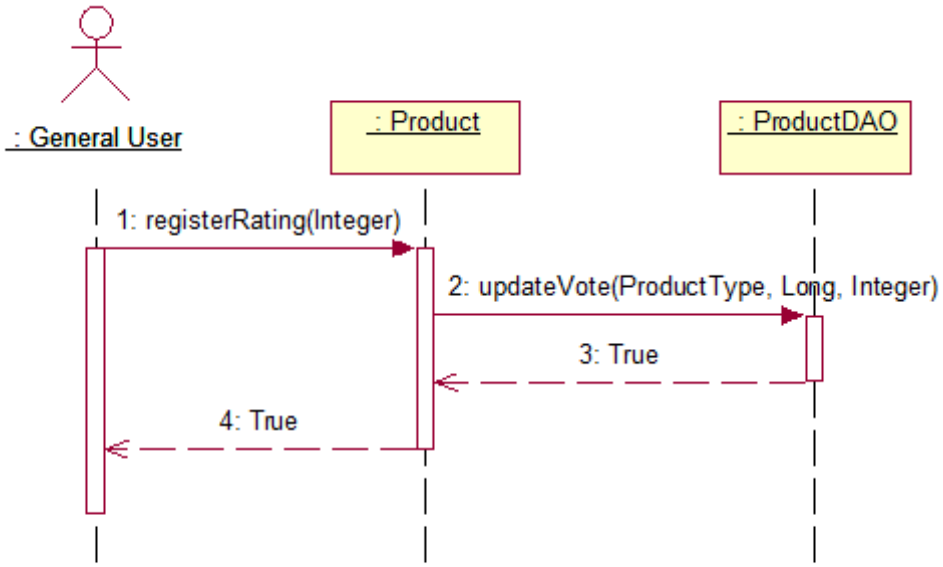
- **Post comments:**



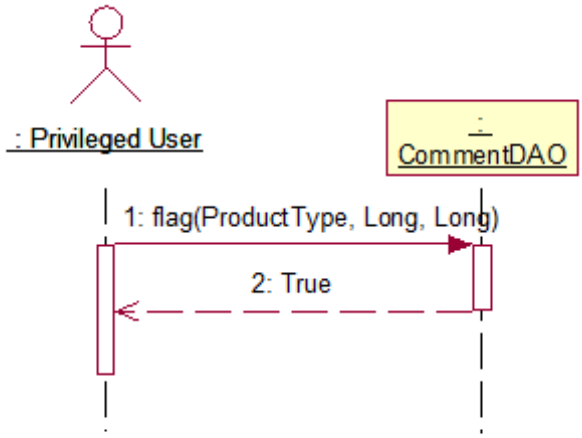
- **Promote\Demote users:**



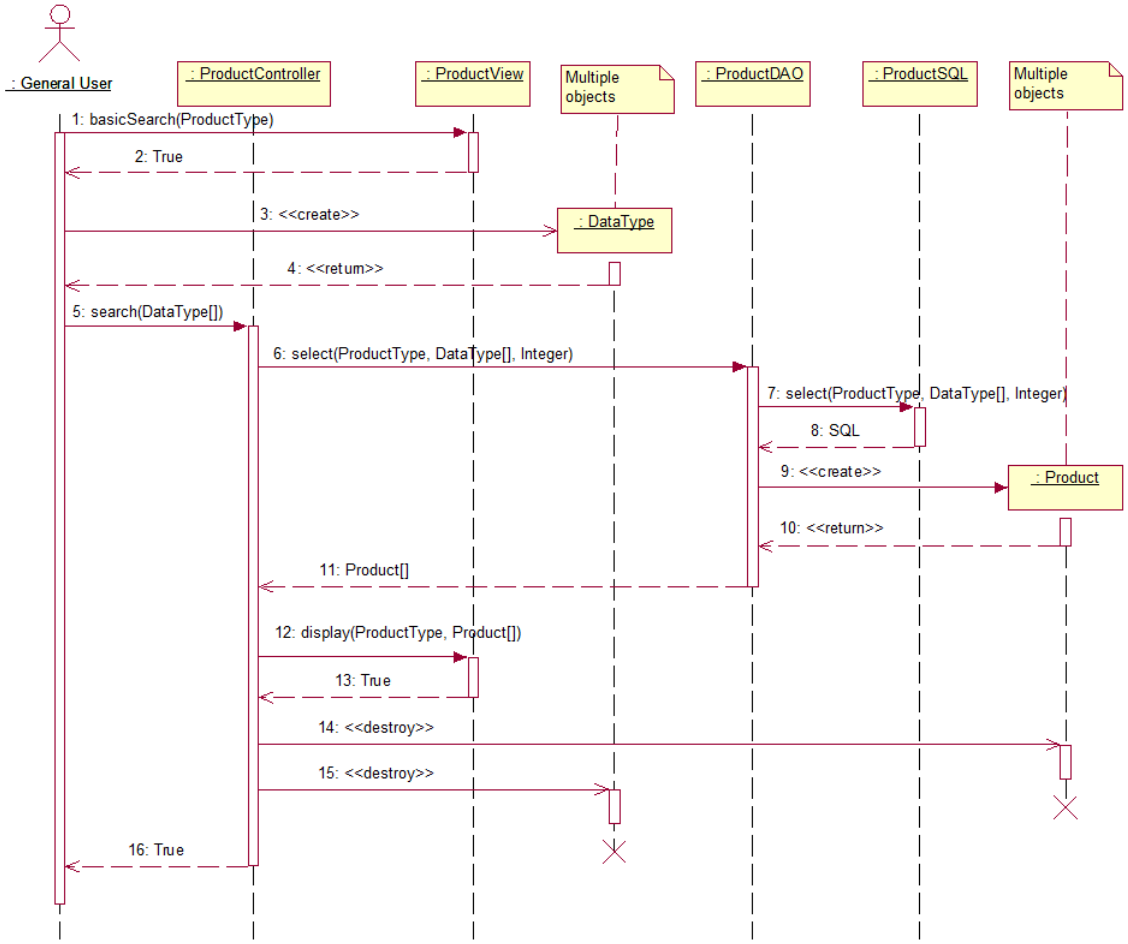
- **Rate products:**



- **Report\Flag comment:**

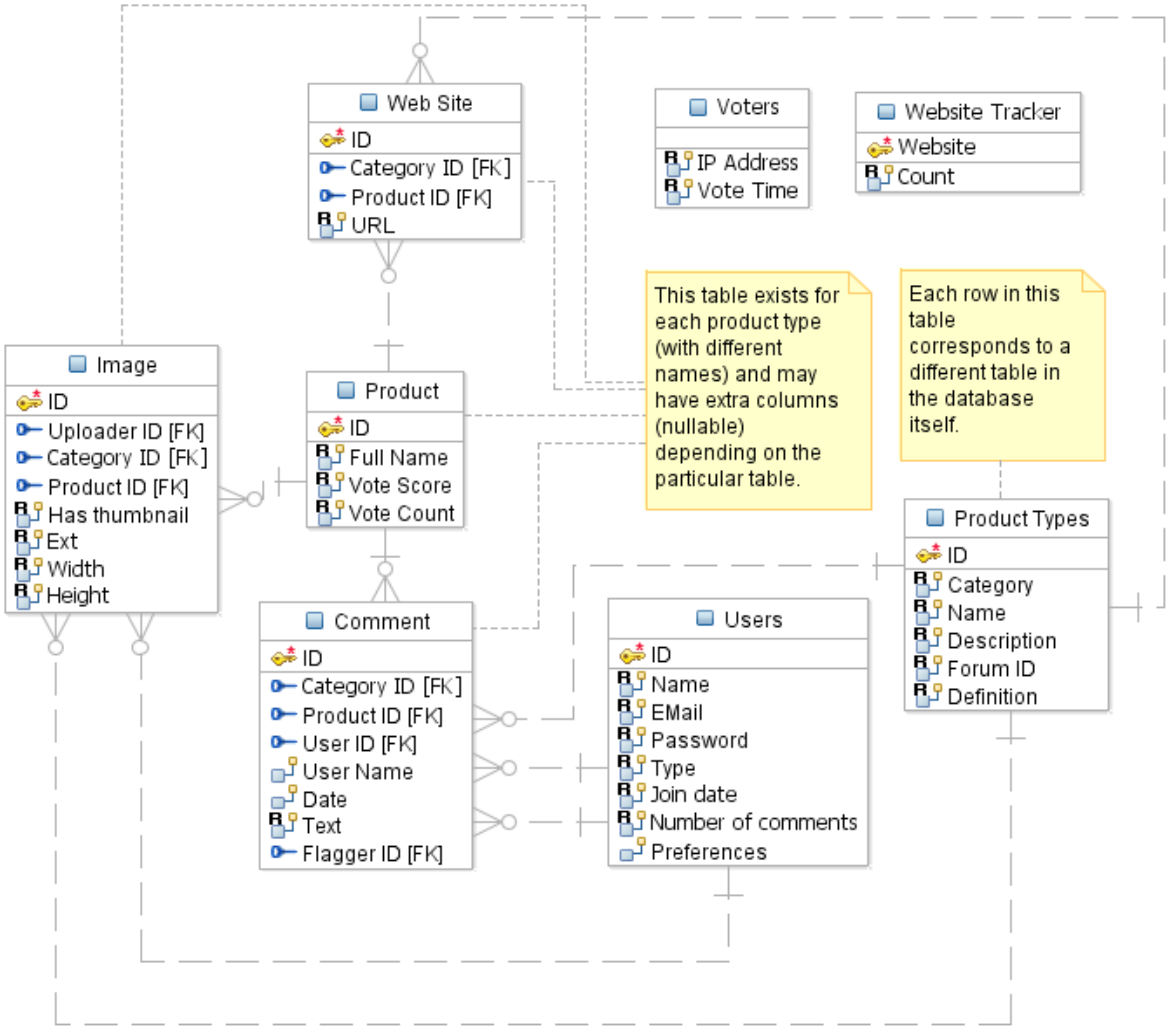


- **Search for products:**

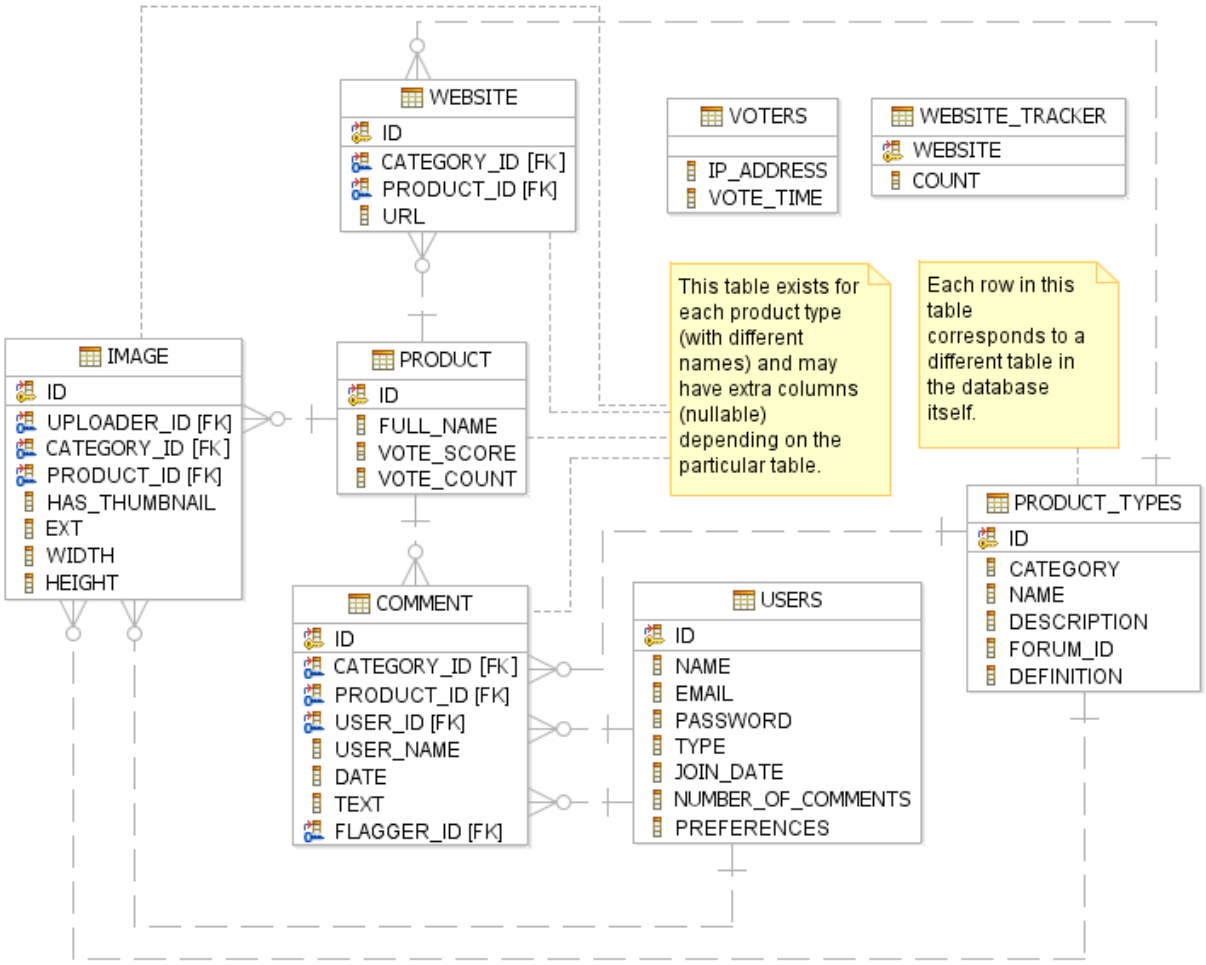


2.11 Database Design

2.11.1 ER Diagram



2.11.2 Schema


WEBSITE
ID
CATEGORY_ID [FK]
PRODUCT_ID [FK]
URL
VOTERS
IP_ADDRESS
VOTE_TIME
WEBSITE_TRACKER
WEBSITE
COUNT
This table exists for each product type (with different names) and may have extra columns (nullable) depending on the particular table.
Each row in this table corresponds to a different table in the database itself.
IMAGE
ID
UPLOADER_ID [FK]
CATEGORY_ID [FK]
PRODUCT_ID [FK]
HAS_THUMBNAIL
EXT
WIDTH
HEIGHT
PRODUCT
ID
FULL_NAME
VOTE_SCORE
VOTE_COUNT
PRODUCT_TYPES
ID
CATEGORY
NAME
DESCRIPTION
FORUM_ID
DEFINITION
COMMENT
ID
CATEGORY_ID [FK]
PRODUCT_ID [FK]
USER_ID [FK]
USER_NAME
DATE
TEXT
FLAGGER_ID [FK]
USERS
ID
NAME
EMAIL
PASSWORD
TYPE
JOIN_DATE
NUMBER_OF_COMMENTS
PREFERENCES

2.12 Assumptions and Dependencies

- It is assumed that the four data-types boolean, number, string and product reference (in which the specification of one product points to an altogether different type of product) are sufficient to describe a wide range of products.

3 Specific Requirements

3.1 Use case reports

(To view sequence diagrams for the use cases, see section 2.10)

- **Add\Edit\Remove product types:**

Main flow of events:

- This use case is entered when the user wants to manage (add, edit or remove) product types.
- If the user is adding a new product type, he\she must enter the product type details.
- If the user is editing a product type, he\she will be displayed a form with existing details which the user may edit.
- If the user is removing an existing product, he\she will be asked for confirmation.
- After receiving input from the user, the system will perform the requested action and provide confirmation that the task has been completed.
- The use case then exits.

Alternate flow of events:

- If the user cancels (or does not provide confirmation in case of removing a product type), the system does nothing.

- **Add\Edit\Remove products:**

Main flow of events:

- This use case is entered when the user wants to manage (add, edit or remove) products.
- First, the user must select a category from the UI.
- If the user is adding a new product, he\she must enter the product details.
- If the user is editing a product, he\she will be displayed a form with existing details which the user may edit.
- If the user is removing an existing product, he\she will be asked for confirmation.
- After receiving input from the user, the system will perform the requested action and provide confirmation that the task has been completed.
- The use case then exits.

Alternate flow of events:

- If the user cancels (or does not provide confirmation in case of removing a product), the system does nothing.

- **Edit product stores:**

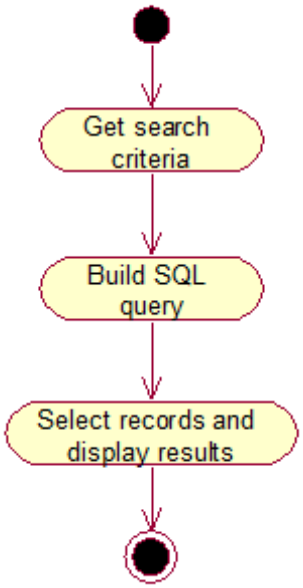
Main flow of events:

- This use case is entered when the user wants to edit stores selling a product.
- First, the user must select a product whose store he\she wants to edit.
- Next, the user may select whether he\she wants to add a new store link, or remove an existing one.
- The user then provides the requisite information (either link to the new store or a confirmation for removing an existing store).
- The system then performs the requested action.
- The use case then exits.

Alternate flow of events:

- If the user cancels (or does not provide confirmation) the system does nothing.

- **Perform advanced search:**



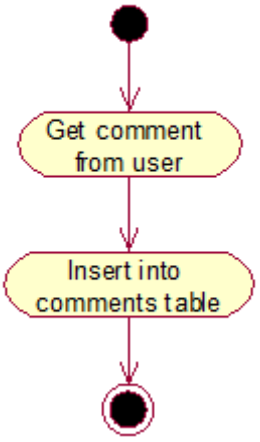
Main flow of events:

- This use case is entered when the user selects the advanced search option on the UI.
- The system shows all searchable fields for performing an advanced search.
- The user selects options from the UI and submits.
- The system then displays all products matching the given options.
- The use case then exits.

Alternate flow of events:

- If the user enters an invalid value in any field, the system informs the user and does not proceed till the error is corrected.

- **Post comments:**



Main flow of events:

- This use case is entered when the user is viewing a product and wants to comment on a product.
- The user enters his\her comment in the text box, and submits it.
- The system stores the comment and it is displayed to all uses who view the product's comments.
- The use case then exits.

Alternate flow of events:

- If the flow is interrupted anywhere before the comment is stored, the system ignores it altogether.

- **Promote\Demote users:**

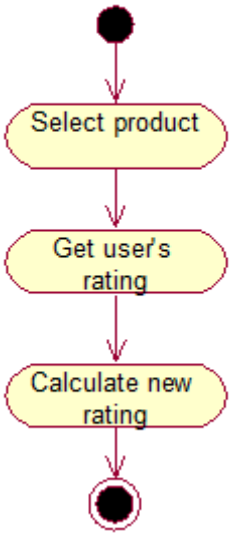
Main flow of events:

- To promote\demote a user, the user must first select the user to promote\demote.
- The user will then select whether to promote or demote the chosen user.
- The system then performs the requested action and the use case exits.

Alternate flow of events:

- If the user cancels, the system does nothing.

- **Rate products:**



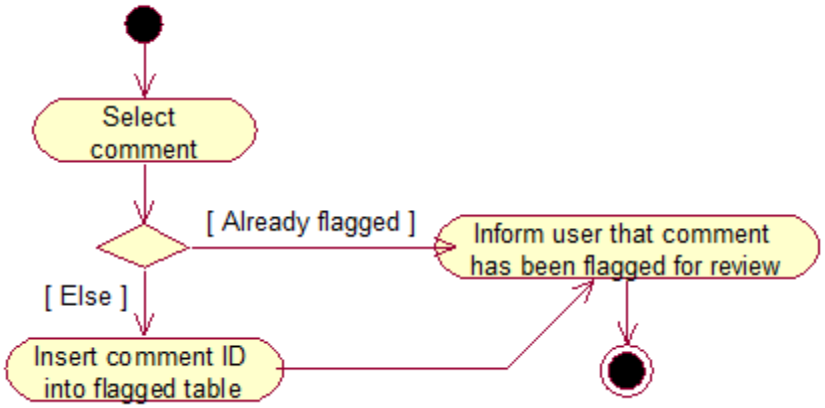
Main flow of events:

- This use case is entered when the user is seeing a product and wants to cast his\her vote for the rating of a product.
- The user selects his\her vote.
- The system registers the vote and calculates new rating.
- The user is informed that his\her vote has been accepted and the new rating is shown.
- The use case then exits.

Alternate flow of events:

- If the flow is interrupted anywhere before the vote casted is stored, the system ignores it altogether.

- **Report\Flag comments:**



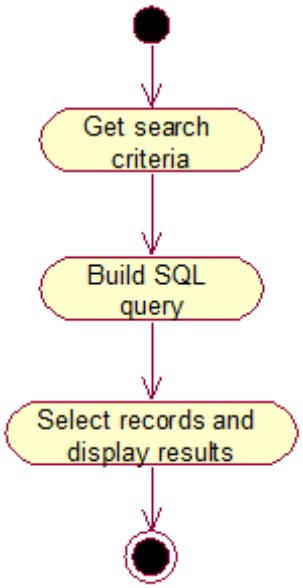
Main flow of events:

- This use case is entered when the user is viewing a comment, and flags it for review.
- If the comment was not already flagged, then it is flagged on confirmation by user.
- The user is informed that the comment has been flagged.
- The use case then exits.

Alternate flow of events:

- If the user does not confirm his\her action, the system does nothing.

- **Search for products:**



Main flow of events:

- This use case is entered when the user selects the basic search (or simply search) option on the UI.
- The system shows the basic set of fields for performing a basic search.
- The user selects options from the UI and submits.
- The system then displays all products matching the given options.
- The use case then exits.

Alternate flow of events:

- If the user enters an invalid value in any field, the system informs the user and does not proceed till the error is corrected.

3.2 Supplementary Requirements

- 24 x 7 availability - Except for the time to backup the database, the system can remain available 24 x 7.
- Better component design to get better performance at peak time.
- Flexible service based architecture - Allows for different database engines to be used with the system.