Object Oriented Analysis and Design

Course Project
Tiki.vn Website

Instructor: Assoc. Prof. Dr. Trương Ninh Thuận

Project team:

Hoàng Thị Tâm	K60CA	Student ID: 15021603
Bùi Mạnh Thắng	K60CA	Student ID: 15021058
Nguyễn Đăng Thế	K60CA	Student ID: 15020912

Table of Contents:

I. REQUIREMENT

1.1 Problem statement

1.1.1 Addressing the problem

Online shopping also known as "electronic retail" or "e-shopping" has seen an impressive development together with the growth of the Internet. As a matter of fact, the richest man in the world owns the world’s biggest electronic commerce company (Amazon), and the second richest man in China also built one of the world's largest and most valuable Internet retailers (Alibaba). Modern people now buy things with Browsers, not with shopping cart. And Vietnamese market does not stand aside with dozens of Online Retailers gaining more and more popularity everyday. As a result, systems have to be developed so that providers have a place to advertise and sell their products while consumers have a place to spend their money.

1.1.2 Solution

Tiki.vn (founded in 3/2010) is a online system where Tiki company themself and providers (sellers) who have contracts with Tiki and consumers (buyers) can interact with each other.

Description

The system being developed is a web application which can be accessed by dozens of internet-connected devices like PC, smartphones, tablets and the like.

All types of users are **Visitors**, **Customers**, **Providers** and **Administrators** (Admins for short).

Customers are people who already have Tiki accounts while Visitors are the ones who does not have.

Customers and Visitors go to the website, see tons of products and select whatever they like for their order, after receiving the orders, they can rate and comment about the items they've got. Customers have to login their accounts while Visitors have to create new ones to use certain functions. They can access information about advertisements, sale-off, ratings, delivery process, guarantee ... of the products and theirs providers.

Providers are shop owners who have signed contracts with Tiki so that their products can appear on the website. Providers have responsibility to act in compliance with terms provided by Tiki and sell legal and genuine products to the customers. They have their own pages displaying what their are selling and sale-off programs to attract buyers. Tiki itself is also the provider of thousands of items on the website.

Administrators (Admins) are responsible for providing services for website's members and monitor all of the Customers' and Providers' accounts. They can end contract with users who act illegally or harm the common market environment by cheating, spamming or distorting the informations.

1.2 Glossary

Introduction

This document is used to define terminology specific to the problem domain, explaining
terms, which may be unfamiliar to the reader of the use-case descriptions or other project documents. Often, this document can be used as an informal data dictionary, capturing data definitions so that use-case descriptions another project documents can focus on what the system must do with the information.

Definitions

The glossary contains the working definitions for the key concepts in the Tiki.vn website.

Account

A record about a customer/administrator/provider containing information about his/her name, e-mail address, password, phone number and optional self-introduction. Each account has a unique user ID and a password, which are used to identify the user/administrator and grant them access to secure parts of the system.

Administrator

A person whose job is to ensure that the site is free of spam advertisements or abusive behaviours and operate the system. This entails approving advertisements before they are published, deleting reported advertisements and deleting users with abusive behaviours.

Member

Providers, Customers have registered account on website Tiki.vn

Provider/Seller

A person or a organization has legal commercial activity recognised by Tiki, allowed to use Tiki's services. Sellers can create showrooms to introduce, sell their products or do other activities to serve customers.

Customer

A person or a organization purchase products or using services on Tiki's website. Each customer have an account to perform transactions or access private data.

Visitor

A person who is interested in viewing advertisements on the website but does not have an account.

Product

Goods or services provided by sellers through Tiki's website. Products are verified by Tiki's for quality and license. Customers can buy products.

1.3 Supplementary specifications

Objectives

The purpose of this document is to define requirements of the Tiki.vn system. This Supplementary Specification lists the requirements that are not rapidly captured in the use case of the use-case model. The Supplementary Specification and the use-case model together capture a complete set of requirements on the system.

Scope

This Supplementary Specification applies to the Tiki.vn system, which is an electronic commerce website in Vietnam. This specification defines the non-functional requirements of the system: such as reliability, usability, performance, and supportability, as well as functional requirements that are common across a number of use cases. (The functional requirements are defined in the Use Case Specification.)

References

None.

Functionality

Multiple users must be able to perform their work concurrently.

Usability

The software must be easy to use so that a new user can learn how to use the system within 1 hour.

The user interface has to be friendly and intuitive.

Reliability

The system must be available 24 hours a day, 7 days a week. The system must also have less than 5% downtime.

Performance

The system shall support up to 2000 simultaneously users against the central database, and up to 100 simultaneous users against the local servers at any given time.

The system shall provide access to the database with no more than 5 seconds latency. The system must be able to complete 90% of all transactions within 30 seconds.

Supportability

None.

Security

The system must prevent Users from logging in if they do not have a password.

Only Administrators can delete User accounts.

An product can only be edited by its Provider and deleted by its Seller or an Administrator.

Only Administrators can approve pending order.

Design Constraints

The system must provide a responsive web-based interface usable on computers and mobile devices.

1.4 Use-case models

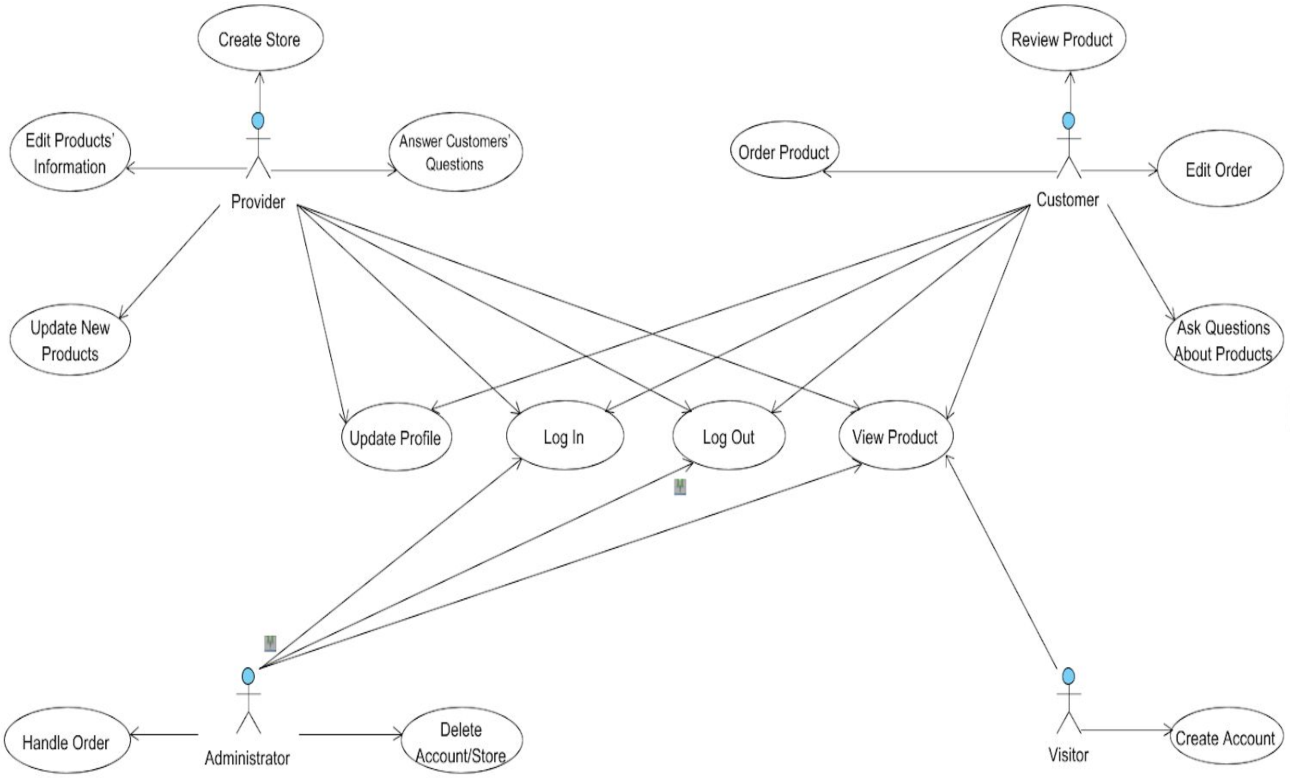


Figure 1.1: Use-case Diagram

1.4.1 Create Account

Brief Description	This use case describes how a visitor registers for an account on the Tiki.vn website.
Basic Flow	This use case starts when the visitor requests to create an account on the website. 1. The system displays a form which asks visitor to enter the following information: ● Full name

	● Email address or phone number ● Password ● Gender ● Date of birth 2. Once the Visitor provides the requested information, the system verifies that the user’s email or phone is unique and all required fields are specified. It then adds a new account with the specified information to the database.
Alternative Flows	**Missing information:** If any required fields are not filled in, the system display an error message. The visitor can continue making changes to the registration form or cancel the registration, at this point the use case ends. **User ID already exists:** If the specified ID already exists, the system display an error message. The visitor can continue making changes to the registration form or cancel the registration, at this point the use case ends.
Special Requirements	None
Pre-conditions	None
Post-conditions	If the use case was successful, a new user is added to the system. Otherwise, the system state remains unchanged.
Extension Points	None

1.4.2 Log In

Brief Description	This use case describes how a Customer, a Provider or an Administrator logs into the Website.
Basic Flow	This use case starts when the actor wishes to log into the Website.

	1. The actor enters his/her username and password. 2. The system validates the entered username and password and logs the actor into the system.
Alternative Flows	**Invalid/Error Information** If, in the **Basic Flow**, the actor enters an invalid or error username and/or password, the system displays an error message. The actor can choose to either return to the beginning of the **Basic Flow** or cancel the login, at which point the use case ends. **Missing Required Information** If any fields in the form are left empty, the system displays an error message. The actor can continue modifying the form or cancel the operation, at which point the use case ends.
Special Requirements	None.
Pre-conditions	The system is in the login state and has the login screen displayed.
Post-conditions	If the use case is successful, the actor is now logged into the system and the system then navigates to the previous window. Otherwise, the system state is unchanged.
Extension Points	None

1.4.3 Log Out

Brief Description	This use case describes how a Customer, a Provider or an Administrator logs out the Website.
Basic Flow	This use case starts when the actor wishes to log out the Website.

	1. The actor click “Account” Menu. 2. Choose “Log out” option.
Alternative Flows	None.
Special Requirements	None.
Pre-conditions	The User must be logged into the system.
Post-conditions	If the use case is successful, the actor is now logged out the system and the system then navigates to the previous window. Otherwise, the system state is unchanged.
Extension Points	None.

1.4.4 Update Profile

Brief Description	This use case describes how a User updates his/her account in the system.
Basic Flow	1. The system displays the following information in editable fields and asks the User to make changes to their account: ● Name ● Password ● Gender ● Phone number /Email address (add not edit) ● Date of Birth (optional) 2. Once the User provides the requested information, the system verifies that all required fields are specified, and updates the account’s information on the user database.
Alternative Flows	Missing Information: If any of the above fields (except for Date of birth) are not filled in, the system displays an error message. The

	actor can continue making changes to the registration form or cancel the registration, at which point the use case ends.
Special Requirements	None.
Pre-conditions	The User must be logged into the system.
Post-conditions	If the use case was successful, the account's information is updated. Otherwise, the system state remains unchanged.
Extension Points	None.

1.4.5 Create Store

Brief Description	This use case describes how a provider create their own store on Tiki's website.
Basic Flow	The use case starts when a provider wishes to create a store for products selling 1. The system display a form and requests to complete the following information ● Contact information: full name, email, phone number, hotline, store name. ● Business information: company name, address, bank account information, etc. 2. Then the provider is required to register a list of products that will be sold on Tiki's website. 3. Once all required information is provided, the system notifies the administrator for verification and adds a new store to approval queue.
Alternative Flows	**Missing Required Information:** If any of the required

	fields are not filled in or selected, the system displays an error message. The User can continue making changes to the form or cancel the registration, at which point the use case ends.
Special Requirements	None.
Pre-conditions	None.
Post-conditions	If the use case was successful, a store is created and scheduled for approval. Otherwise, the system state remains unchanged.
Extension Points	None.

1.4.6 View Product

Brief Description	This use case describes how a Member or Admin views a product showed on the website.
Basic Flow	This use case starts when a user of the website clicks on a link of a product. The system displays detailed information about the product like product's description, rating, comment, etc and some information about its Provider. Also, a list of products of the same Provider and a list of suggested products will be displayed.
Special Requirements	None.
Pre-conditions	None.
Post-conditions	None.
Extension Points	None.

1.4.7 Order Product

Brief Description	After choosing products, Customers confirm that they want to buy them.
Basic Flow	1. Click the “Order” button. 2. Customers fill in required personal information and the shipping address. Enter sale-off code if they have. 3. Choose payment method. 4. If Customer chooses online payment - Customer chooses type of credit card - Customer chooses bank - Enter required information - The system confirm with the Bank If Customer chooses postpaid method - The system save “postpaid method” status 5. Confirm.
Alternative Flows	If information required for payment is not valid, ask the Customer to re-check.
Special Requirements	None.
Pre-conditions	The Customers must be logged onto the system before this use case begins.
Post-conditions	User’s basic information and shipping address will be sent to Tiki’s Transportation Partner. Payment Information will be handled by Administrator.
Extension Points	None.

1.4.8 Delete Account/Store

Brief Description	This use case allows an Administrator to remove a Account/Store from the system.
Basic Flow	This use case starts when an Administrator wishes to remove a Account/Store from the system: 1. The system requests that the Administrator specify the user ID. 2. The Administrator enters the user ID. The system retrieves and display the user's information. 3. The system prompts the Administrator to confirm the deletion of the user. 4. The Administrator verifies the deletion. 5. The system deletes the user from the system.
Alternative Flows	User Not Found: If the user with the specified ID does not exist, the system displays an error message. The Administrator can then enter a different ID or cancel the operation, at which point the use case ends. Delete Cancelled: If the Administrator decides not to delete the user, the removal is cancelled and the use case is re-started at the beginning.
Special Requirements	None.
Pre-conditions	The Administrator must be logged onto the system before this use case begins.
Post-conditions	If the use case was successful, the user and the user's published products are removed from the system. Otherwise, the system state is unchanged.
Extension Points	None.

1.4.9 Edit Products' Information

Brief Description	This use case allows an Provider to edit a Products' Information on the system.
Basic Flow	This use case starts when an Provider wishes to add, alter or delete any product's information: 1. Click on "Edit Information" button. 2. Delete wrong information and add new detail information. 3. Submit the change to the system
Alternative Flows	None.
Special Requirements	None.
Pre-conditions	The Provider must be logged onto the system before this use case begins.
Post-conditions	If the use case was successful, the product's information
Extension Points	None.

1.4.10 Edit Order

Brief Description	This use case describes how a customer edits his/her order on Tiki's website.
Basic Flow	This use case starts when the customer wishes to modify order information: 1. The system displays the order information in editable fields.

	2. If the customer selects “**Delete**”, the **Delete Order** subflow is executed. Otherwise, the **Update Order Information** is executed. **Delete Order** 1. The system asks the customer to confirm the deletion of the order. 2. The customer verifies the deletion. 3. The system removes the order from the system. **Update Order Information** 1. The customer makes the desired changes to the order information, including any of the information specified in the Order Product use case. 2. The customer selects “Update”. 3. The system updates the order record in the database.
Alternative Flows	None
Special Requirements	None.
Pre-conditions	The customer must be logged onto the system and has an order which hasn’t delivered or made payment yet.
Post-conditions	If the use case is successful, the order is updated or removed. Otherwise, it is unchanged.
Extension Points	None.

1.4.11 Review Product

Brief Description	This use case allows a Customer to submit product’ reviews on the system.
Basic Flow	This use case starts when a Customer wishes to send any

	feedback to the system: 1. Click on each product. 2. Click the “Write your review” button. 3. Choose the product rating by level 5*. 4. Write title of customer’s review. 5. Enter the review. 6. Add product’s photo. (optional) 7. Submit the review. 8. The review will be sent to the Admins 9. Display customer’s review based on the number of likes per comment.
Alternative Flows	Cannot submit: If the review’s content contains less than 50 characters or send more than 5 product’s photo, the review cannot be submitted.
Special Requirements	Review standard: ● Reviews should be written in Vietnamese with accents. ● The review is not intended to promote the purchase of products between individuals, not to call for the purchase of products from other websites, or was posted on other websites, do not deliberately smear products for the reason personal.
Pre-conditions	The Provider must be logged onto the system before this use case begins.
Post-conditions	If the use case was successful, reviews after being censored will be displayed below the product so that other customers have more information.
Extension Points	None.

1.4.12 Update New Products

Brief Description	This use case allows a Provider to create and complete product' reviews on the system.
Basic Flow	This use case starts when a Provider wishes to update all information of new product: 1. Click on "Create new product" button 2. Write all information about new product such as: Title, Price, Brand, Technical specifications, Payment method and several photo of products. 3. Submit to show new product on the system
Alternative Flows	**Missing Required Information:** If any of the above fields are not filled in or selected the system displays an error message. The User can continue making changes to the form or cancel the publish, at which point the use case ends.
Special Requirements	None.
Pre-conditions	The Provider must be logged onto the system before this use case begins.
Post-conditions	If the use case was successful, the new product will be displayed on the system. Otherwise, the system state is unchanged.
Extension Points	None.

1.4.13 Handle Order

Brief Description	After a Customer has ordered a list of products, Administrators will have to update the information about that order like “packaged”, “shipped”, ... and the estimated arrival date.
Basic Flow	1. The system requires the Admin to enter the order’s ID. 2. The Admin choose the current state of the Customer’s order. 3. The system updates new state and information.
Alternative Flows	**Order ID not found** If the system cannot find the Order ID, it displays an error message. The Administrator can then enter a different ID or cancel the operation, at which point the use case ends.
Special Requirements	None.
Pre-conditions	The Administrator must be logged onto the system before this use case begins.
Post-conditions	None.
Extension Points	None.

1.4.14 Ask Questions About Products

Brief Description	Customers ask questions about the products that they are looking for. The questions will be handled by Providers.
Basic Flow	1. Click on the product 2. Choose the Q&A box 3. Fill in compulsory information (the question content) 4. Send the question

Alternative Flows	None
Special Requirements	None
Pre-conditions	The Customer must login his/her account before ask and send the question.
Post-conditions	The Providers are ensured that they receive the whole question without information loss or changes.
Extension Points	None

1.4.15 Answer Customers’ Questions

Brief Description	Providers take full responsibility to answer the questions from Customers.
Basic Flow	1.Click on the list of question. 2.Answer each question about different products. Click link product to see detail information in order to answer exactly. 3. Send the answer. 4. The answer will be displayed on Q&A box.
Alternative Flows	None.
Special Requirements	None.
Pre-conditions	The Provider must login his/her account before send the question.
Post-conditions	The Customer are ensured that they receive the whole answer without information loss or changes.
Extension Points	None.

II. USE-CASE ANALYSIS

2.1 Architecture Analysis

2.1.1 High-level organisation of the model

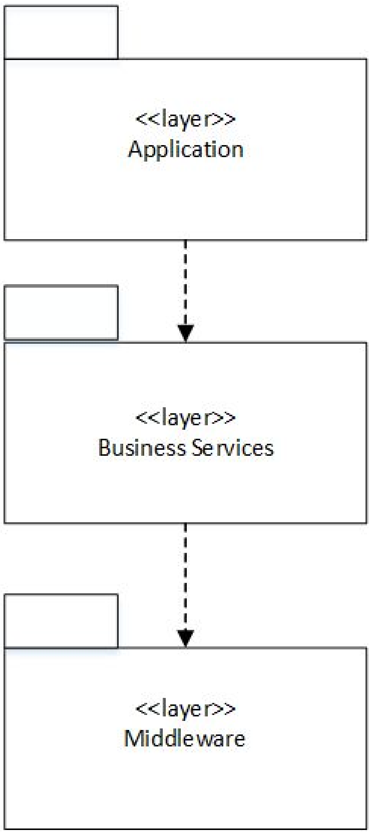


Figure 2.1: Layer Approach

The above figure describes the high-level organisation of the software system. The system consists of three layers:

- The Application layer contains the design elements that are specific to each use case of the system.

• The Business Services layer encapsulates some key abstractions and services common to all use cases. It is accessible from the Application layer.
• The Middleware layer offers services to enable data communication and management on distributed systems.

2.1.2 Key abstractions

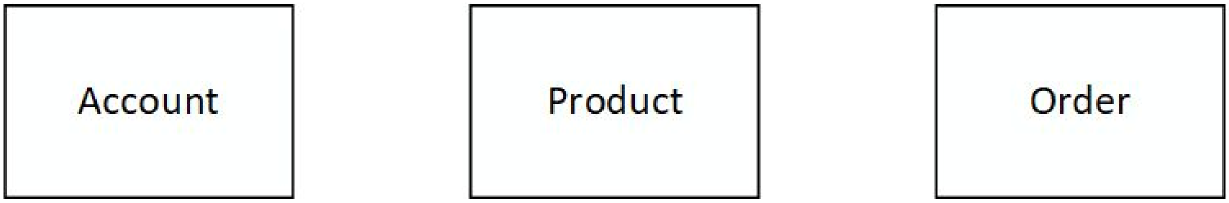


Figure 2.2. Key abstractions used in the application

- **Account**

A record about a customer/administrator/provider containing information about his/her name, e-mail address, password, phone number and optional self-introduction. Each account has a unique user ID and a password, which are used to identify the user/administrator and grant them access to secure parts of the system.

- **Product**

Goods or services provided by sellers through Tiki's website. Products are verified by Tiki's for quality and license. Customers can buy products.

- **Order**

List of products that Customers decides to buy. All of ordered products have full Information. Customers can Order product and Edit Order. Administrations can Handle Order.

2.2 Use-case Realisations

2.2.1 Sequence Diagrams

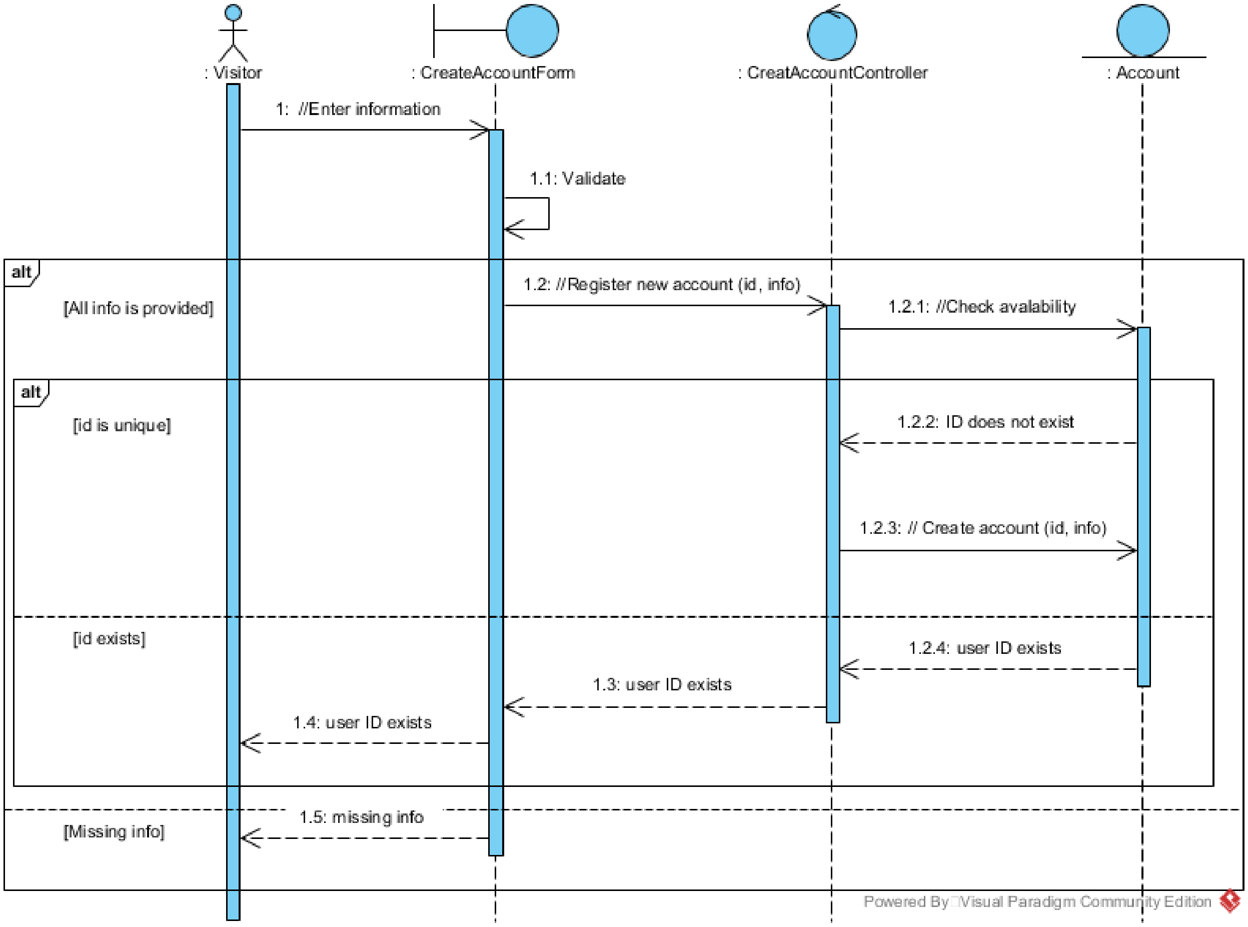


Figure 2.3: Sequence diagram for Create Account use-case

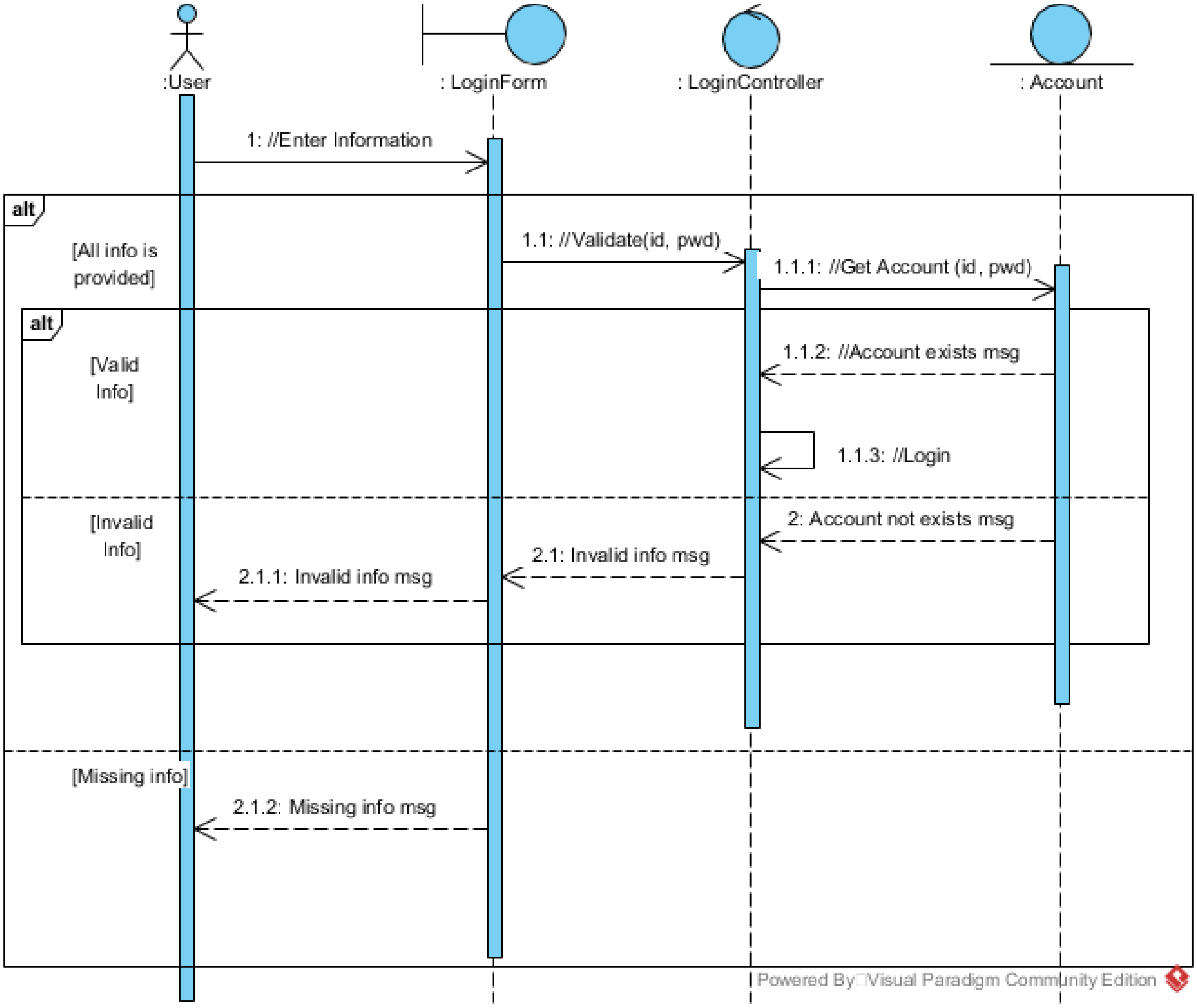


Figure 2.4: Sequence diagram for Login use-case

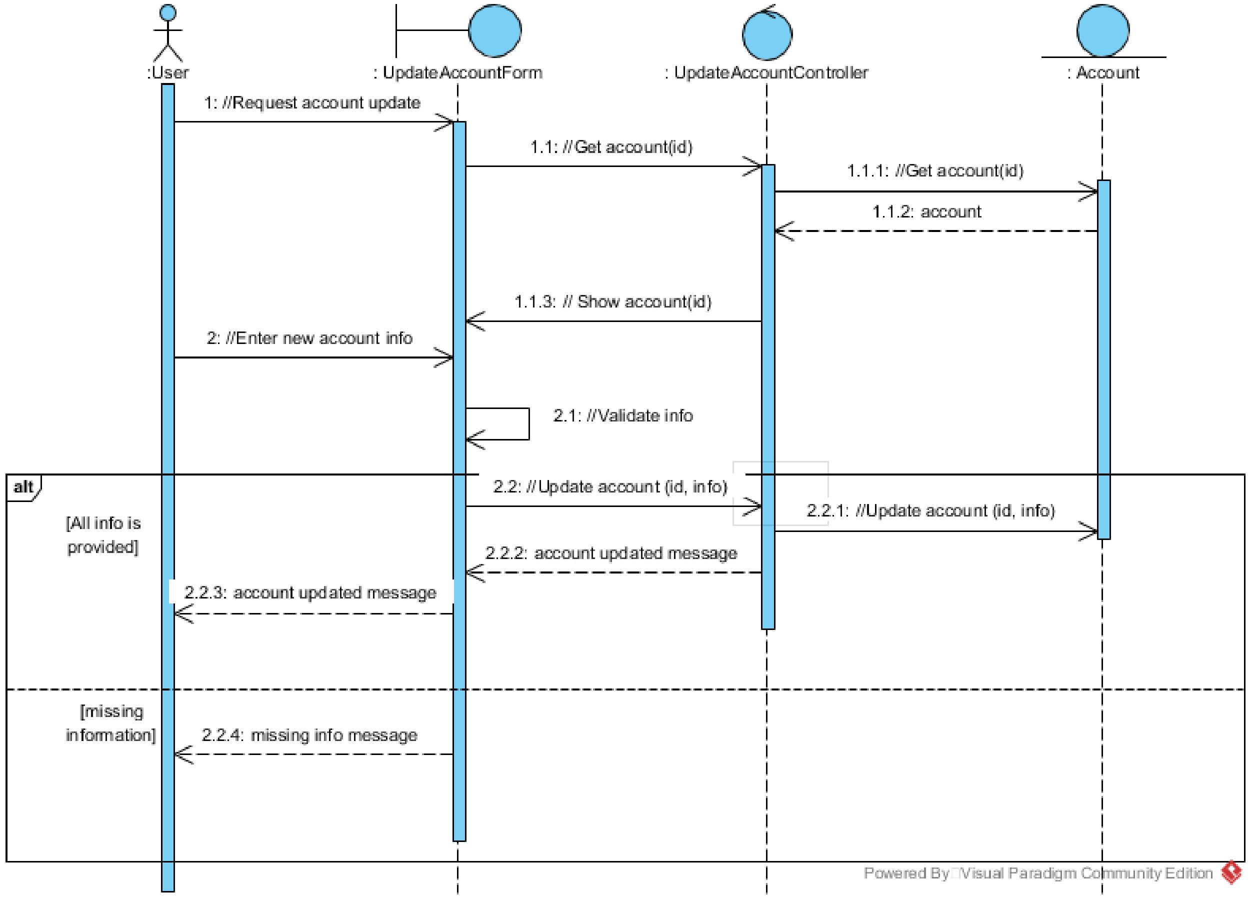

:User
: UpdateAccountForm
: UpdateAccountController
: Account
1: //Request account update
1.1: //Get account(id)
1.1.1: //Get account(id)
1.1.2: account
1.1.3: // Show account(id)
2: //Enter new account info
2.1: //Validate info
alt
[All info is provided]
2.2: //Update account (id, info)
2.2.1: //Update account (id, info)
2.2.2: account updated message
2.2.3: account updated message
[missing information]
2.2.4: missing info message
Powered By Visual Paradigm Community Edition

Figure 2.5: Sequence diagram for Update Profile use-case

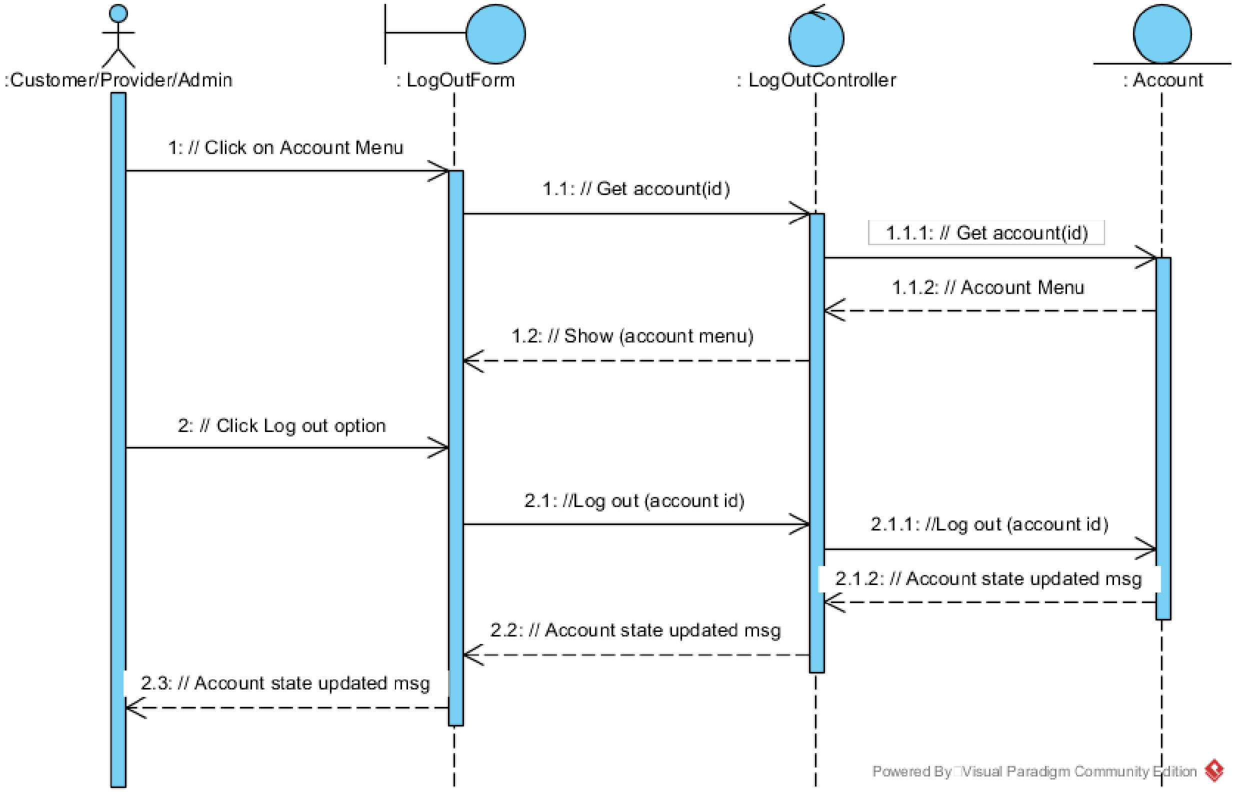

:Customer/Provider/Admin
: LogOutForm
: LogOutController
: Account
1: // Click on Account Menu
1.1: // Get account(id)
1.1.1: // Get account(id)
1.1.2: // Account Menu
1.2: // Show (account menu)
2: // Click Log out option
2.1: //Log out (account id)
2.1.1: //Log out (account id)
2.1.2: // Account state updated msg
2.2: // Account state updated msg
2.3: // Account state updated msg
Powered By Visual Paradigm Community Edition

Figure 2.5: Sequence diagram for Logout use-case

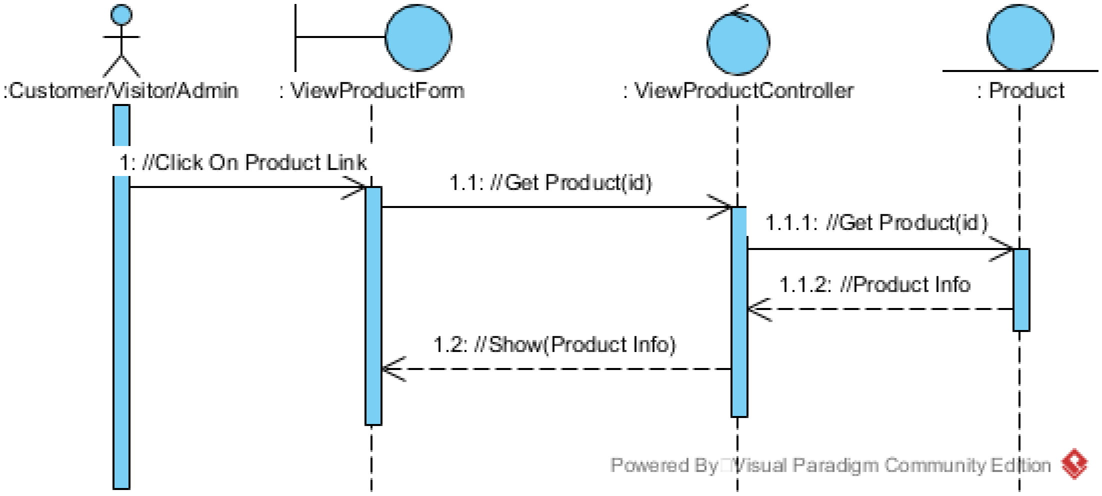


Figure 2.6: Sequence diagram for View Product use-case

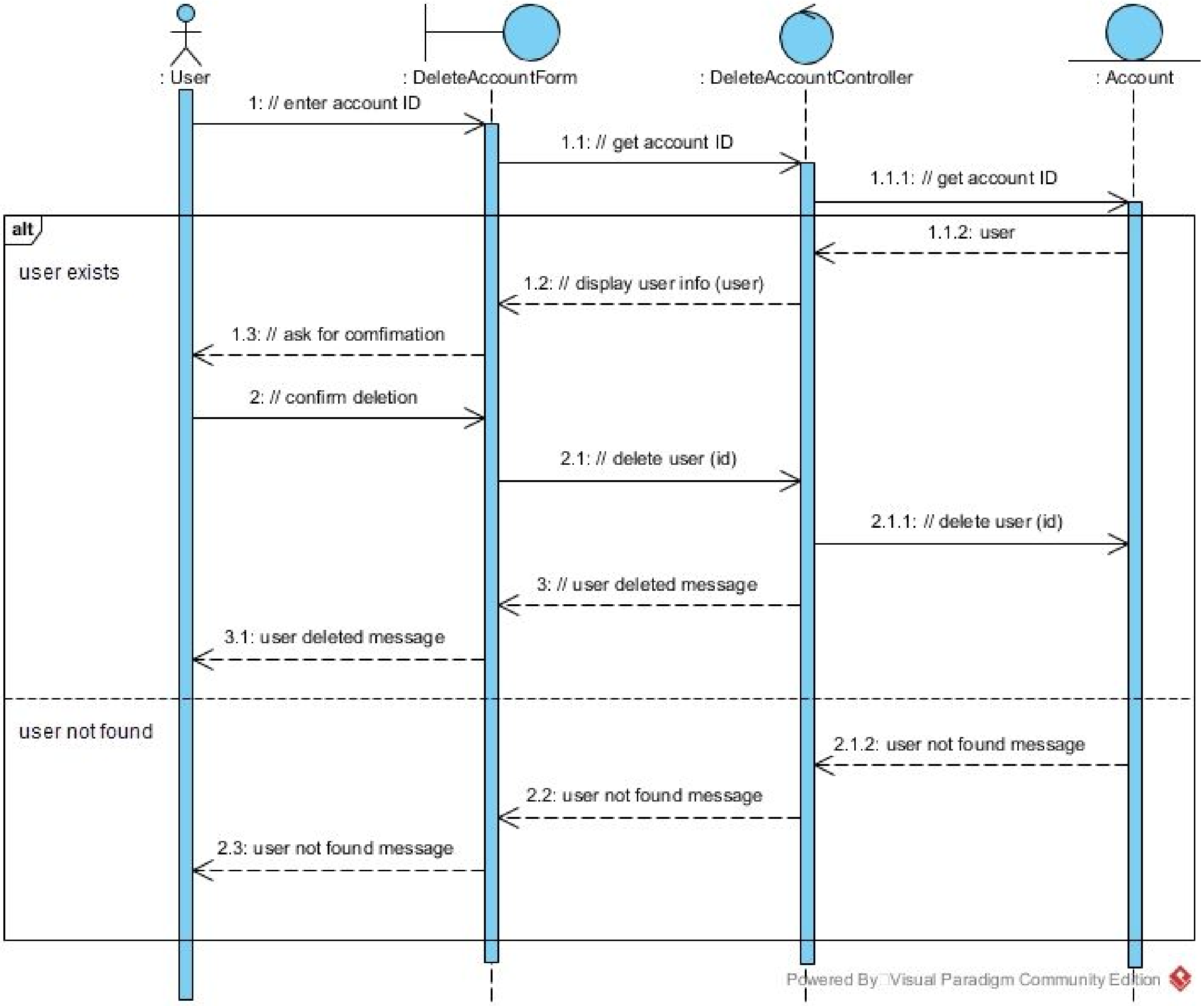

: User
: DeleteAccountForm
: DeleteAccountController
: Account
1: // enter account ID
1.1: // get account ID
1.1.1: // get account ID
alt
user exists
1.1.2: user
1.2: // display user info (user)
1.3: // ask for comfimation
2: // confirm deletion
2.1: // delete user (id)
2.1.1: // delete user (id)
3: // user deleted message
3.1: user deleted message
user not found
2.1.2: user not found message
2.2: user not found message
2.3: user not found message
Powered By Visual Paradigm Community Edition

Figure 2.7: Sequence diagram for Delete Account/Store use-case

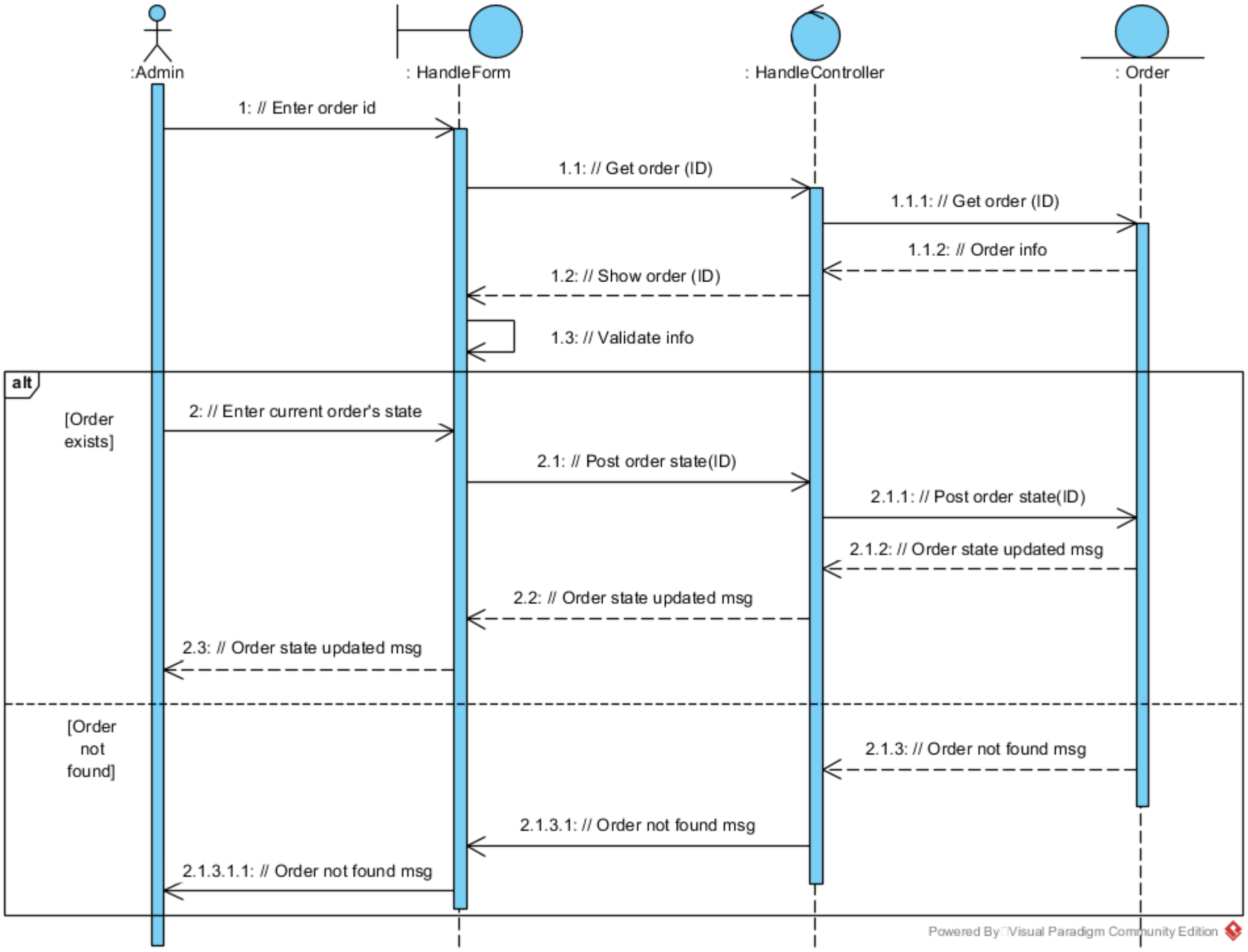

:Admin
: HandleForm
: HandleController
: Order
1: // Enter order id
1.1: // Get order (ID)
1.1.1: // Get order (ID)
1.1.2: // Order info
1.2: // Show order (ID)
1.3: // Validate info
alt
[Order exists]
2: // Enter current order's state
2.1: // Post order state(ID)
2.1.1: // Post order state(ID)
2.1.2: // Order state updated msg
2.2: // Order state updated msg
2.3: // Order state updated msg
[Order not found]
2.1.3: // Order not found msg
2.1.3.1: // Order not found msg
2.1.3.1.1: // Order not found msg
Powered By Visual Paradigm Community Edition

Figure 2.8: Sequence diagram for Handle Order use-case

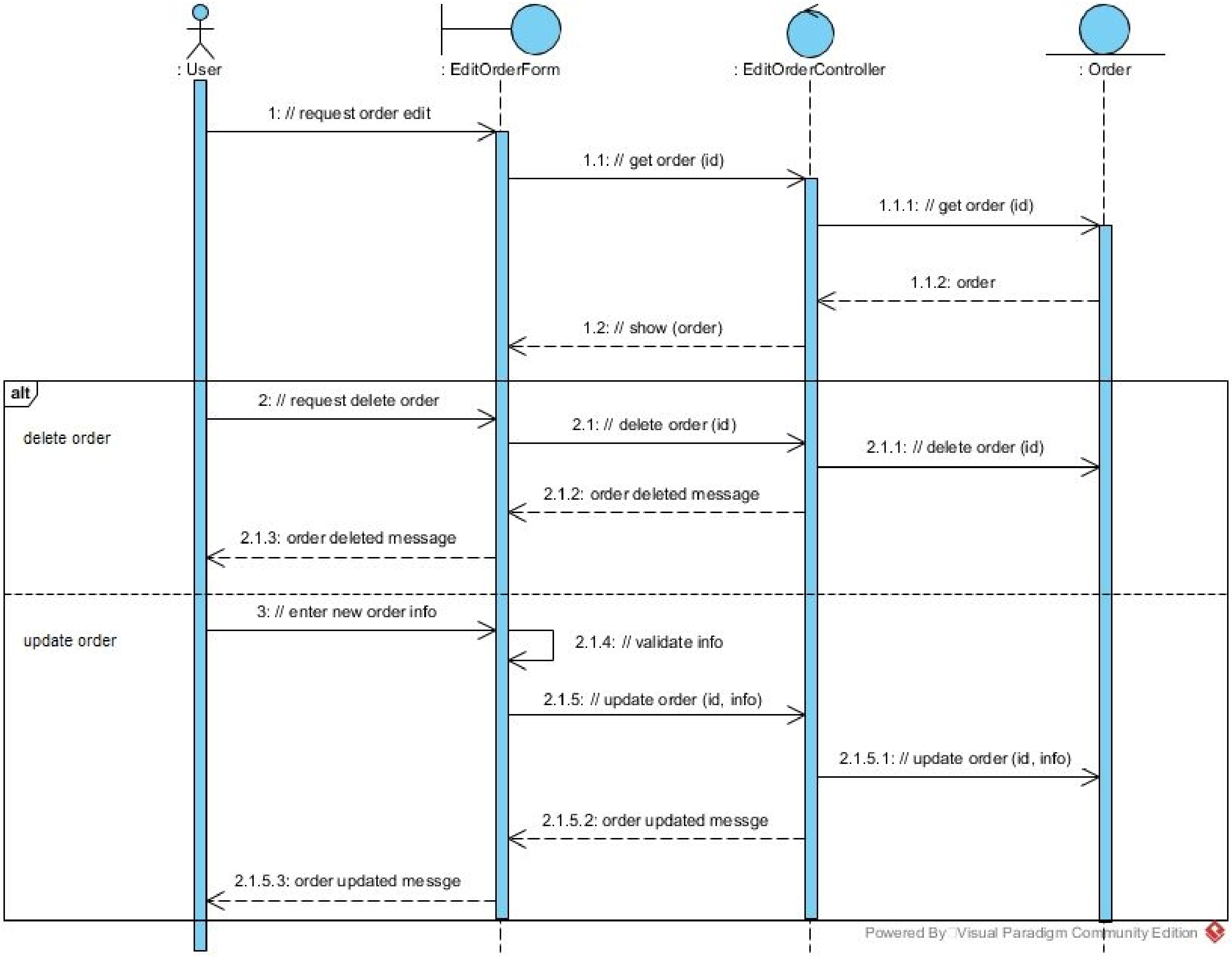

: User
: EditOrderForm
: EditOrderController
: Order
1: // request order edit
1.1: // get order (id)
1.1.1: // get order (id)
1.1.2: order
1.2: // show (order)
alt
delete order
2: // request delete order
2.1: // delete order (id)
2.1.1: // delete order (id)
2.1.2: order deleted message
2.1.3: order deleted message
update order
3: // enter new order info
2.1.4: // validate info
2.1.5: // update order (id, info)
2.1.5.1: // update order (id, info)
2.1.5.2: order updated messge
2.1.5.3: order updated messge
Powered By Visual Paradigm Community Edition

Figure 2.8: Sequence diagram for Edit Order use-case

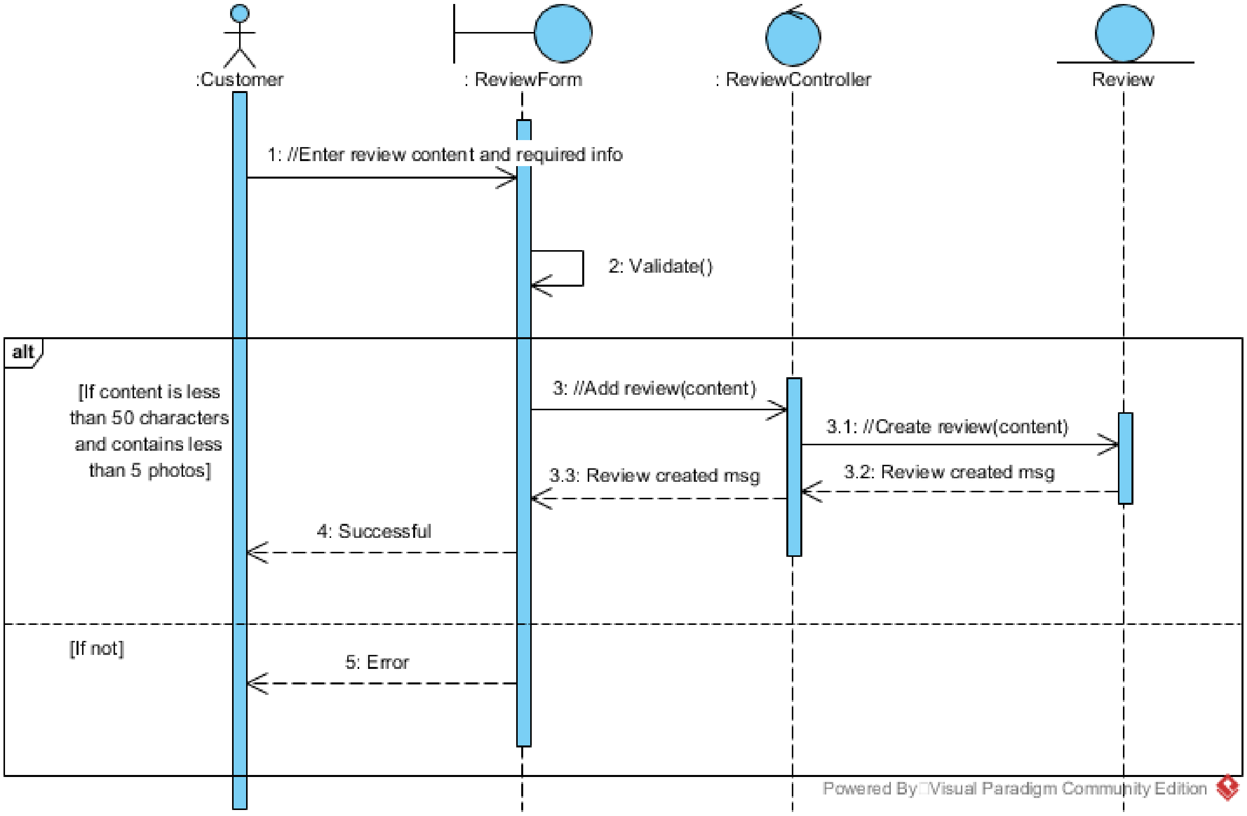

:Customer
: ReviewForm
: ReviewController
Review
1: //Enter review content and required info
2: Validate()
alt
[If content is less than 50 characters and contains less than 5 photos]
3: //Add review(content)
3.1: //Create review(content)
3.2: Review created msg
3.3: Review created msg
4: Successful
[If not]
5: Error
Powered By Visual Paradigm Community Edition

Figure 2.9: Sequence diagram for Review Product use-case

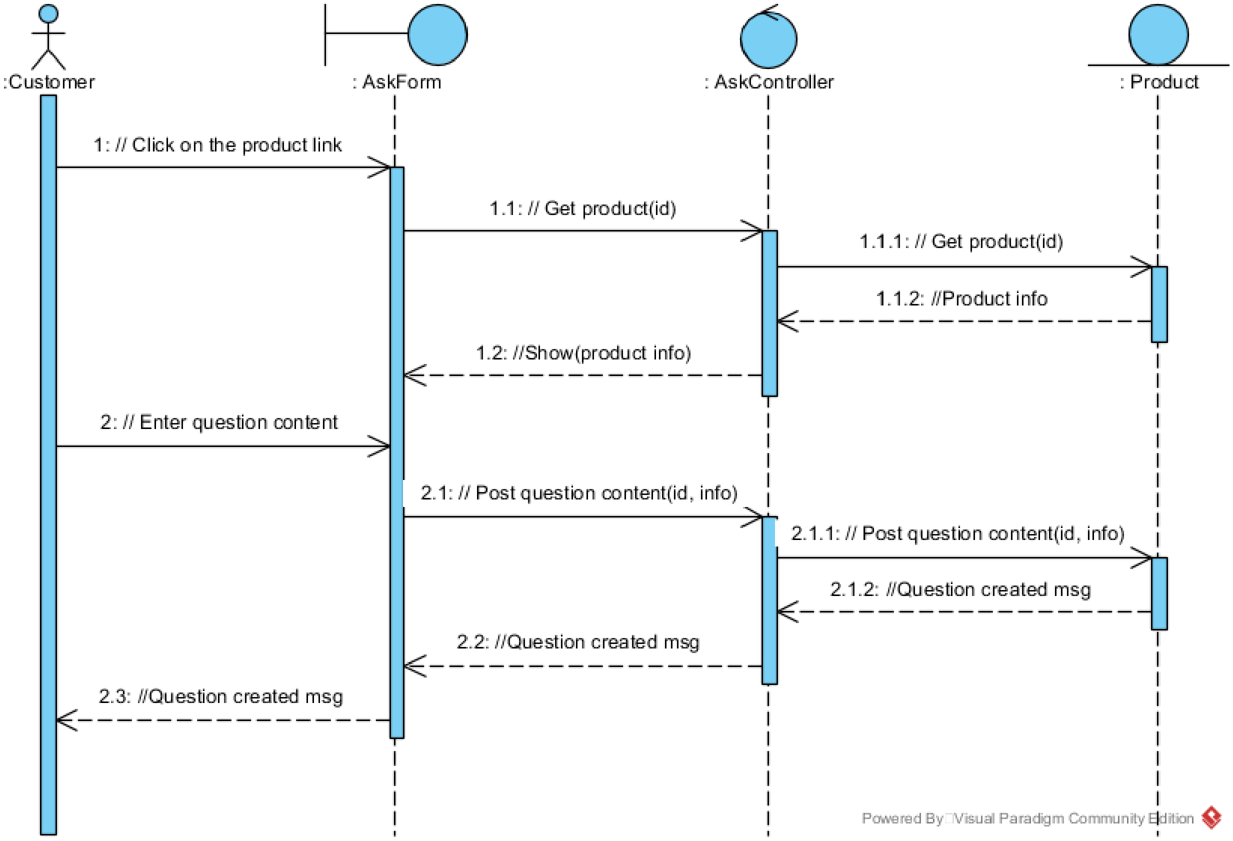
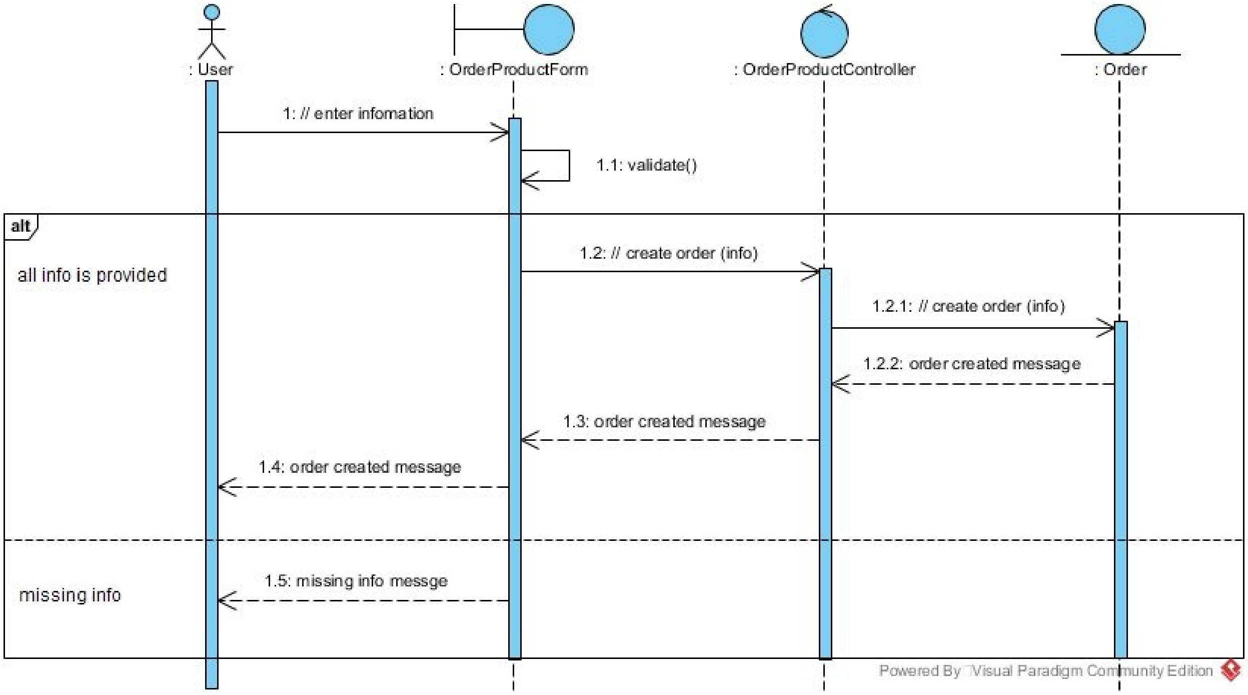

:Customer
: AskForm
: AskController
: Product
1: // Click on the product link
1.1: // Get product(id)
1.1.1: // Get product(id)
1.1.2: //Product info
1.2: //Show(product info)
2: // Enter question content
2.1: // Post question content(id, info)
2.1.1: // Post question content(id, info)
2.1.2: //Question created msg
2.2: //Question created msg
2.3: //Question created msg
Powered By Visual Paradigm Community Edition

Figure 2.10: Sequence diagram for Ask Question about Product use-case



: User
: OrderProductForm
: OrderProductController
: Order
1: // enter infomation
1.1: validate()
alt
all info is provided
1.2: // create order (info)
1.2.1: // create order (info)
1.2.2: order created message
1.3: order created message
1.4: order created message
missing info
1.5: missing info messge
Powered By Visual Paradigm Community Edition

Figure 2.11: Sequence diagram for Order Product use-case

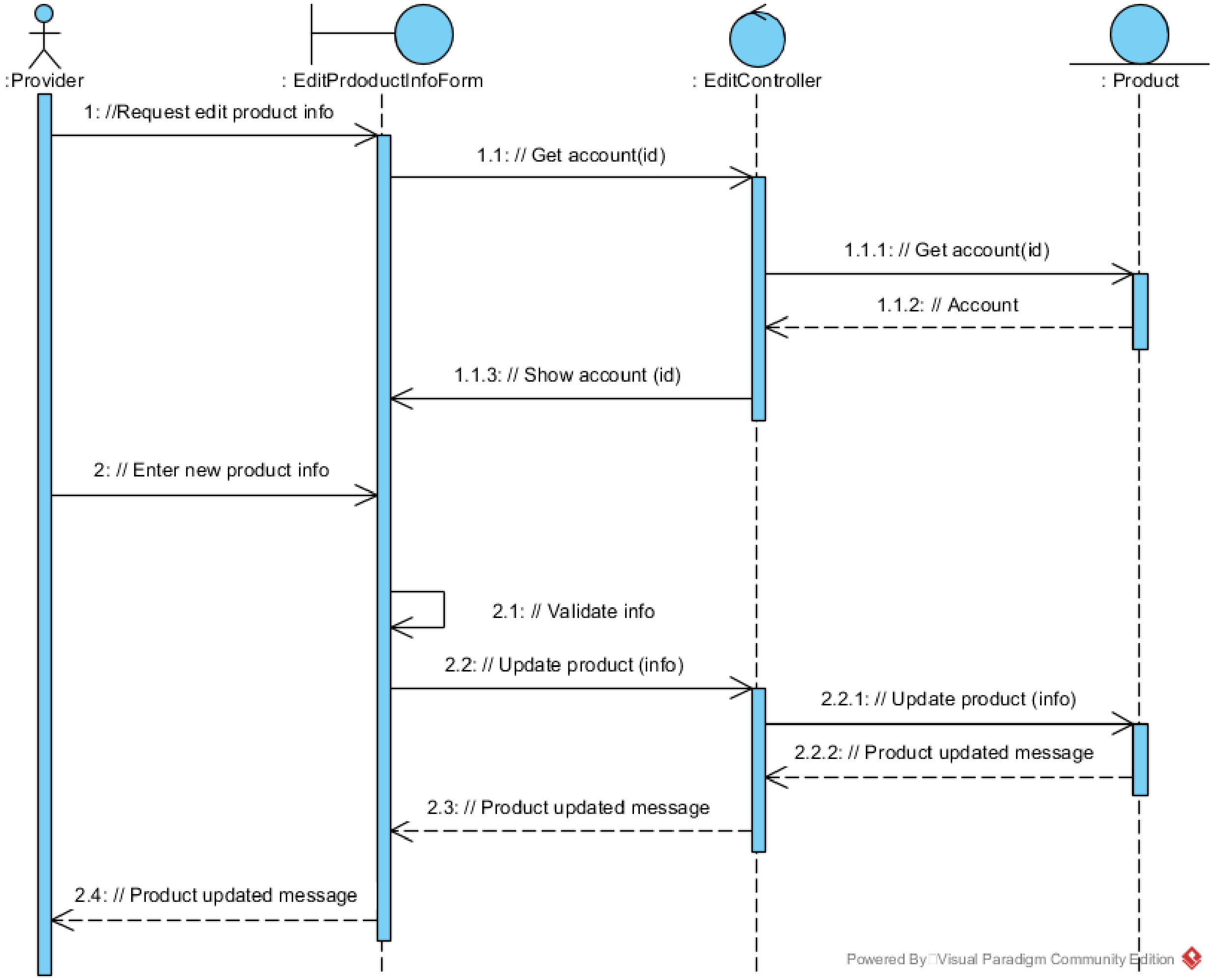

:Provider
: EditPrdoductInfoForm
: EditController
: Product
1: //Request edit product info
1.1: // Get account(id)
1.1.1: // Get account(id)
1.1.2: // Account
1.1.3: // Show account (id)
2: // Enter new product info
2.1: // Validate info
2.2: // Update product (info)
2.2.1: // Update product (info)
2.2.2: // Product updated message
2.3: // Product updated message
2.4: // Product updated message
Powered By Visual Paradigm Community Edition

Figure 2.12: Sequence diagram for Edit Products' Information use-case

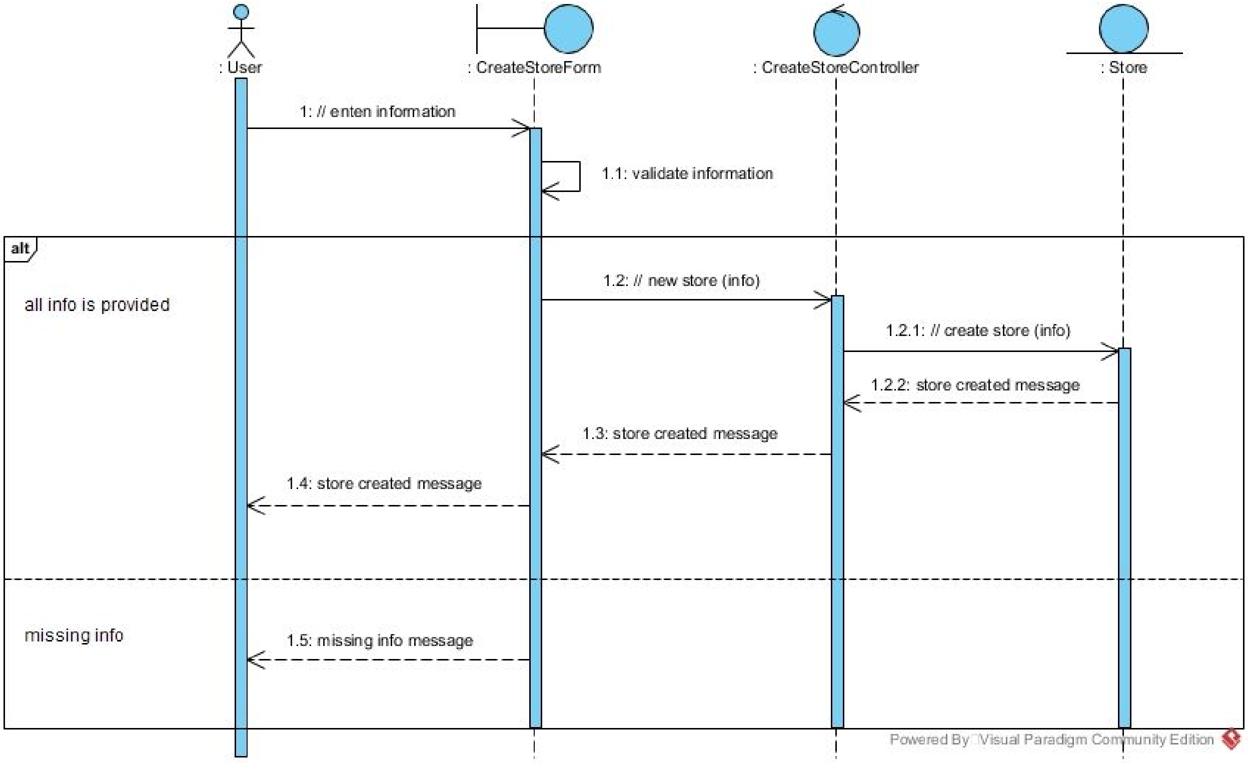

: User
: CreateStoreForm
: CreateStoreController
: Store
1: // enten information
1.1: validate information
alt
all info is provided
1.2: // new store (info)
1.2.1: // create store (info)
1.2.2: store created message
1.3: store created message
1.4: store created message
missing info
1.5: missing info message
Powered By Visual Paradigm Community Edition

Figure 2.13: Sequence diagram for Create Store use-case

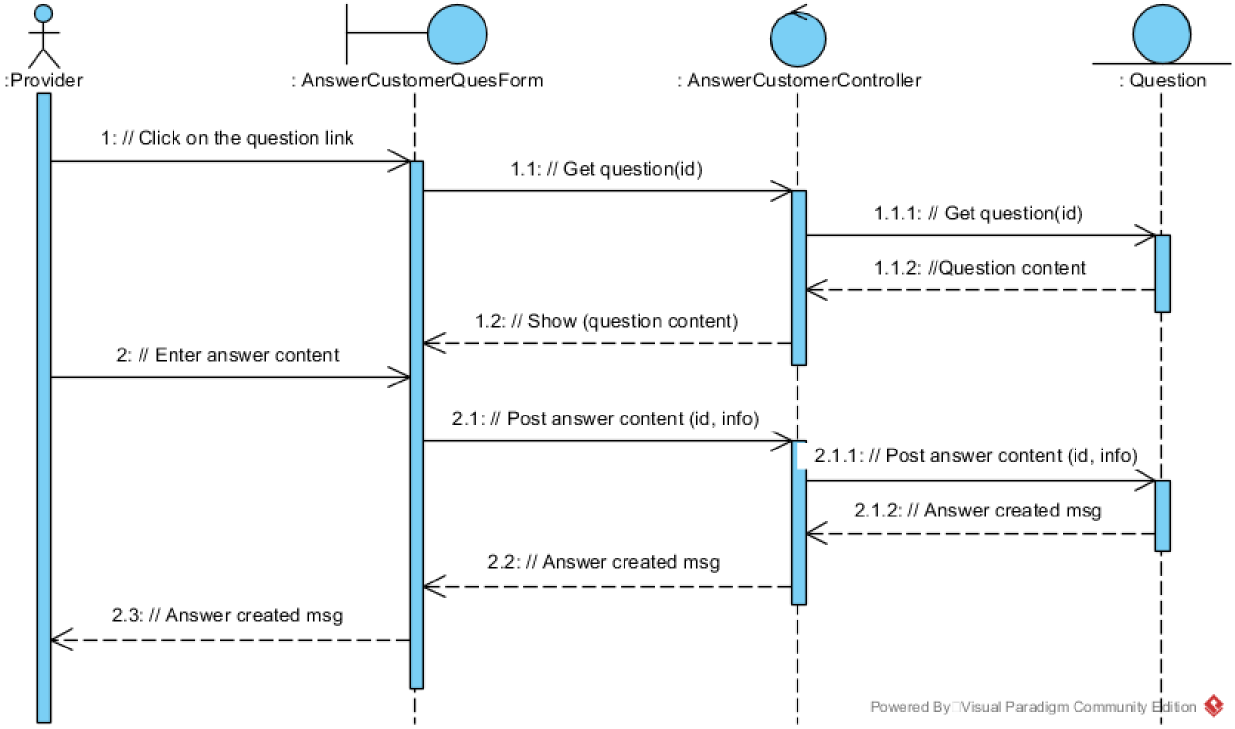

:Provider
: AnswerCustomerQuesForm
: AnswerCustomerController
: Question
1: // Click on the question link
1.1: // Get question(id)
1.1.1: // Get question(id)
1.1.2: //Question content
1.2: // Show (question content)
2: // Enter answer content
2.1: // Post answer content (id, info)
2.1.1: // Post answer content (id, info)
2.1.2: // Answer created msg
2.2: // Answer created msg
2.3: // Answer created msg
Powered By Visual Paradigm Community Edition

Figure 2.14: Sequence diagram for Answer Customers' Questions use-case

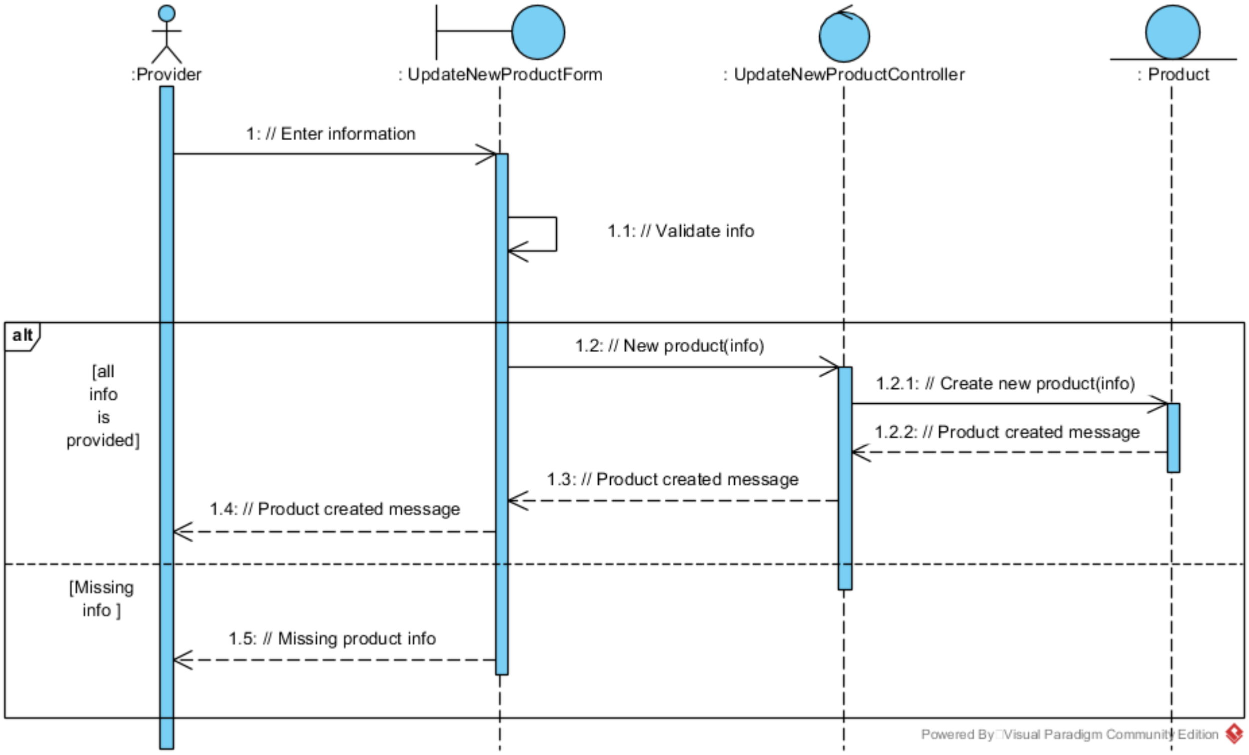

:Provider
: UpdateNewProductForm
: UpdateNewProductController
: Product
1: // Enter information
1.1: // Validate info
alt
[all info is provided]
1.2: // New product(info)
1.2.1: // Create new product(info)
1.2.2: // Product created message
1.3: // Product created message
1.4: // Product created message
[Missing info]
1.5: // Missing product info
Powered By Visual Paradigm Community Edition

Figure 2.15: Sequence diagram for Update New Products use-case

2.2.2 Views Of Participating Classes

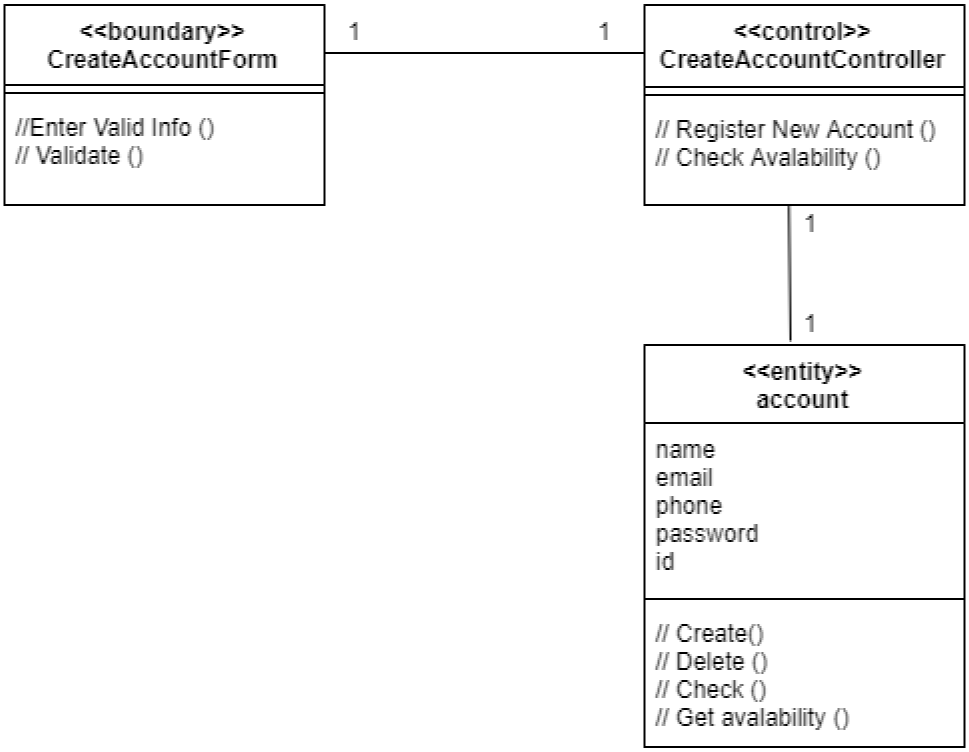


Figure 2.16: VOPC for Create Account use-case

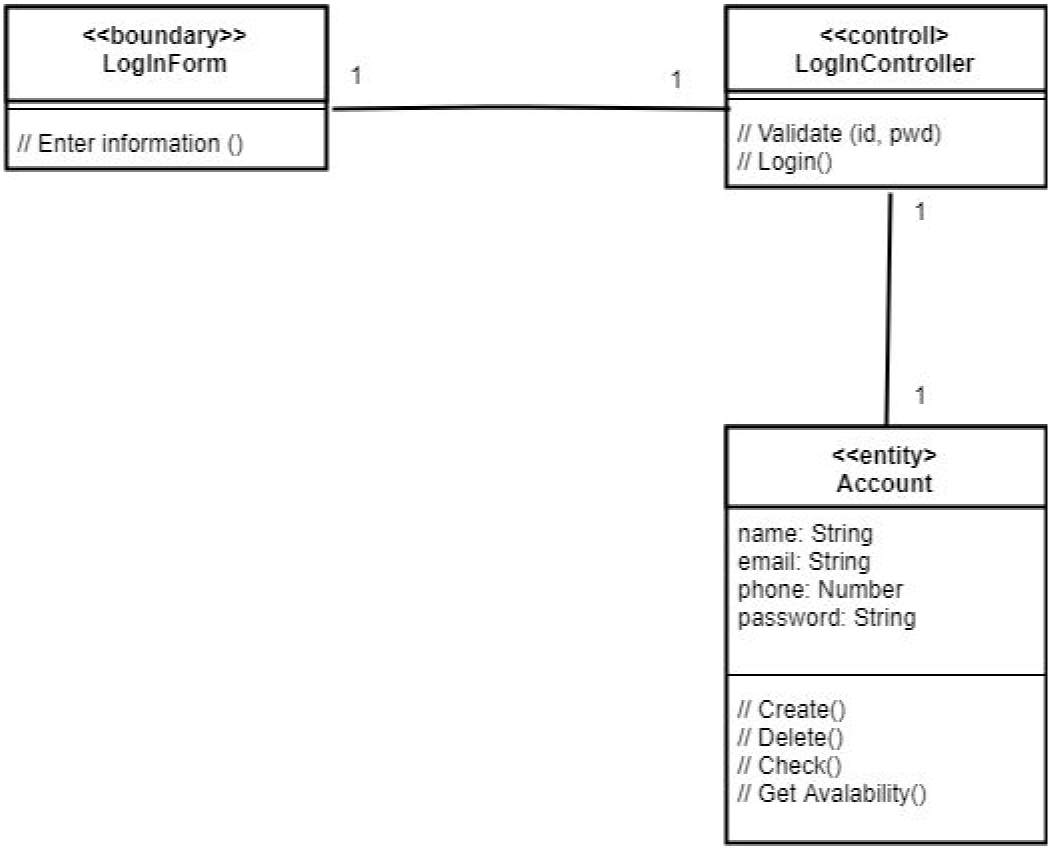


*Figure 2.17: VOPC for Login use-cas*e

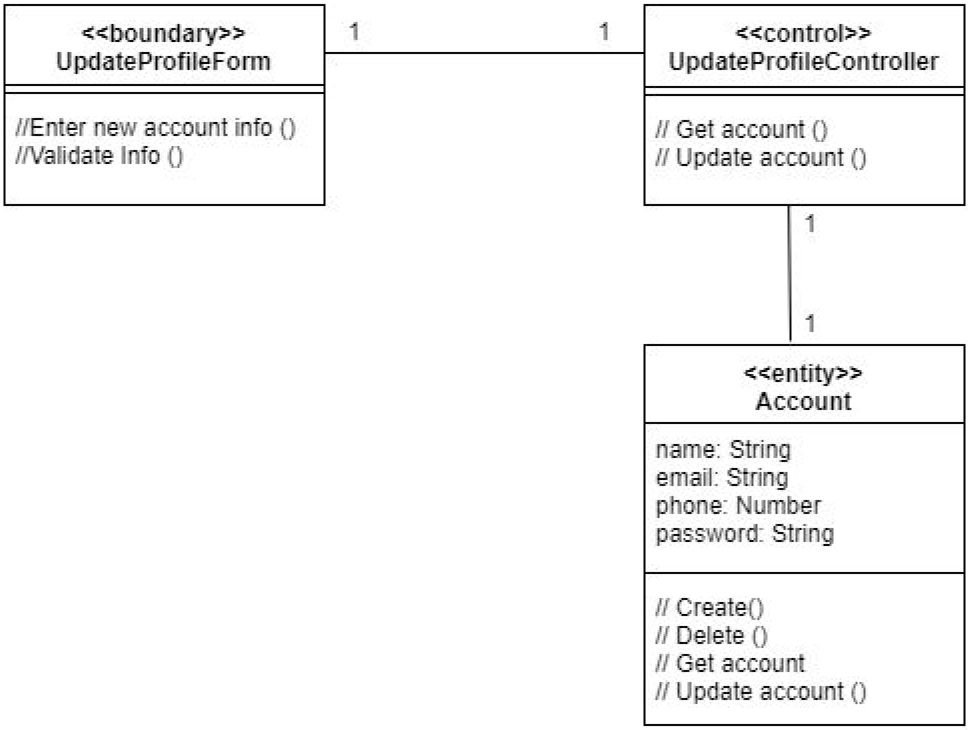


Figure 2.18: VOPC for Update Profile use-case

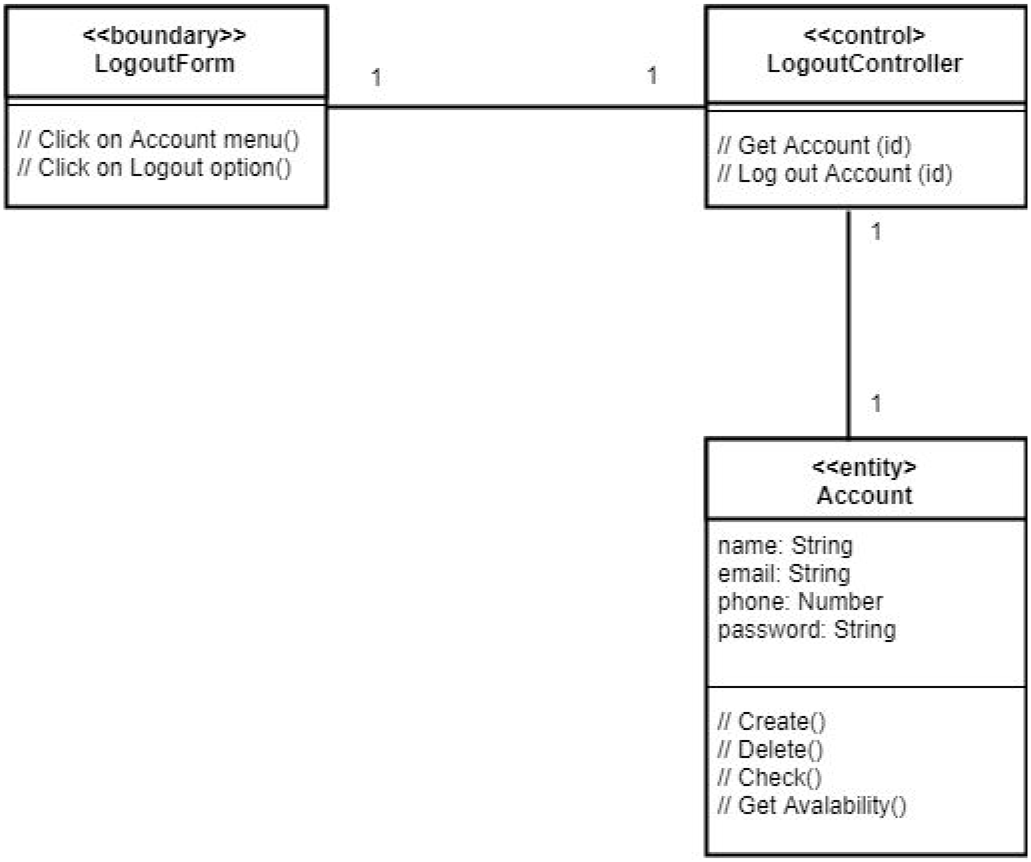


Figure 2.19: VOPC for Logout use-case

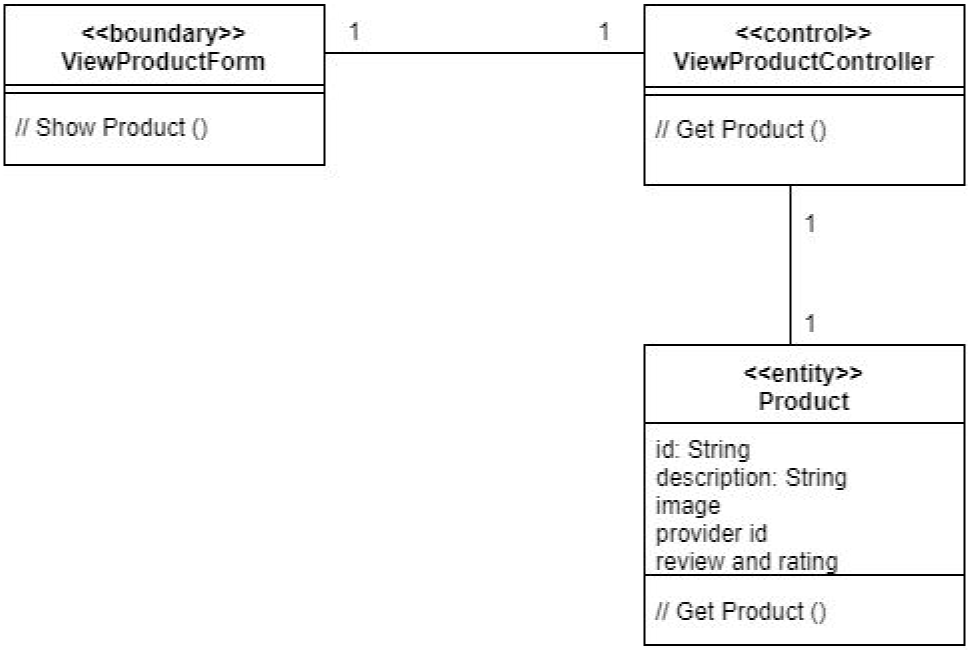


Figure 2.20: VOPC for View Product use-case

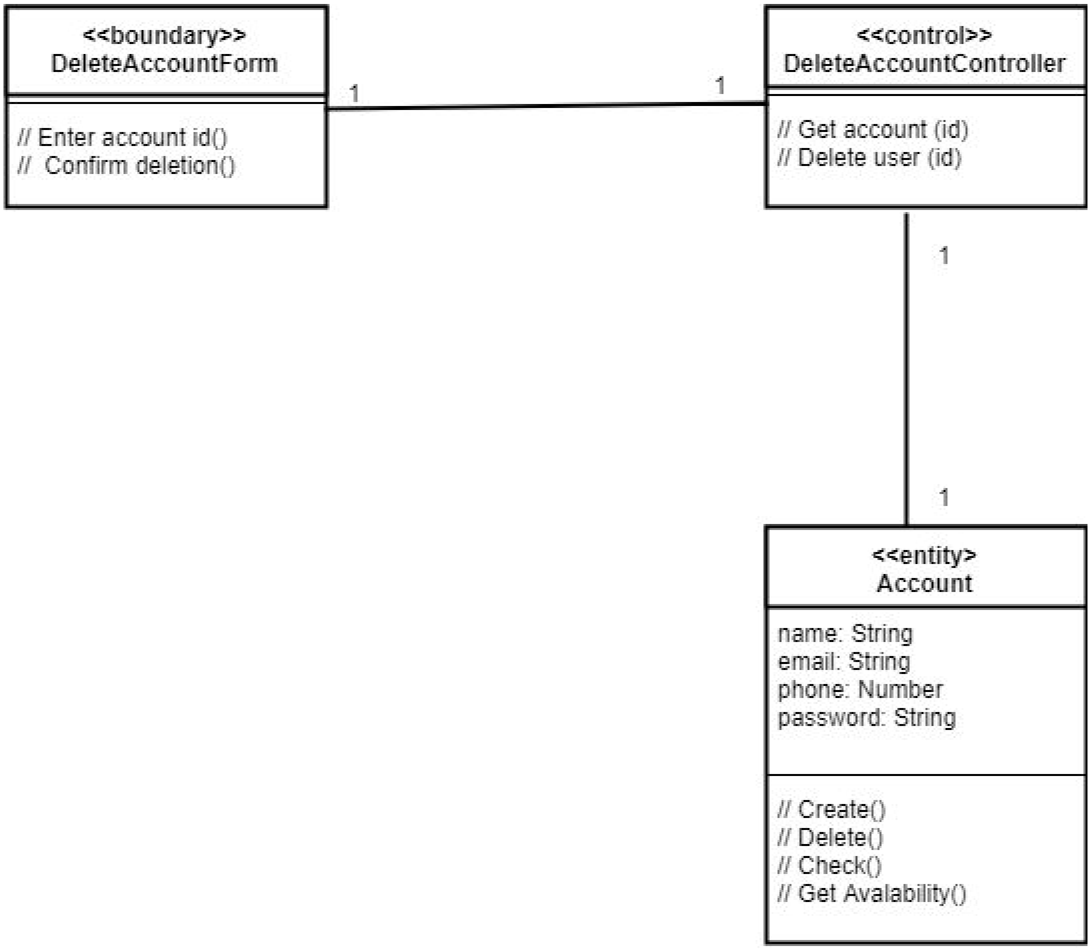


Figure 2.21: VOPC for Delete Account/Store use-case

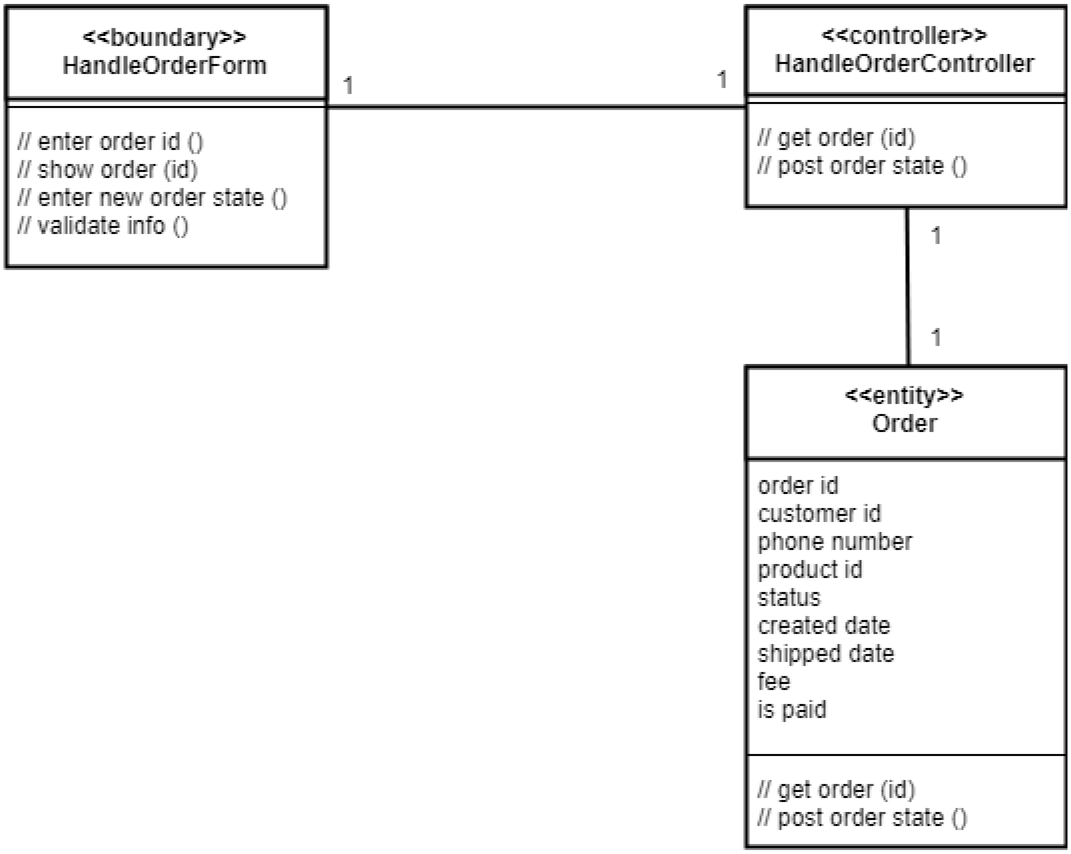


Figure 2.22: VOPC for Handle Order use-case

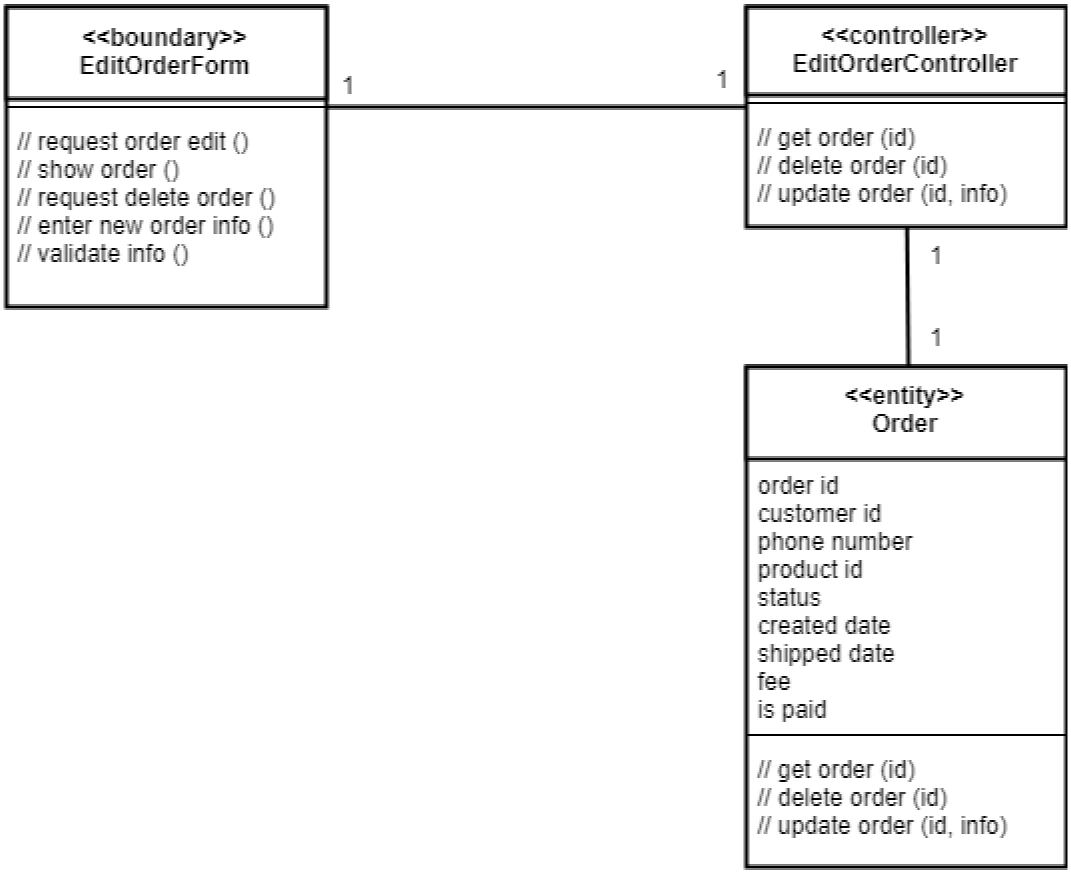


Figure 2.23: VOPC for Edit Order use-case

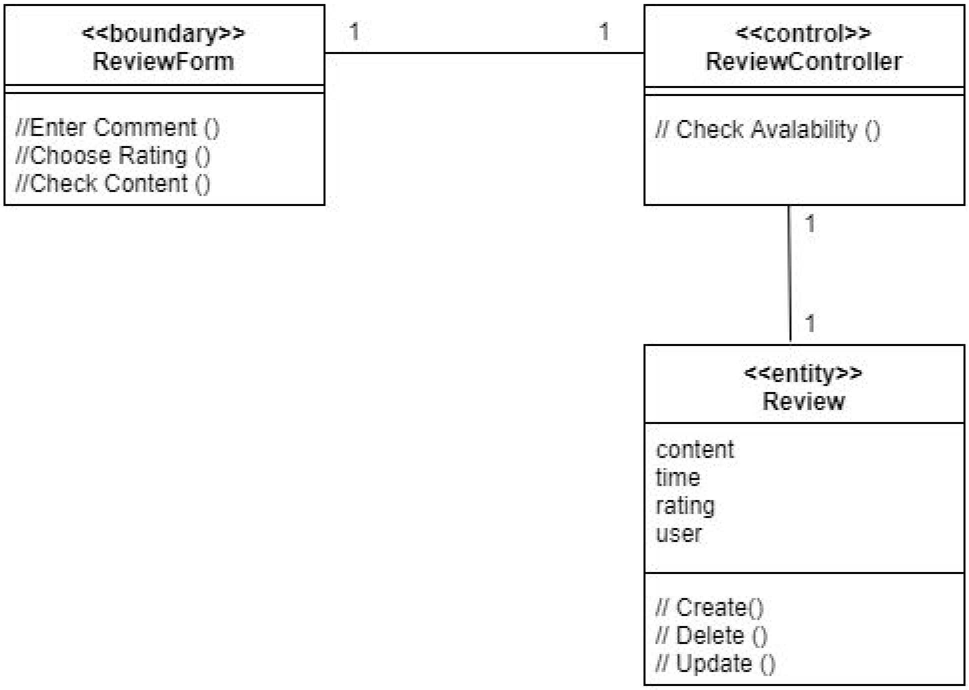


Figure 2.24: VOPC for Review Product use-case

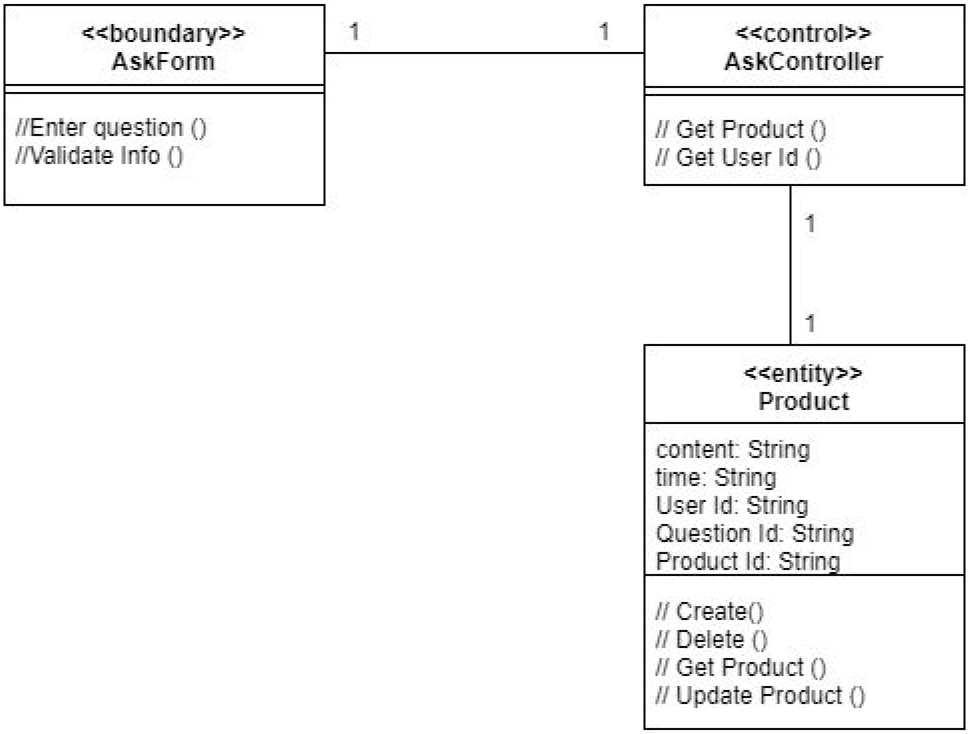


Figure 2.25: VOPC for Ask Question about Product use-case

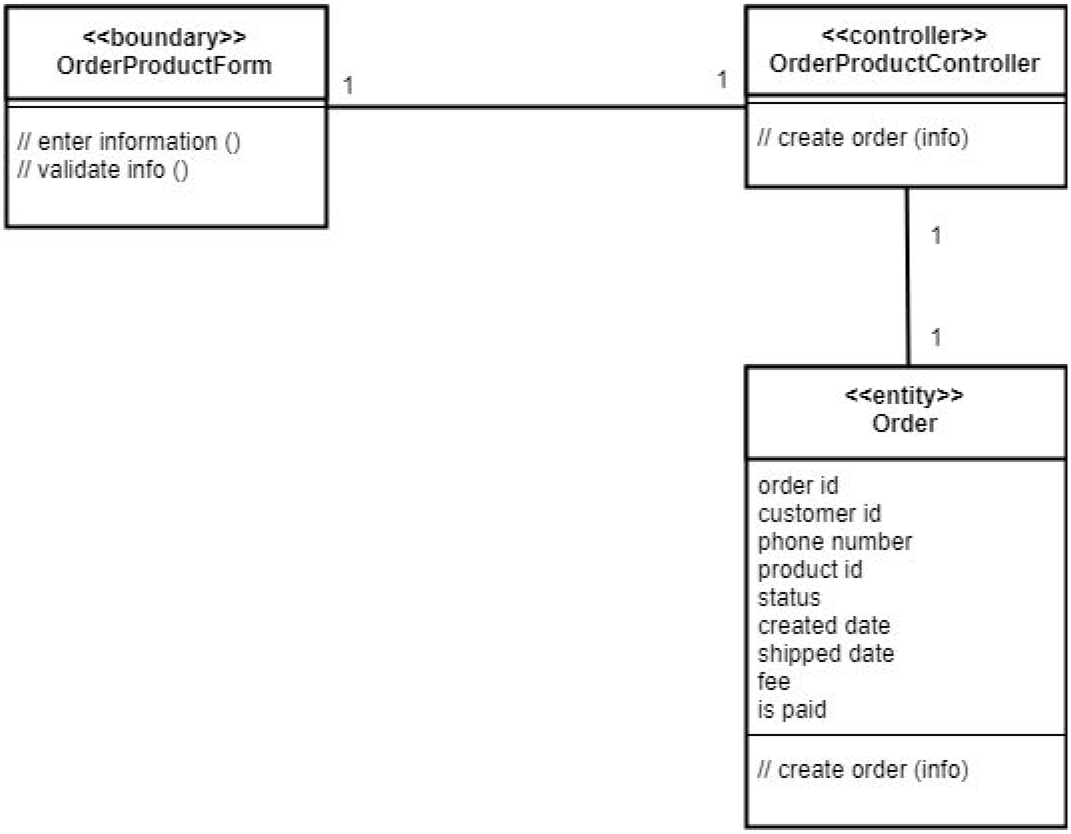


Figure 2.26: VOPC for Order Product use-case

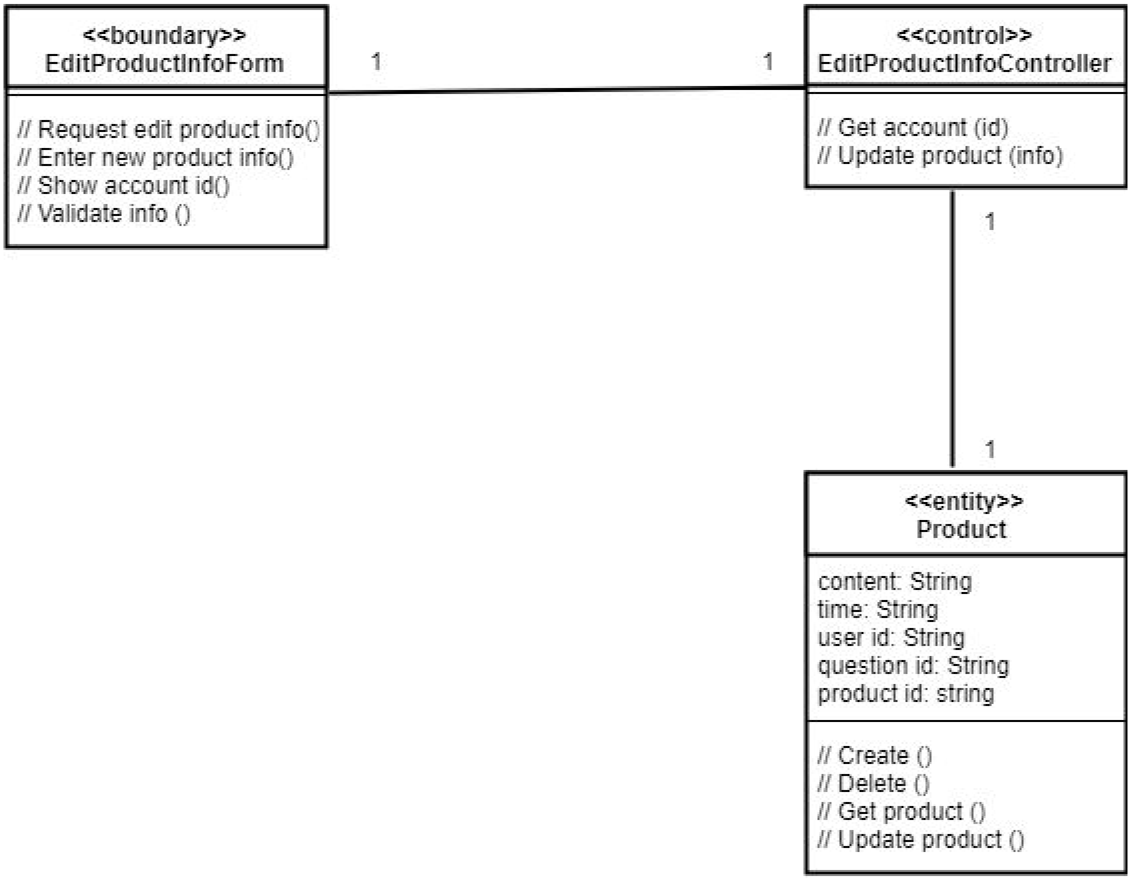


Figure 2.27: VOPC for Edit Products' Information use-case

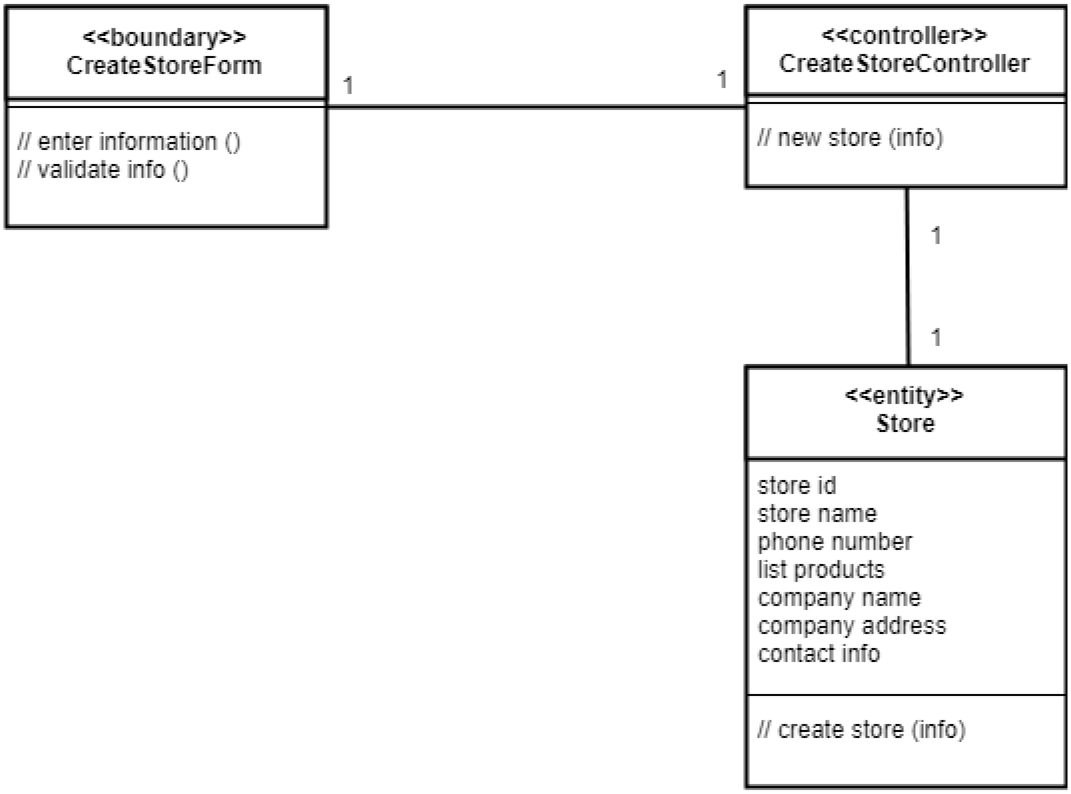


Figure 2.28: VOPC for Create Store use-case

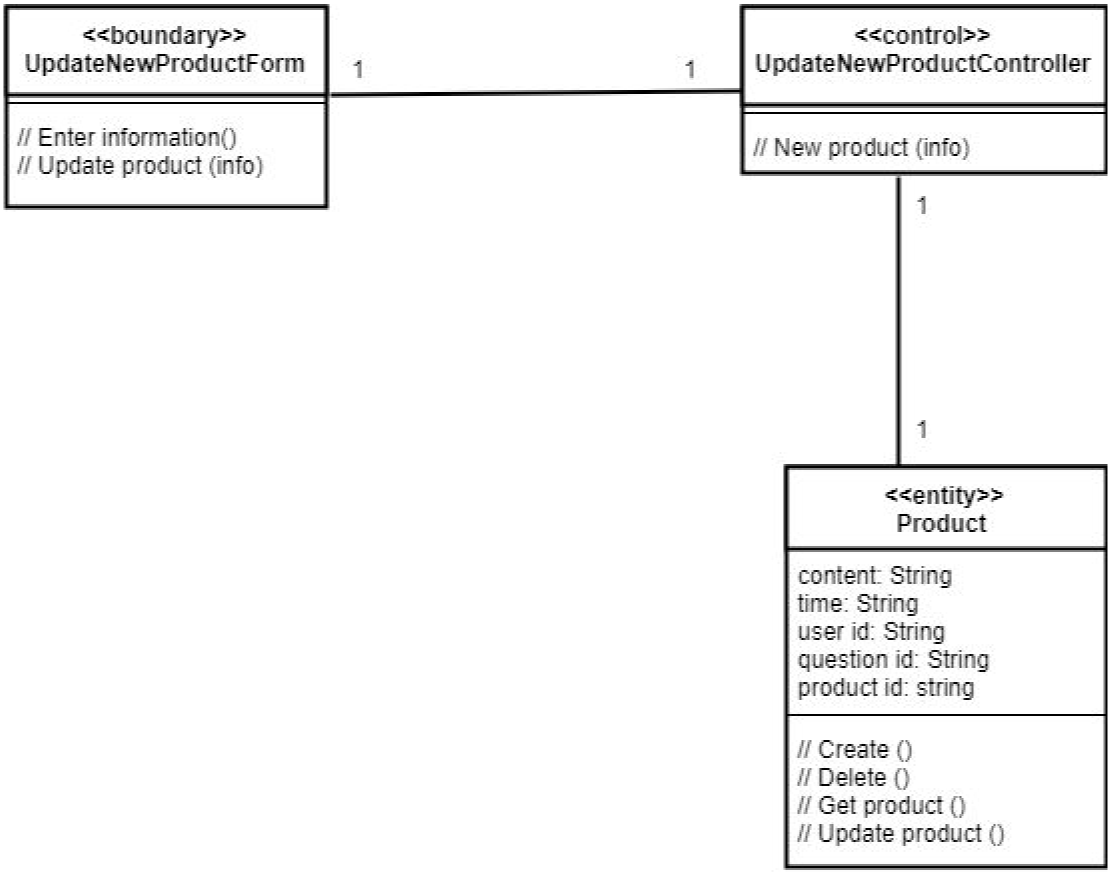


Figure 2.39: VOPC for Update New Products use-case

2.2.3 Describe analysis mechanism

Analysis class	Analysis mechanism
Account	Persistency, Security
Product	
Order	
CreateAccountController	
LogInController	

LogOutController	Distribution
UpdateAccountController	
CreateStoreController	
ViewProductController	
OrderProductController	
DeleteAccountController	
EditController	
EditOrderController	
ReviewController	
UpdateNewProductsController	
HandleController	
AskController	
AnswerCustomerController	

Table 2.1. Analysis-Class-To-Analysis-Mechanism map

Analysis mechanism characteristics security:

- Data granularity: attribute level
- User granularity: three roles – unregistered users, registered users and administrators
- Security rules:
- Only registered users/administrators may log into the system.
- Any user can view product.
- Only logged in users may review products.
- Only logged in users may view and edit their own account profile.
- Only Admin could delete account, delete store and handle order.

- The order could only be altered by its owner.
- The product could only be edited by its provider.

Persistency:

Class	Account	Product	Order
Granularity	100 KB per product	1 to 4 KB per product	1 to 10 MB per product
Volume	Up to 100,000	Up to 1,000	Up to 500,000
Access frequency	Create: 1,000 per day Update: 500 per day Delete: 50 per day	Create: 500 per day Delete: 500 per day	Create: 2,000 per day Read: 10,000 per day Update: 5,000 per day Delete: 1,000 per day

III. Use-case design

3.1 Architectural refinement

3.1.1 Identify design elements

3.1.1.1 Identify classes

Analysis class	**Design element**
Account	Account, Database system
Product	Product, Database system
Order	Order, Database system

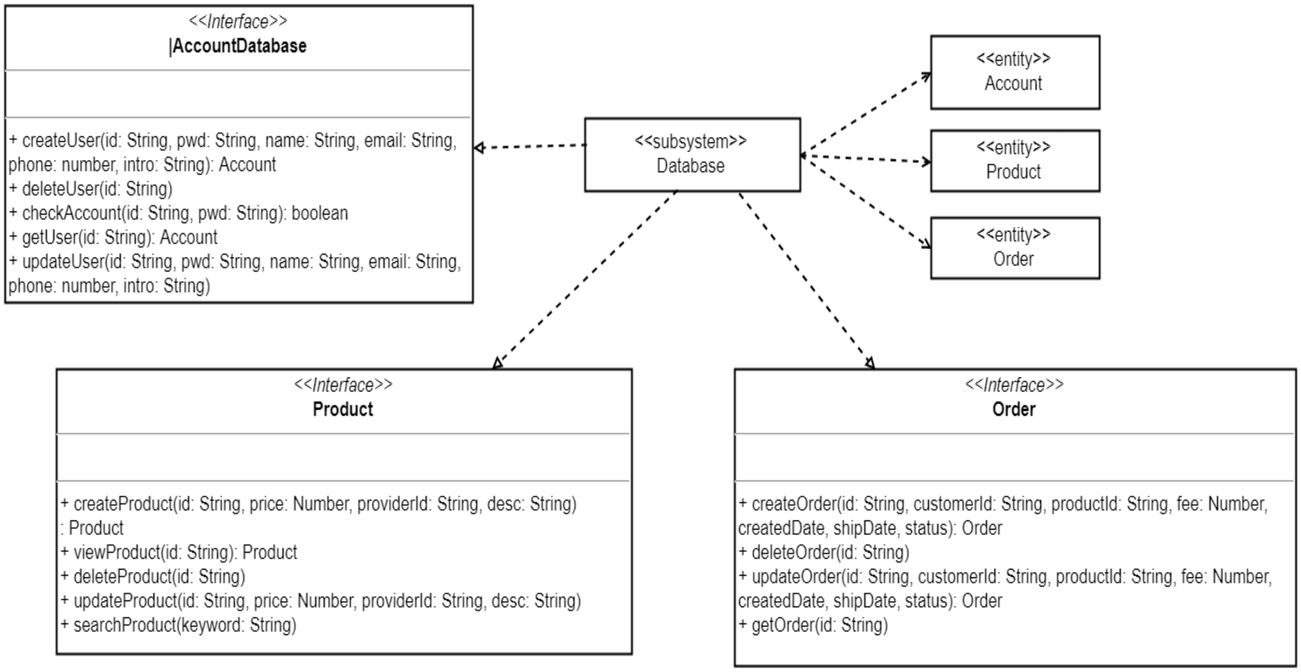
CreateAccountController	Map directly to design classes
LogInController	
LogOutController	
UpdateAccountController	
CreateStoreController	
ViewProductController	
OrderProductController	
DeleteAccountController	
EditController	
EditOrderController	
ReviewController	
UpdateNewProductsController	
HandleController	
AskController	
AnswerCustomerController	
CreateAccountForm	
LogInForm	
LogOutForm	
UpdateAccountForm	
CreateStoreForm	
ViewProductForm	
OrderProductForm	

DeleteAccountForm	
EditForm	
EditOrderForm	
ReviewForm	
UpdateNewProductsForm	
HandleForm	
AskFormr	
AnswerCustomerForm	

Table 3.1. Analysis-Class-To-Design-Element map

3.1.1.2 Identify subsystems and interfaces

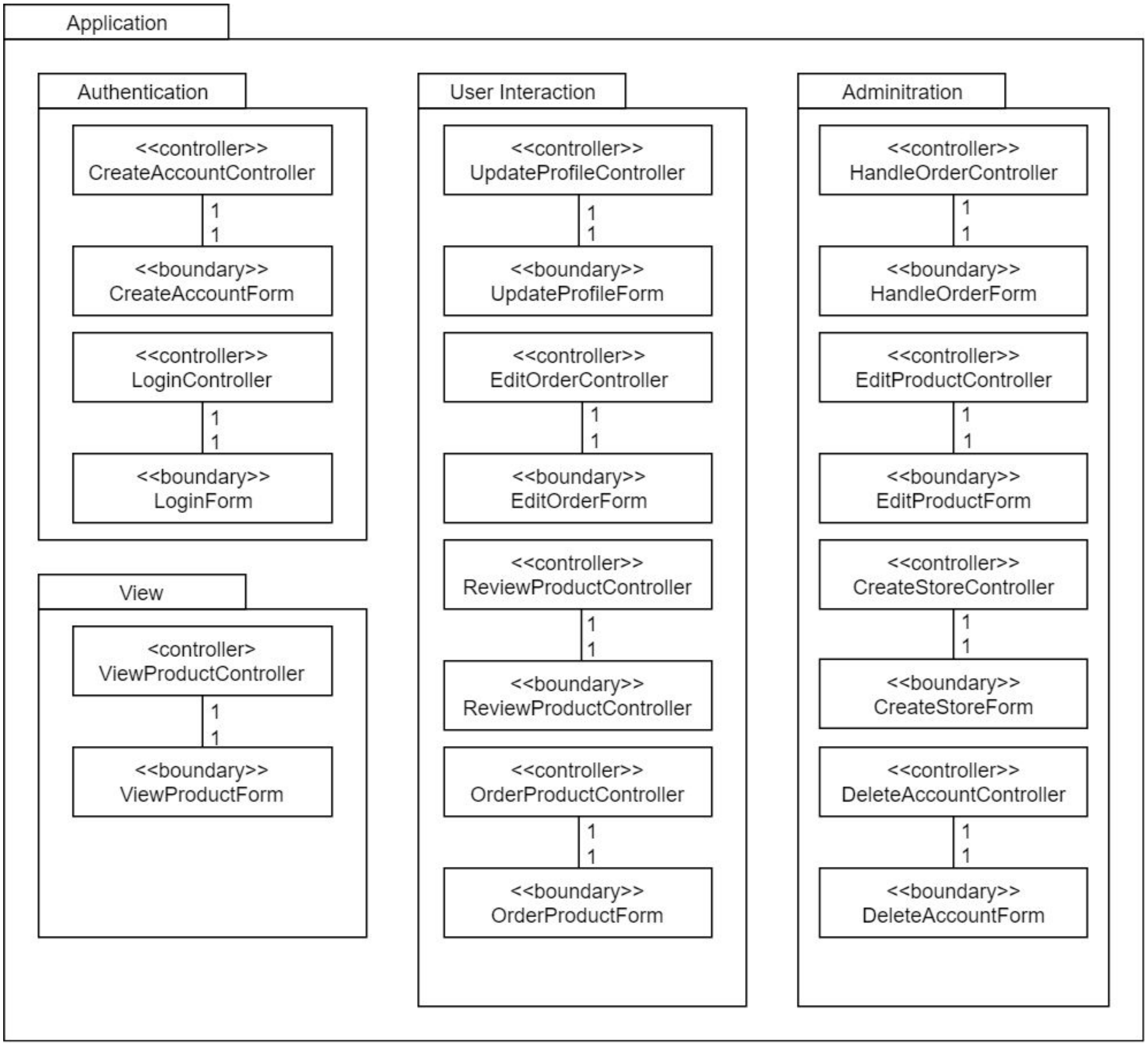
The Database subsystem provides support for relational databases written in the SQL language. The subsystem is designed as follows:


<<Interface>>
|AccountDatabase
+ createUser(id: String, pwd: String, name: String, email: String, phone: number, intro: String): Account
+ deleteUser(id: String)
+ checkAccount(id: String, pwd: String): boolean
+ getUser(id: String): Account
+ updateUser(id: String, pwd: String, name: String, email: String, phone: number, intro: String)
<<subsystem>>
Database
<<entity>>
Account
<<entity>>
Product
<<entity>>
Order
<<Interface>>
Product
+ createProduct(id: String, price: Number, providerId: String, desc: String) : Product
+ viewProduct(id: String): Product
+ deleteProduct(id: String)
+ updateProduct(id: String, price: Number, providerId: String, desc: String)
+ searchProduct(keyword: String)
<<Interface>>
Order
+ createOrder(id: String, customerId: String, productId: String, fee: Number, createdDate, shipDate, status): Order
+ deleteOrder(id: String)
+ updateOrder(id: String, customerId: String, productId: String, fee: Number, createdDate, shipDate, status): Order
+ getOrder(id: String)

3.1.1.3 Identify packages

Each layer in the analysis corresponds to a high-level package in the system.

The Application package



The *Business Services* package:

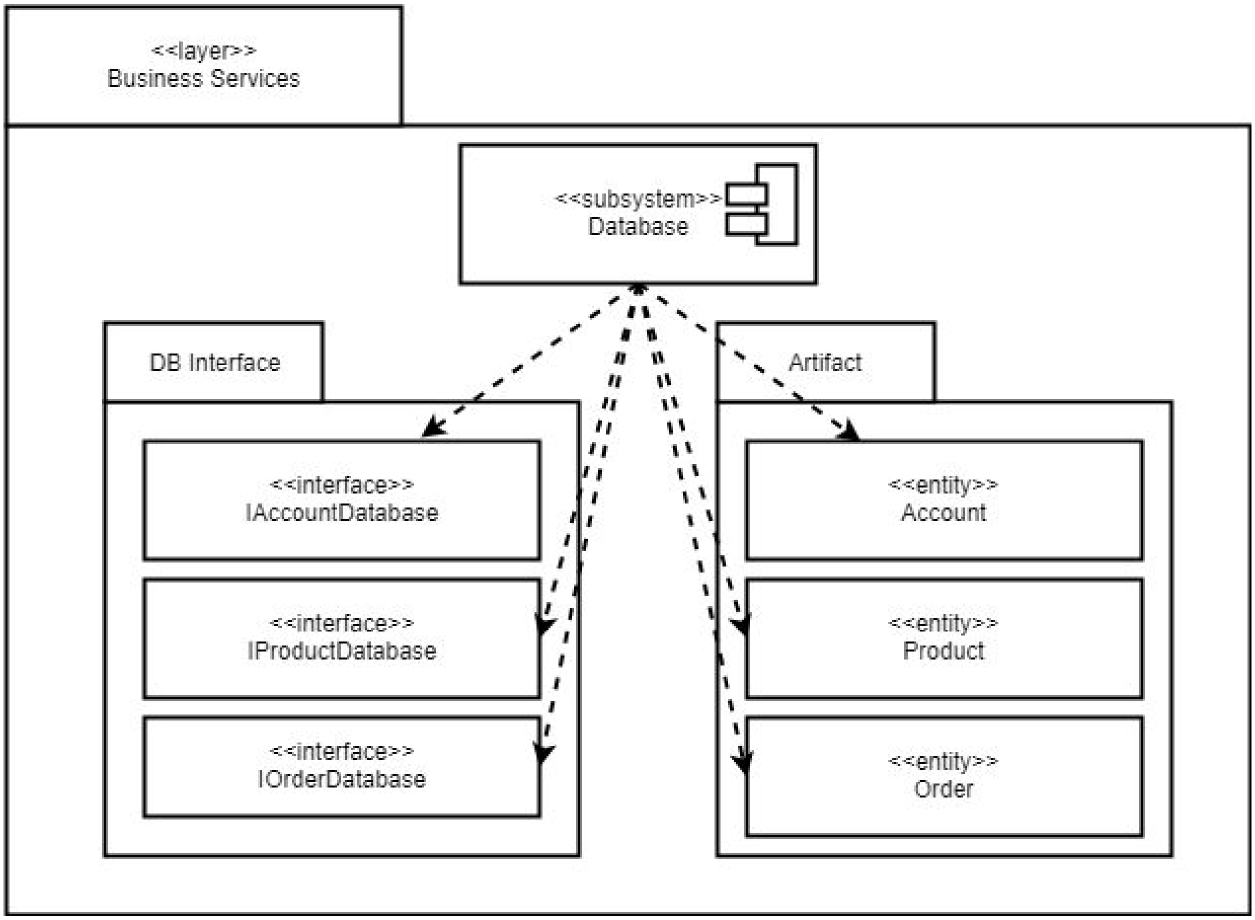


Figure 3.3. The Business Services package

The *Business Services* package contains the *Databas*e subsystem and its interfaces, as well as the entity classes. These elements are common to all use cases.

The *Middleware* package:

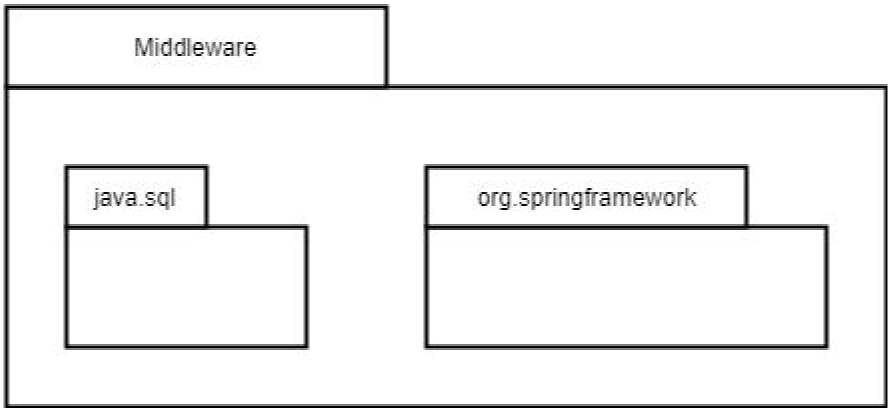


Figure 3.4. The Middleware package

The Middleware package includes Java's SQL package, which provides access to databases and the Java Spring framework, which provides network services.

Packages and their dependencies

As already stated, the Application package depends on the Business Services package, which in turn depends on the Middleware package.

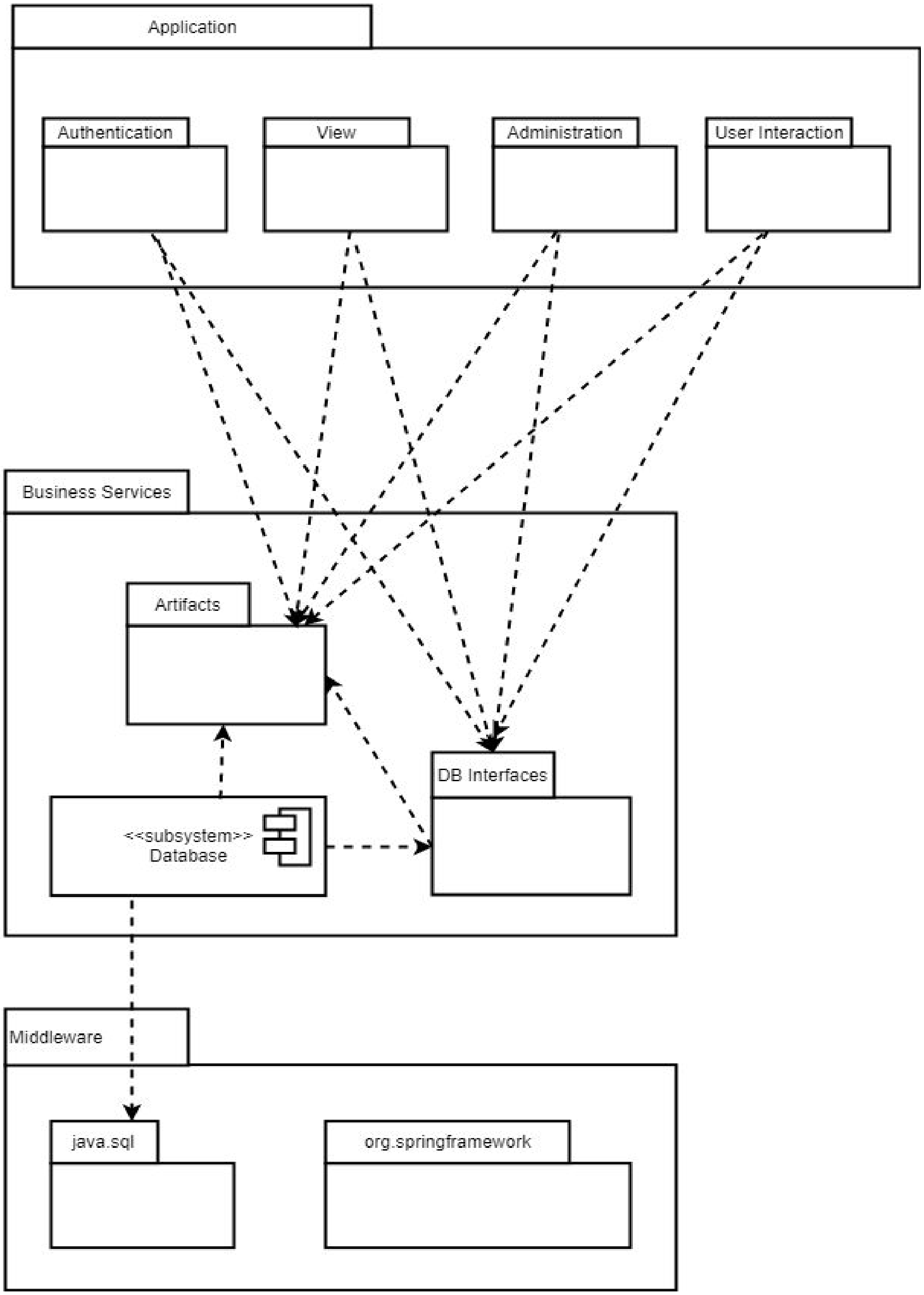


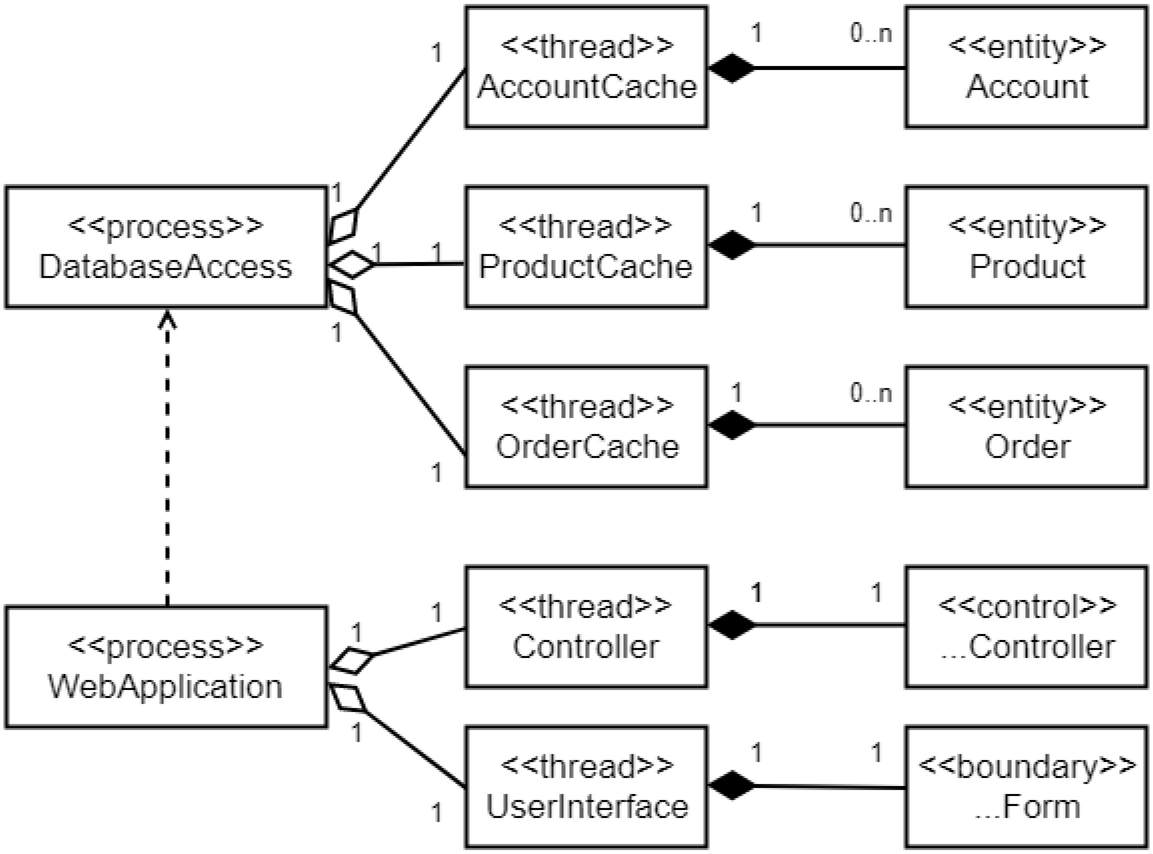
Figure 3.5. Package dependencies diagram

3.1.2 Identify design mechanisms

Analysis mechanism	**Design mechanism**	**Implementation mechanism**
Persistency	RDBMS	JDBC
Security	Web tokens	Java Spring framework
Distribution	REST API	Java Spring framework

Table 3.2. Design and implementation mechanisms

3.2 Describe the run-time architecture



<<thread>> AccountCache
<<entity>> Account
<<process>> DatabaseAccess
<<thread>> ProductCache
<<entity>> Product
<<thread>> OrderCache
<<entity>> Order
<<process>> WebApplication
<<thread>> Controller
<<control>> ...Controller
<<thread>> UserInterface
<<boundary>> ...Form
1
0..n

3.3 Describe distribution

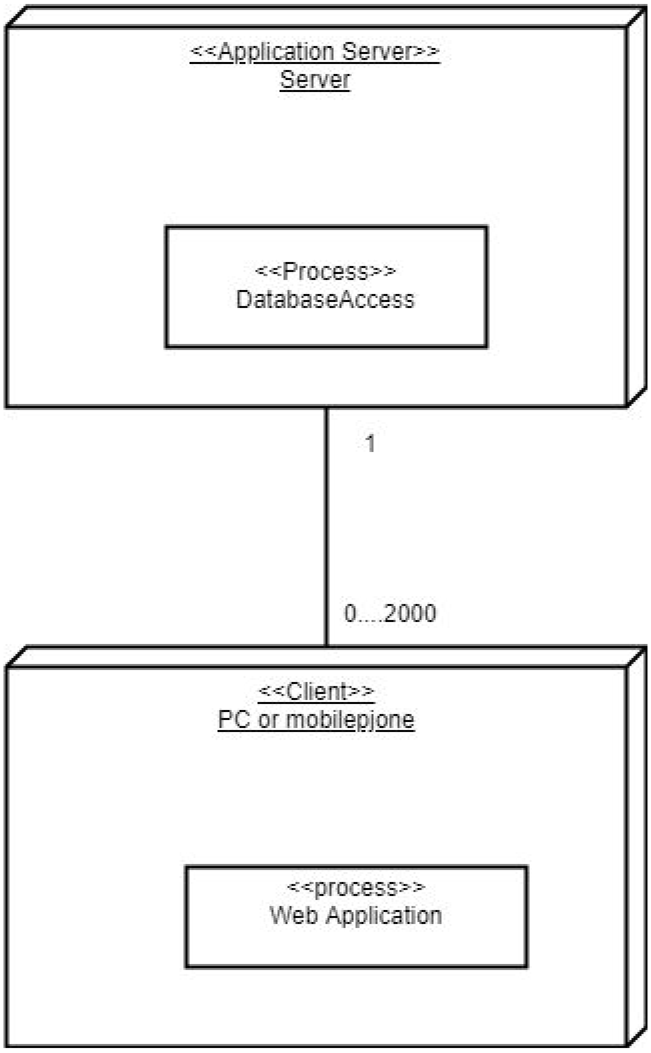


Figure 3.7. The deployment view of the system

3.4 Use-case design

3.4.1 Design sequence diagrams

After incorporating the *Database* subsystem, the model's sequence diagrams are updated as follows. Some method parameters are elided for conciseness and legibility – they are shown in full in the *Class Design* section.

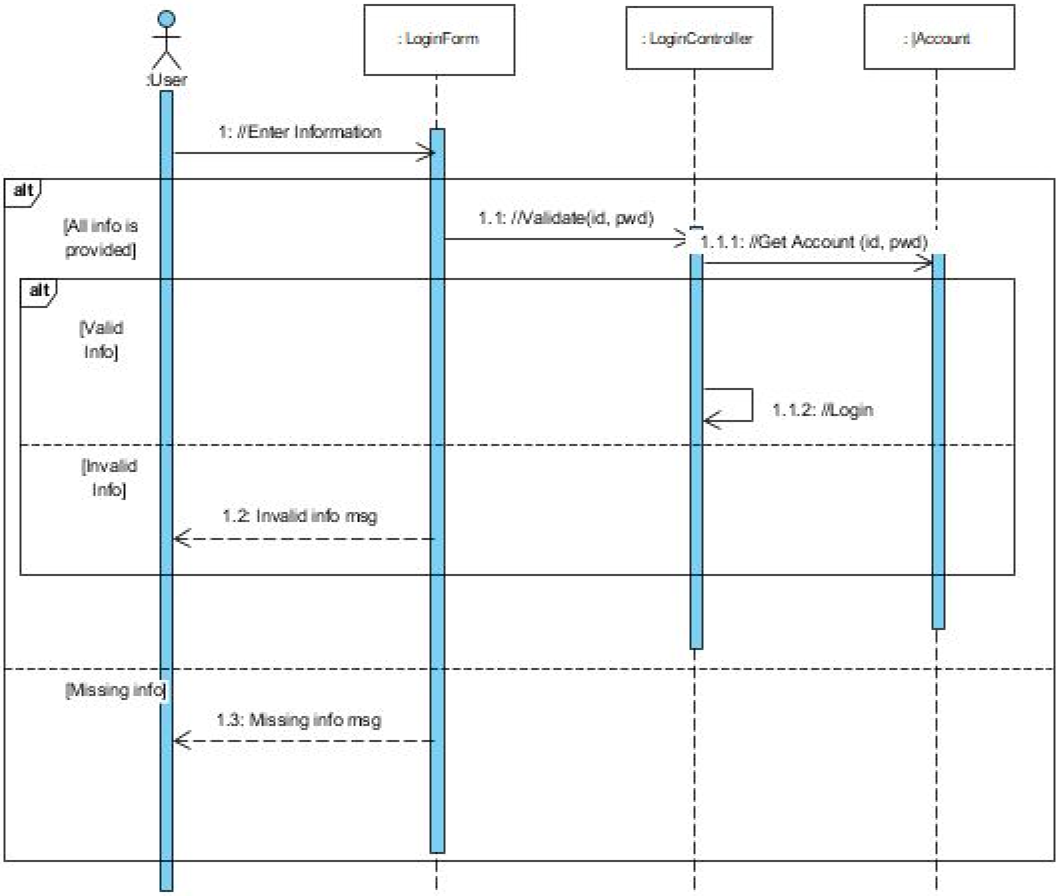


Figure 3.7: Design sequence diagram for the Login use case

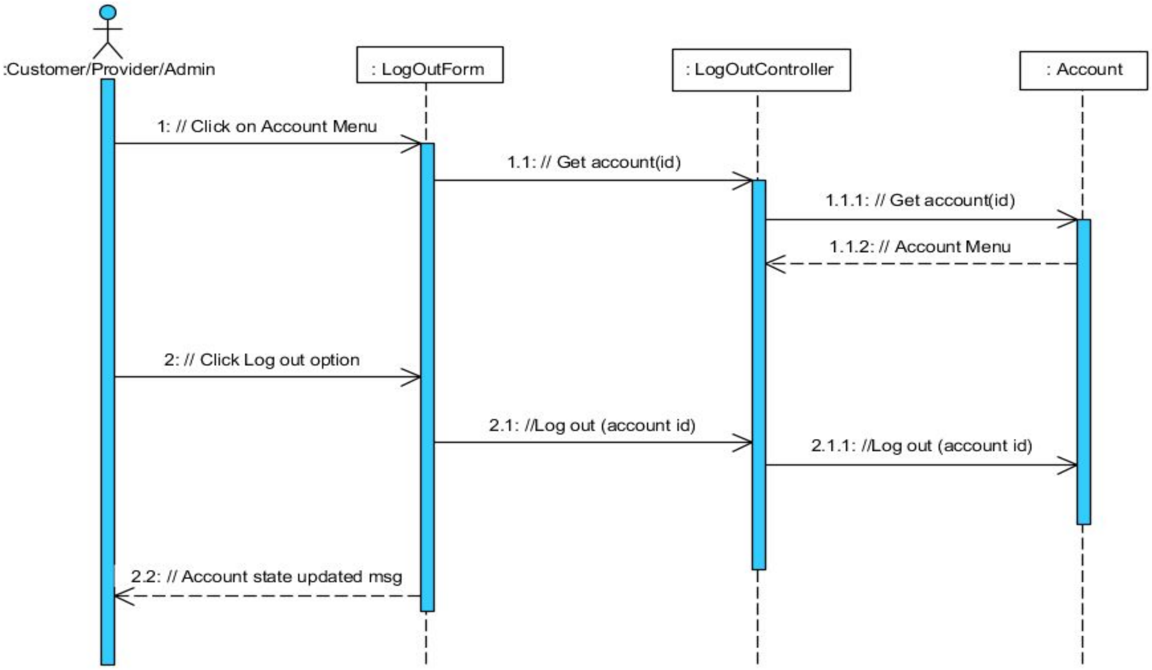

:Customer/Provider/Admin
: LogOutForm
: LogOutController
: Account
1: // Click on Account Menu
1.1: // Get account(id)
1.1.1: // Get account(id)
1.1.2: // Account Menu
2: // Click Log out option
2.1: //Log out (account id)
2.1.1: //Log out (account id)
2.2: // Account state updated msg

Figure 3-8. Design sequence diagram for the Log out use case

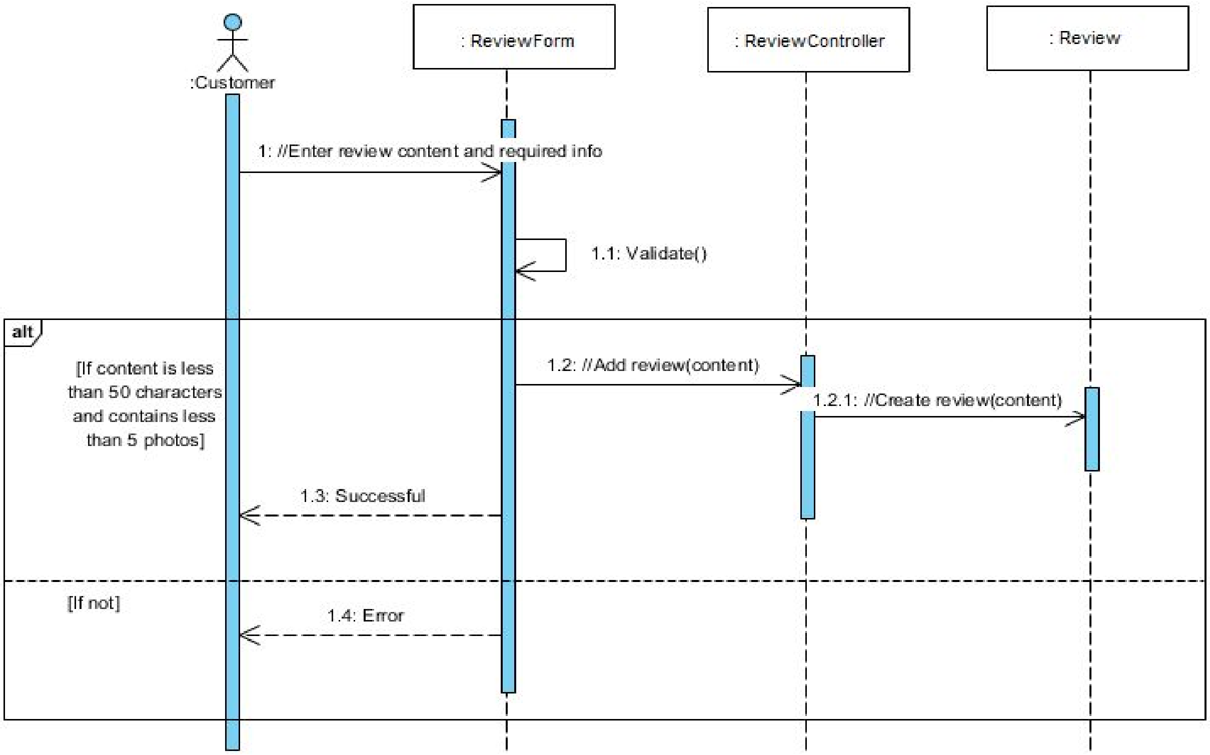
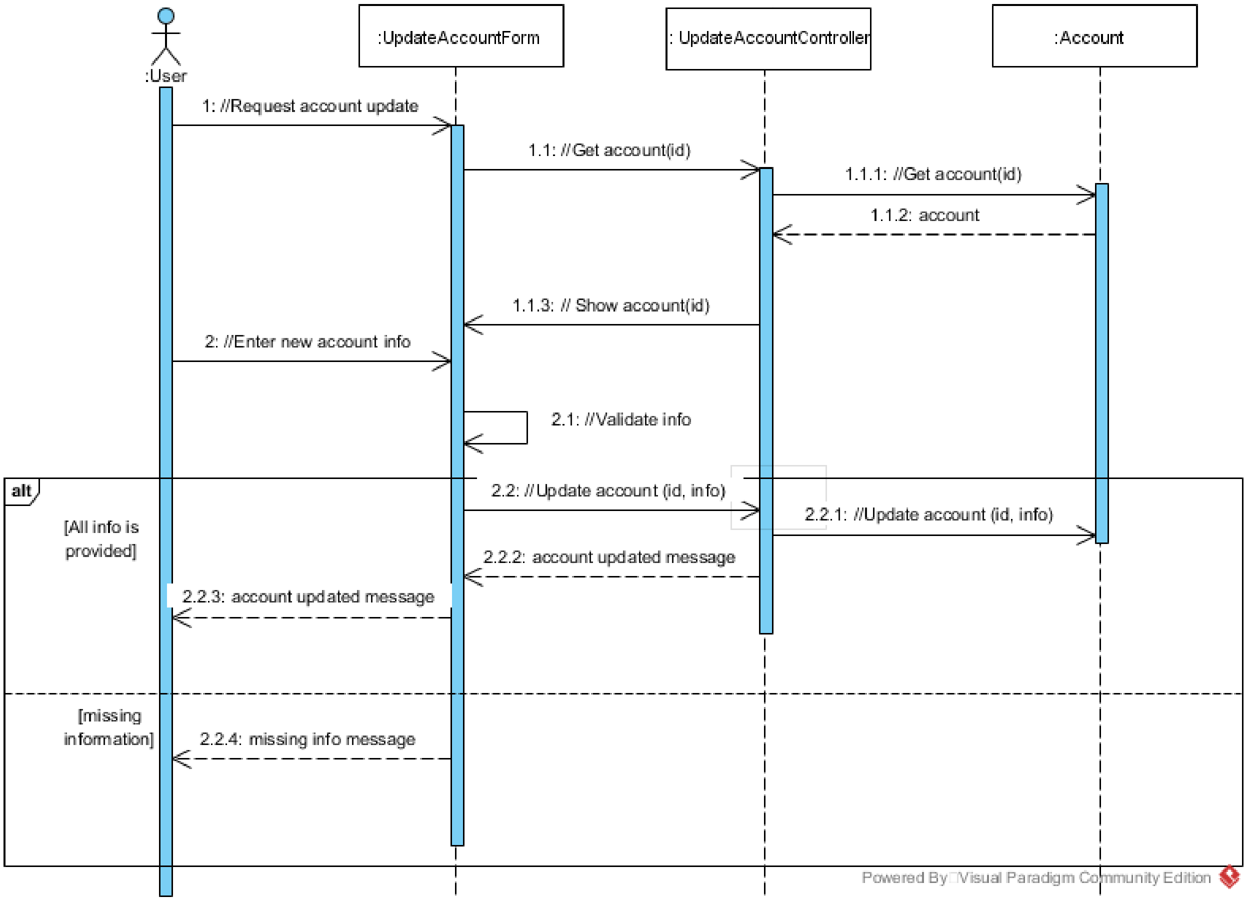

:Customer
: ReviewForm
: ReviewController
: Review
1: //Enter review content and required info
1.1: Validate()
alt
[If content is less than 50 characters and contains less than 5 photos]
1.2: //Add review(content)
1.2.1: //Create review(content)
1.3: Successful
[If not]
1.4: Error

Figure 3.9: Design sequence diagram for the Review Product use case



:User
:UpdateAccountForm
: UpdateAccountController
:Account
1: //Request account update
1.1: //Get account(id)
1.1.1: //Get account(id)
1.1.2: account
1.1.3: // Show account(id)
2: //Enter new account info
2.1: //Validate info
alt
[All info is provided]
2.2: //Update account (id, info)
2.2.1: //Update account (id, info)
2.2.2: account updated message
2.2.3: account updated message
[missing information]
2.2.4: missing info message
Powered By Visual Paradigm Community Edition

Figure 3.11: Design sequence diagram for the Update Profile use case

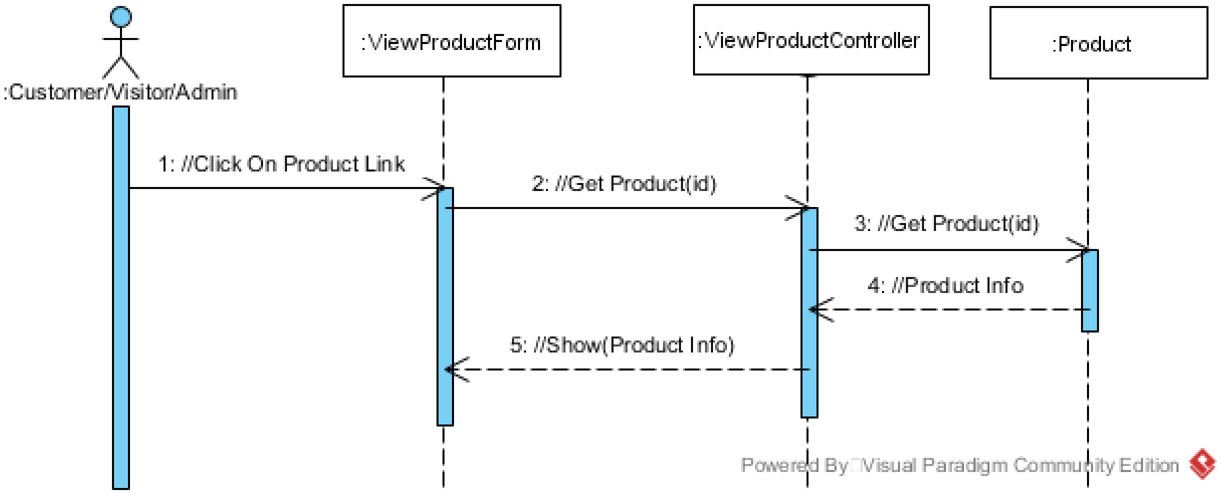


Figure 3.12: Design sequence diagram for the View Product use case

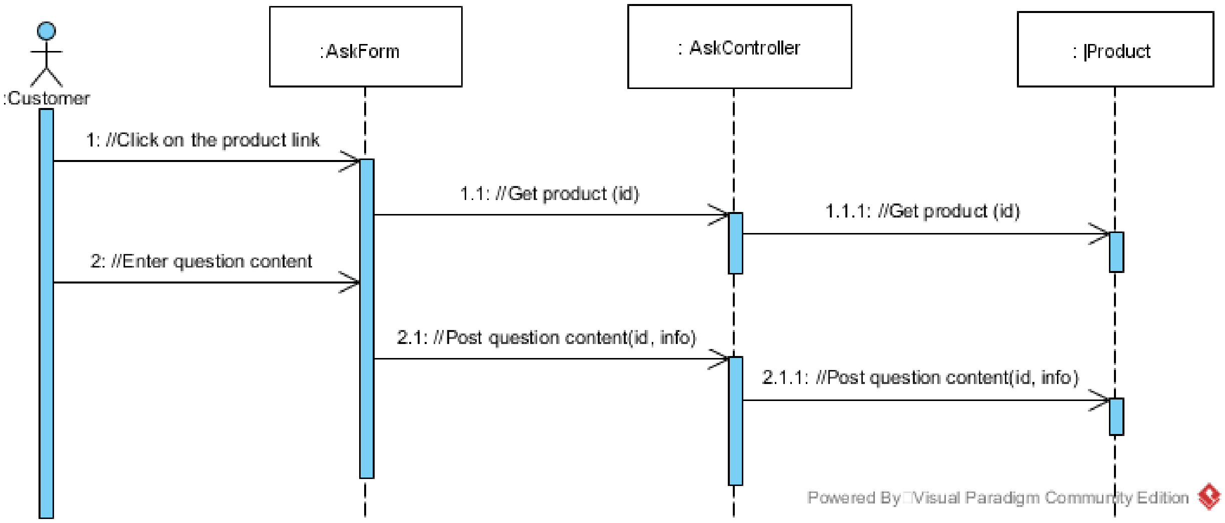

:Customer
:AskForm
: AskController
: |Product
1: //Click on the product link
1.1: //Get product (id)
1.1.1: //Get product (id)
2: //Enter question content
2.1: //Post question content(id, info)
2.1.1: //Post question content(id, info)
Powered By Visual Paradigm Community Edition

Figure 3.13: Design sequence diagram for the Ask Question use case

3.4.2 Design views of participating classes

The updated views of participating classes for each use case are described in the Class Design section.

3.5 Subsystem design

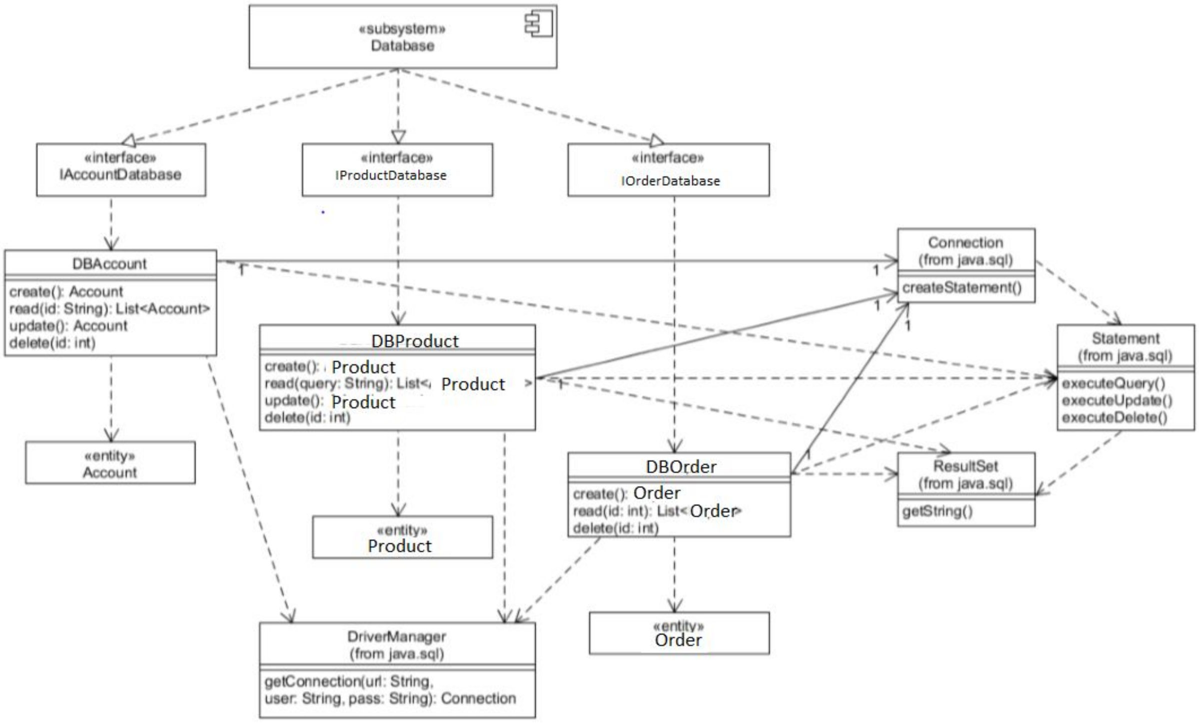


Figure 3-20. Database subsystem elements diagram

3.6 Class design

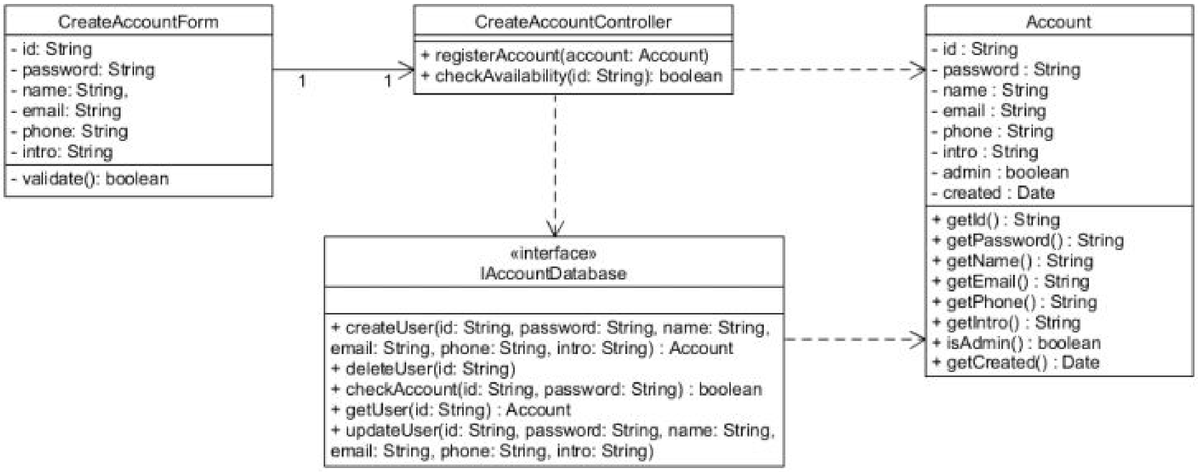


Figure 3-21. Design VOPC for the Create Account use case

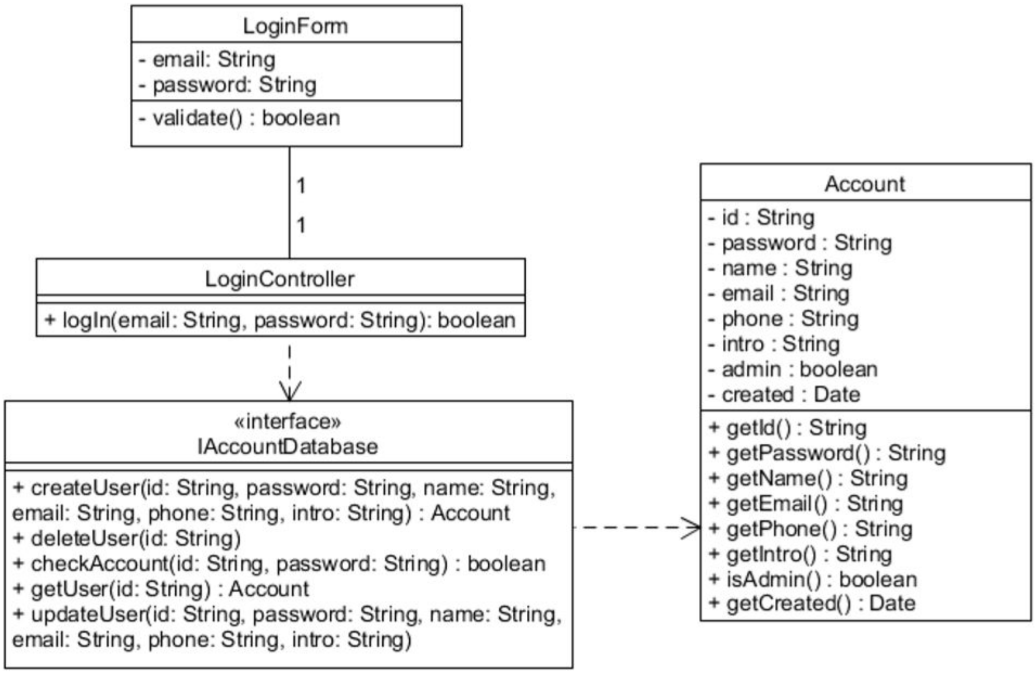


Figure 3-22. Design VOPC for the Login use case

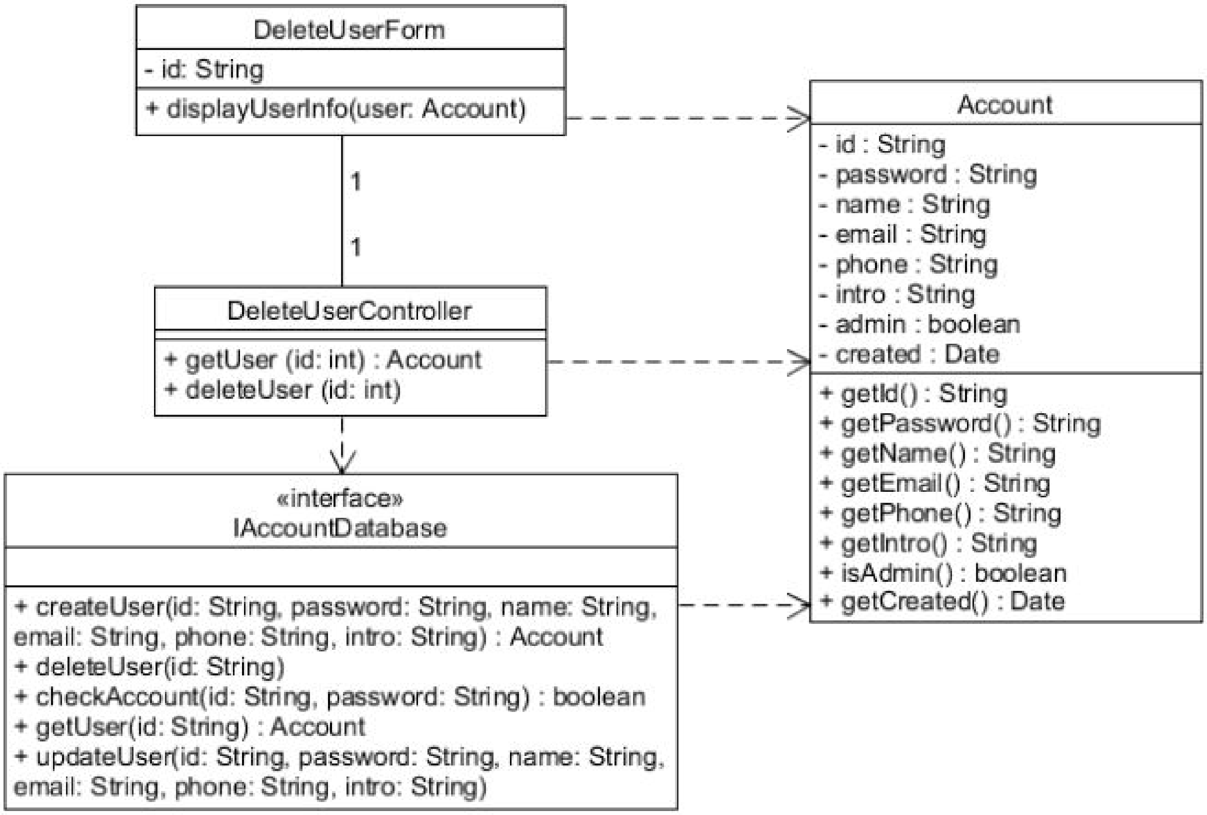


Figure 3-25. Design VOPC for the Delete User use case

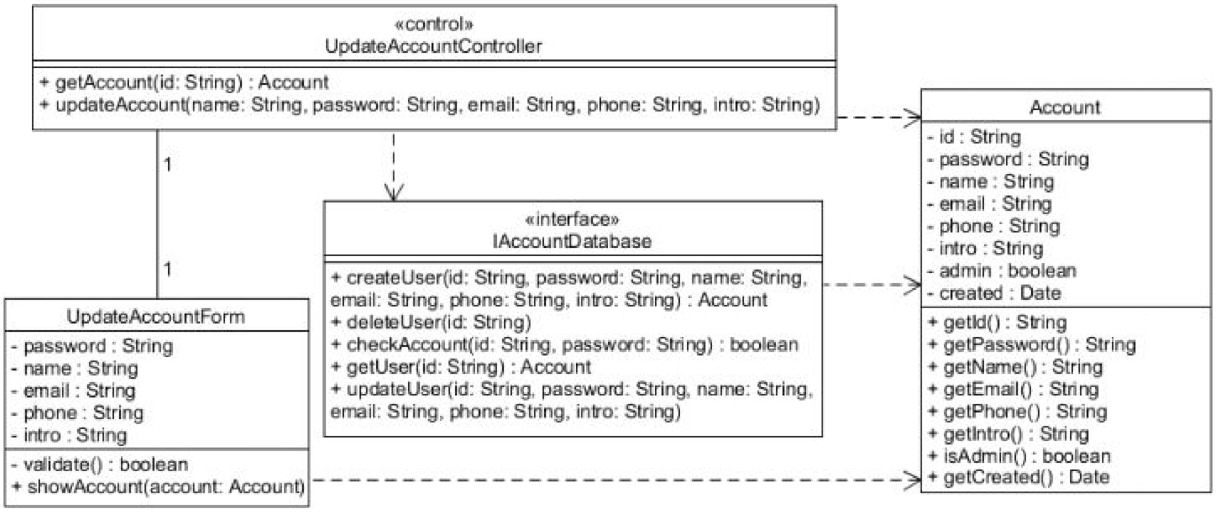


Figure 3-26. Design VOPC for the Update Account use case

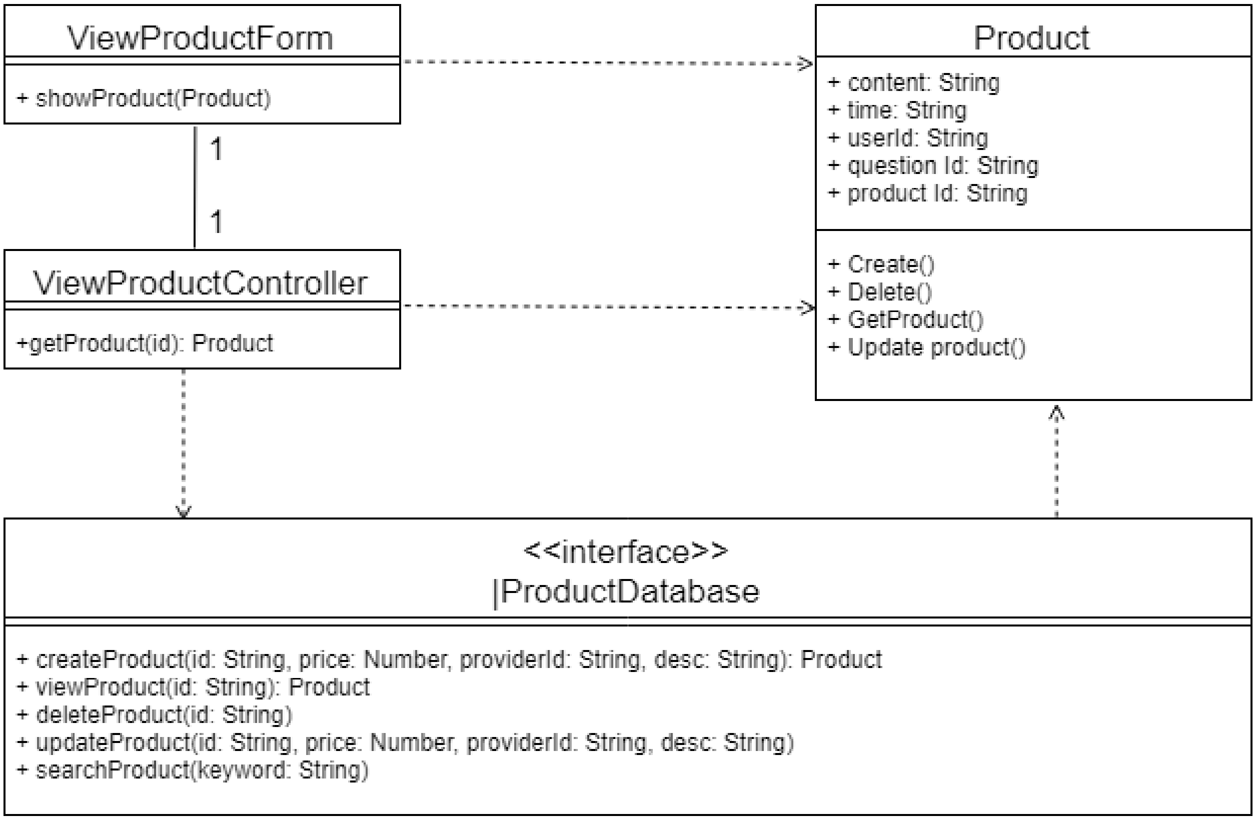


Figure 3-27. Design VOPC for the View Product use case

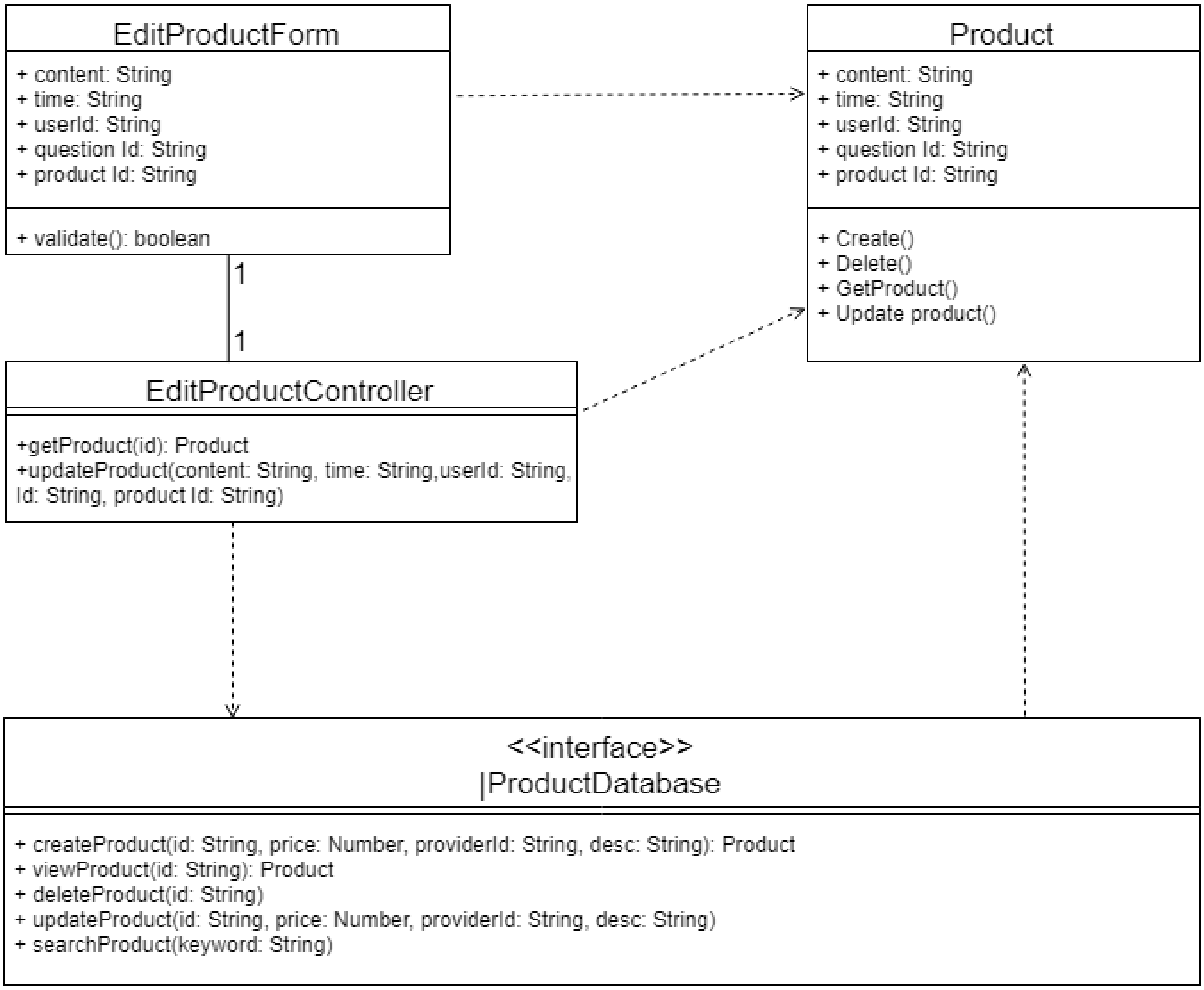


Figure 3-28. Design VOPC for the Edit Product's Information use case

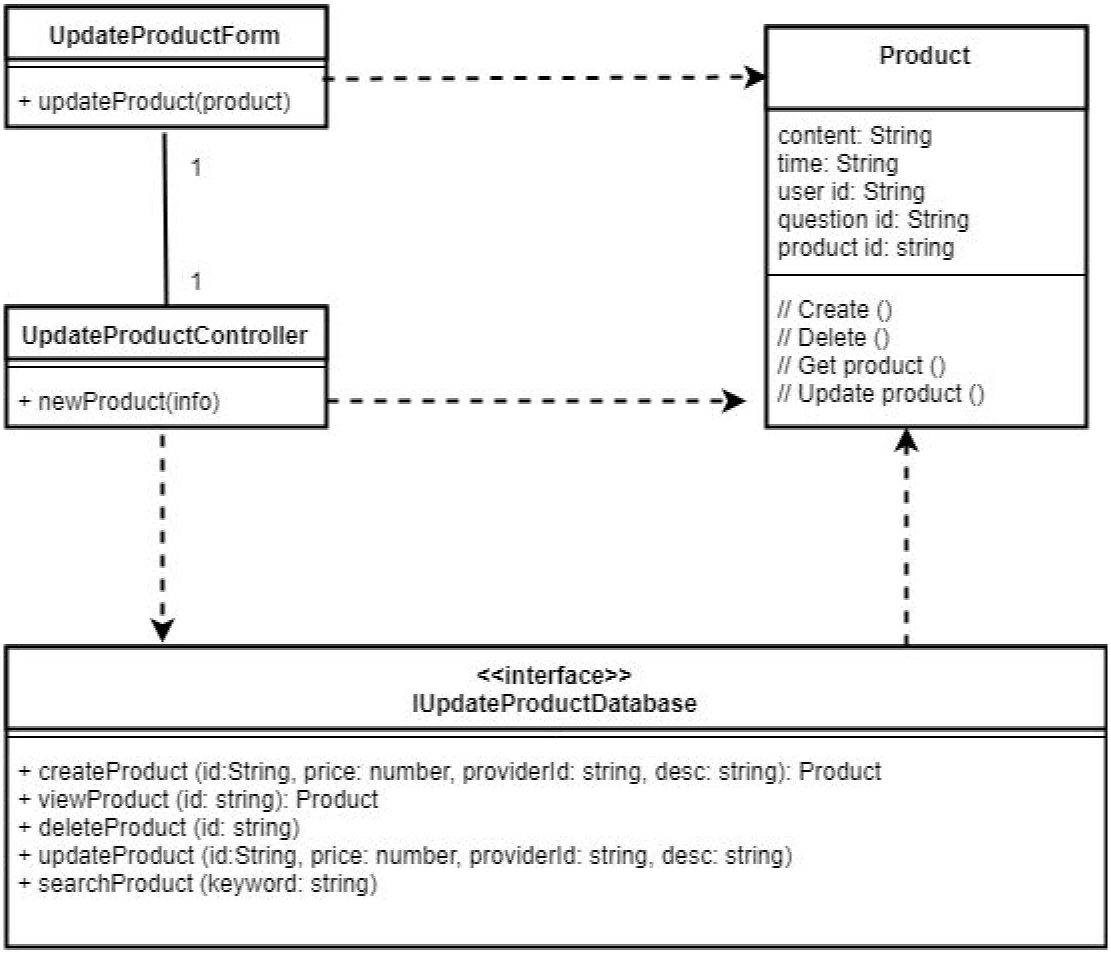


Figure 3-29. Design VOPC for the Update New Product use case

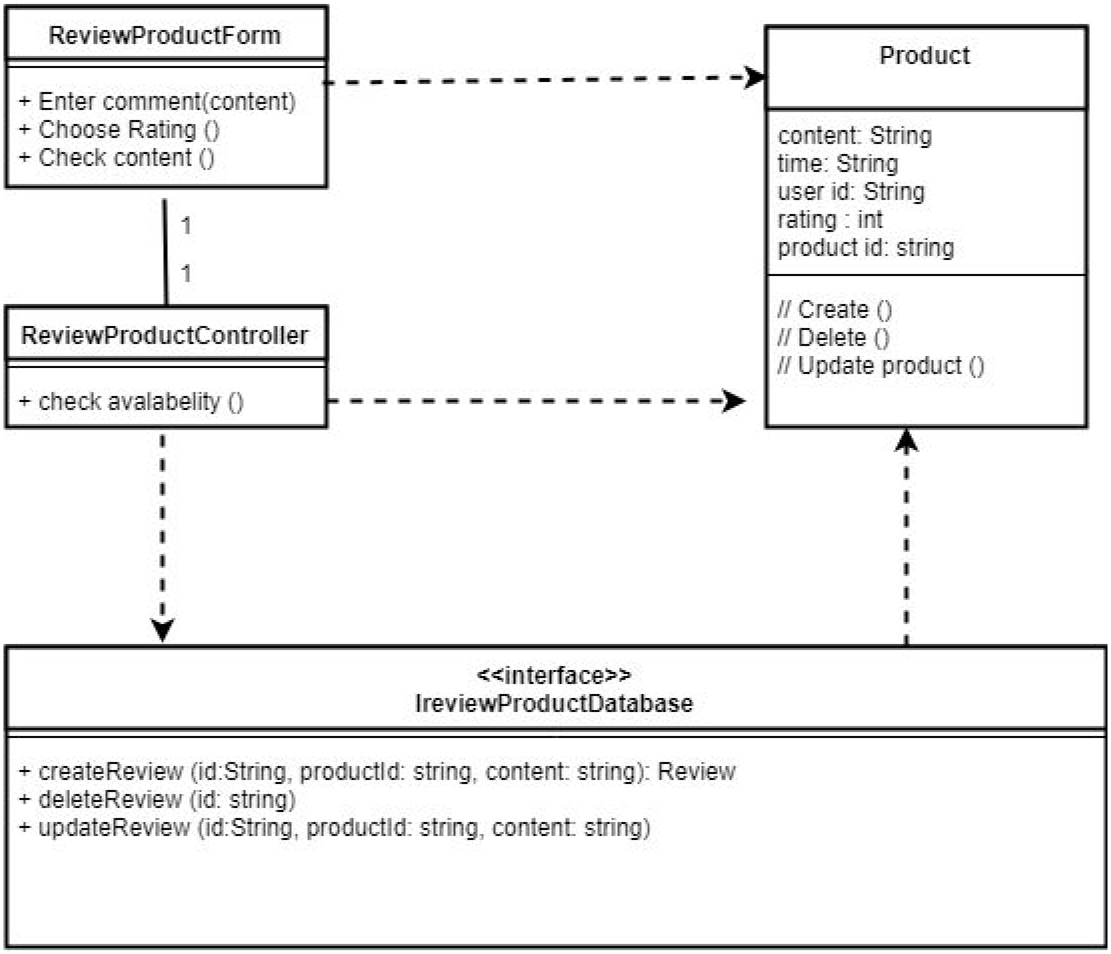


Figure 3-30. Design VOPC for the Review Product use case

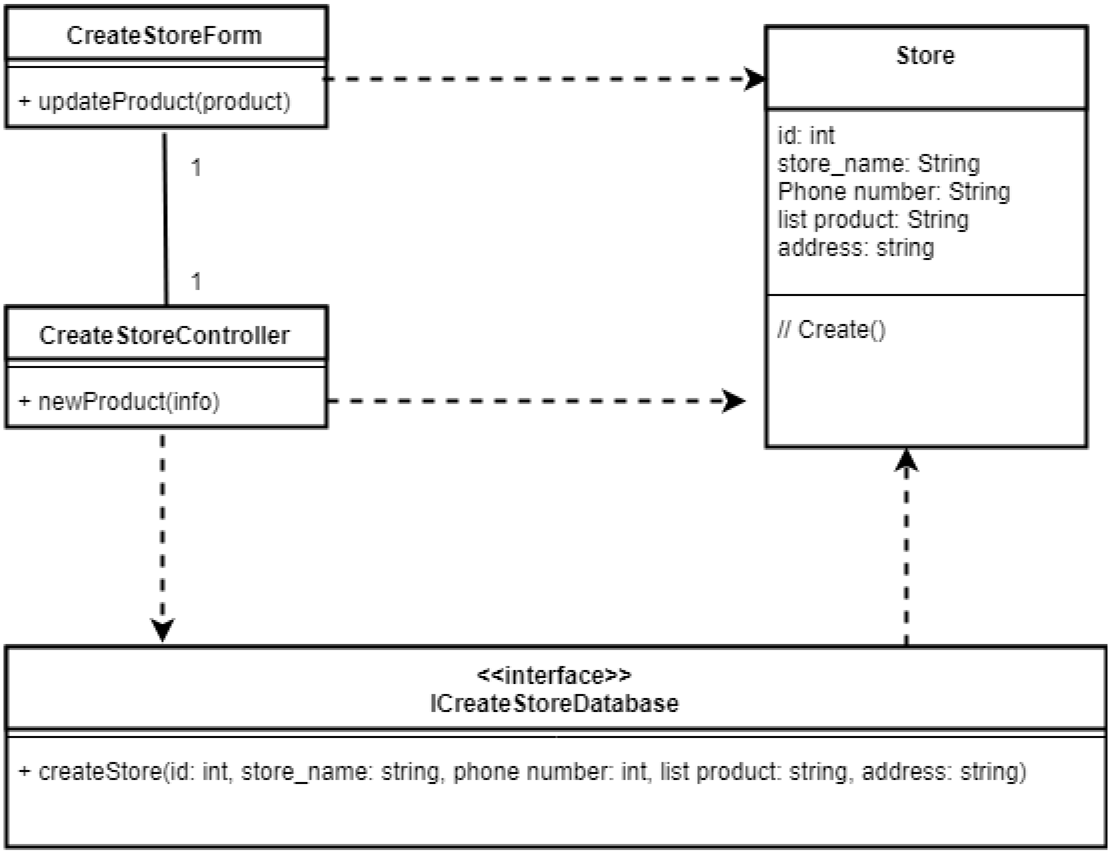


Figure 3-31. Design VOPC for the Create Store use case

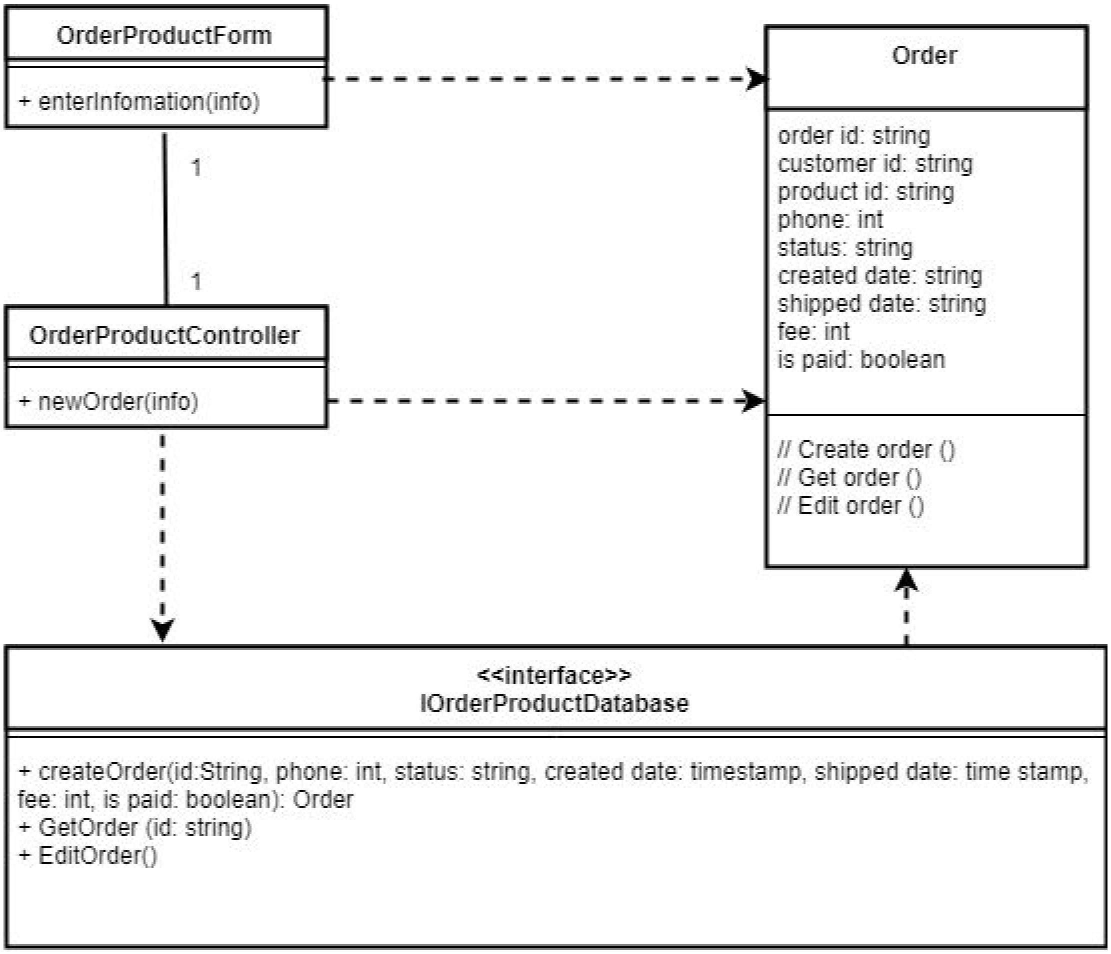


Figure 3-31. Design VOPC for the Order Product use case

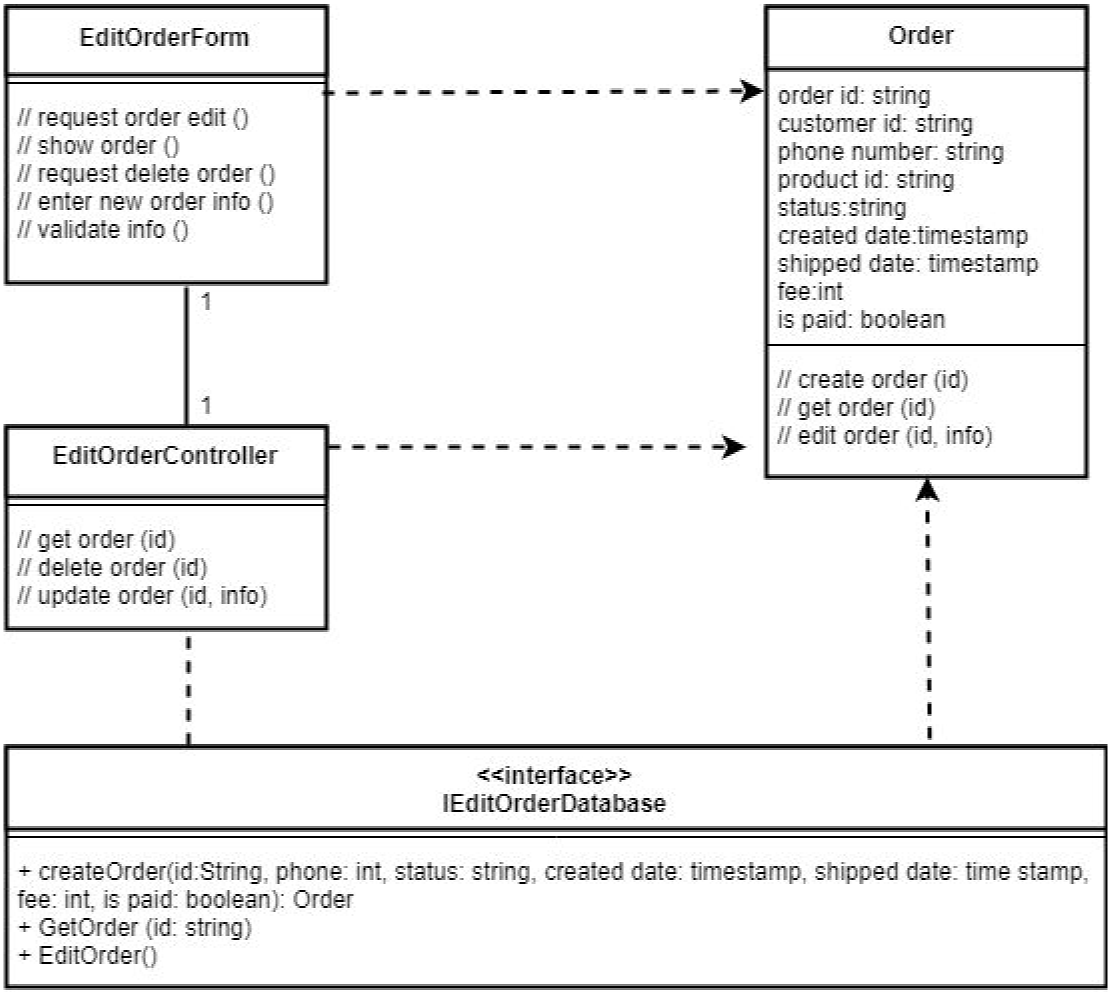


Figure 3-32. Design VOPC for the Edit Order use case

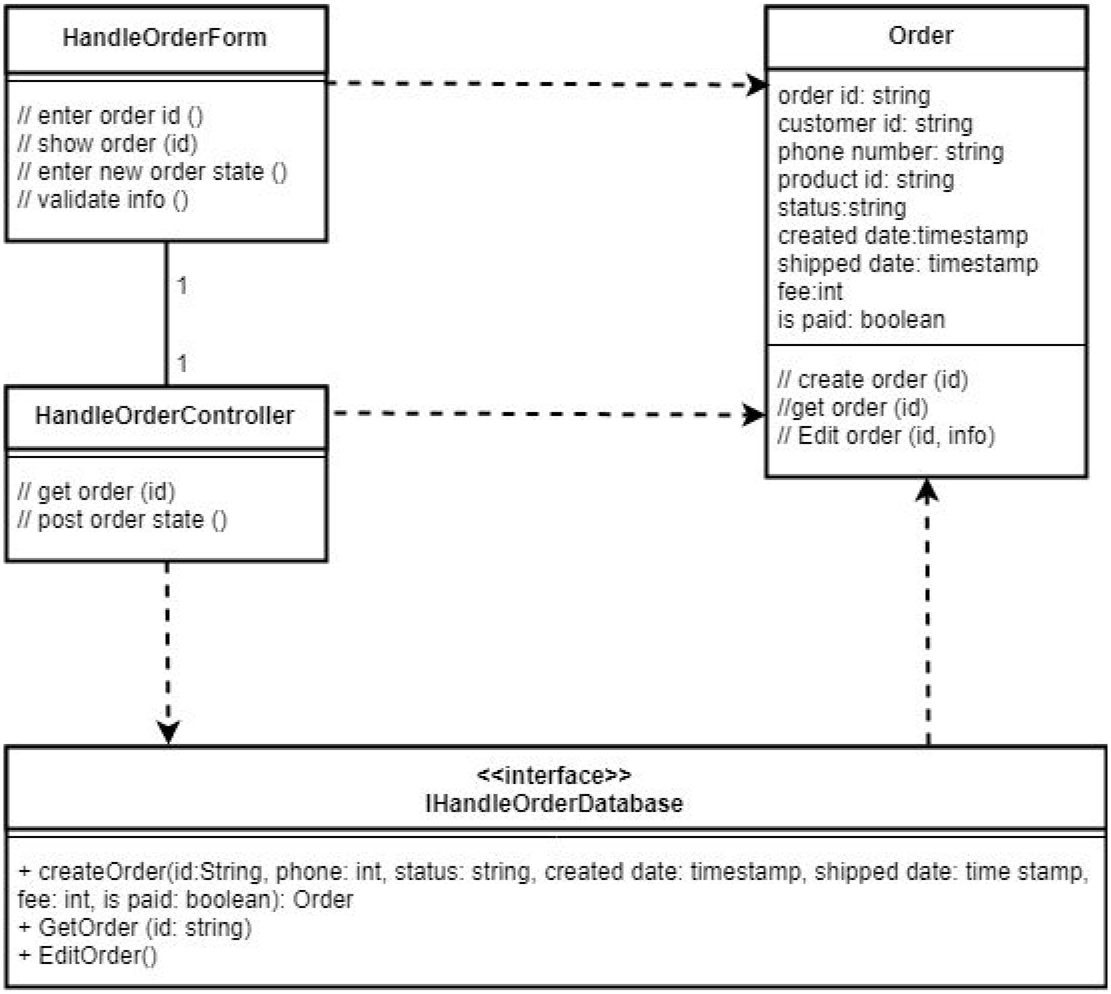


Figure 3-33. Design VOPC for the Handle Order use case

3.7 Database design

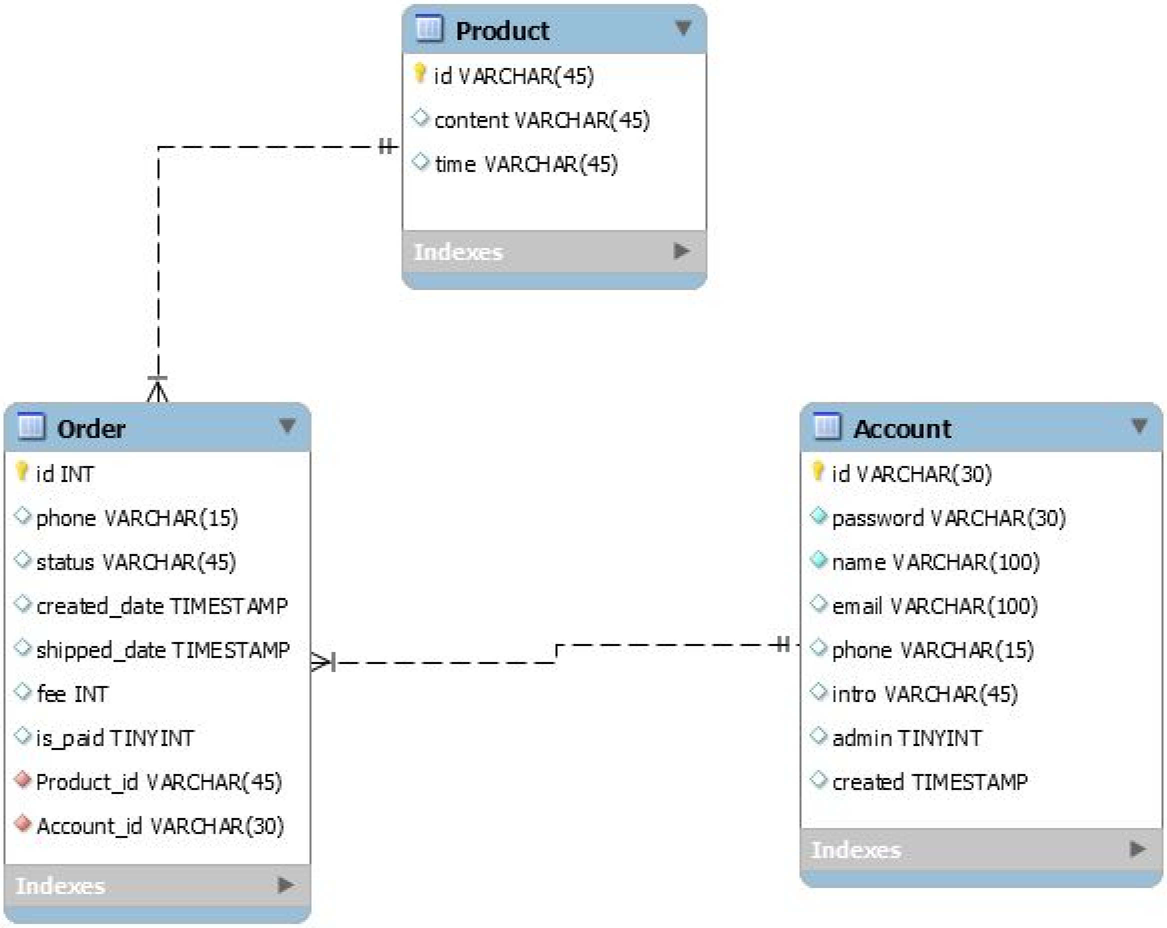


Figure 3-34: The relational data model