

**CAPSTONE PROJECT REPORT**

**Report 4 – Software Design Document**

– HoChiMinh, September 2020 –

**Table of Contents**

[I. Project Report 7](#_heading=h.gjdgxs)

[1. Status Report 7](#_heading=h.30j0zll)

[2. Team Involvements 7](#_heading=h.1fob9te)

[3. Issues/Suggestions 7](#_heading=h.3znysh7)

[4. 状況報告 7](#_heading=h.2et92p0)

[5. チーム-関与事項 8](#_heading=h.tyjcwt)

[6. 課題/提案 8](#_heading=h.3dy6vkm)

[II. Software Design Document 9](#_heading=h.1t3h5sf)

[1. Overall Description 9](#_heading=h.4d34og8)

[1.1 Assumptions 9](#_heading=h.2s8eyo1)

[1.2 Design Constraints 9](#_heading=h.17dp8vu)

[2. System Architecture Design 9](#_heading=h.3rdcrjn)

[2.1 Overall Architecture 9](#_heading=h.26in1rg)

[2.2 System Architecture 9](#_heading=h.49x2ik5)

[2.3 Package Diagram 10](#_heading=h.147n2zr)

[3. System Detailed Design 14](#_heading=h.meukdy)

[3.1 Overall 14](#_heading=h.36ei31r)

[a. Class Diagram 14](#_heading=h.1ljsd9k)

[b. Sequence Diagram(s) 14](#_heading=h.41mghml)

[c. Activity Diagram(s) 19](#_heading=h.rjefff)

[d. State Machine Diagram(s) 25](#_heading=h.2pta16n)

[4. Class Specifications 26](#_heading=h.3oy7u29)

[4.1 User 26](#_heading=h.243i4a2)

[4.2 Role 27](#_heading=h.338fx5o)

[4.3 ERole 27](#_heading=h.42ddq1a)

[4.4 Package 28](#_heading=h.wnyagw)

[4.5 Service 28](#_heading=h.1vsw3ci)

[4.6 Booking 29](#_heading=h.2uxtw84)

[4.7 EBookingStatus 29](#_heading=h.3u2rp3q)

[4.8 Album 30](#_heading=h.odc9jc)

[4.9 Image 30](#_heading=h.1nia2ey)

[4.10 Category 31](#_heading=h.2mn7vak)

[4.11 DayOfWeek 31](#_heading=h.3ls5o66)

[4.12 Location 32](#_heading=h.4kx3h1s)

[4.13 BookingComment 32](#_heading=h.1f7o1he)

[4.14 BookingDetail 33](#_heading=h.2eclud0)

[4.15 BusyDay 33](#_heading=h.3dhjn8m)

[4.16 Thread 34](#_heading=h.l7a3n9)

[4.17 ThreadTopic 35](#_heading=h.1kc7wiv)

[4.18 ThreadComment 35](#_heading=h.2jh5peh)

[4.19 ReturningType 36](#_heading=h.3im3ia3)

[4.20 EReturningType 36](#_heading=h.4hr1b5p)

[4.21 TimeLocationDetail 36](#_heading=h.1c1lvlb)

[4.22 Variable 37](#_heading=h.2b6jogx)

[5. Data & Database Design 38](#_heading=h.3abhhcj)

[5.1 Database Design 38](#_heading=h.1pgrrkc)

[5.1.1 users 38](#_heading=h.34g0dwd)

[5.1.2 roles 39](#_heading=h.13qzunr)

[5.1.3 photographer\_packages 39](#_heading=h.22vxnjd)

[5.1.4 packages\_services 39](#_heading=h.320vgez)

[5.1.5 services 39](#_heading=h.415t9al)

[5.1.6 bookings 39](#_heading=h.vgdtq7)

[5.1.7 albums 40](#_heading=h.1ulbmlt)

[5.1.8 locations 40](#_heading=h.2tq9fhf)

[5.1.9 categories 41](#_heading=h.3sv78d1)

[5.1.10 album\_images 41](#_heading=h.n5rssn)

[5.1.11 images 41](#_heading=h.1maplo9)

[5.1.12 customers\_albums 41](#_heading=h.2lfnejv)

[5.1.13 working\_day\_of\_week 41](#_heading=h.3kkl7fh)

[5.1.14 busy\_days 42](#_heading=h.4jpj0b3)

[5.1.15 returning\_types 42](#_heading=h.1e03kqp)

[5.1.16 booking\_details 42](#_heading=h.2d51dmb)

[5.1.17 time\_location\_details 42](#_heading=h.3c9z6hx)

[5.1.18 booking\_comments 43](#_heading=h.4bewzdj)

[5.1.19 threads 43](#_heading=h.15phjt5)

[5.1.20 thread\_comments 43](#_heading=h.24ufcor)

[5.1.21 thread\_topics 43](#_heading=h.33zd5kd)

[5.1.22 variables 43](#_heading=h.434ayfz)

[5.2 Data File Design 44](#_heading=h.xevivl)

[6. Algorithm 44](#_heading=h.1wjtbr7)

[6.1 Multiple-factors weighted sorting 44](#_heading=h.4gjguf0)

[6.1.1 Definition 44](#_heading=h.2vor4mt)

[6.1.2 Define problem 44](#_heading=h.1au1eum)

[6.1.3 Solution 44](#_heading=h.3utoxif)

[6.1.4 Complexity 45](#_heading=h.29yz7q8)

[6.1.5 Flowchart 46](#_heading=h.p49hy1)

[6.2 Naive Bayes Classifier 47](#_heading=h.1o97atn)

[6.2.1 Definition 47](#_heading=h.488uthg)

[6.2.2 Define problem 47](#_heading=h.2ne53p9)

[6.2.3 Solution 47](#_heading=h.12jfdx2)

[6.2.4 Complexity 50](#_heading=h.3mj2wkv)

**List of Figures**

[Figure 1 Overall Architecture 9](#_heading=h.lnxbz9)

[Figure 2 Mobile App Architecture 10](#_heading=h.2p2csry)

[Figure 3 Webapi Package Diagram 11](#_heading=h.3o7alnk)

[Figure 4 Package Diagram – Mobile app 13](#_heading=h.3s49zyc)

[Figure 5 Class Diagram 14](#_heading=h.45jfvxd)

[Figure 6 Sequence Diagram - Add Category 15](#_heading=h.2koq656)

[Figure 7 Sequence Diagram - Edit Category 15](#_heading=h.zu0gcz)

[Figure 8 Sequence Diagram - Remove Category 16](#_heading=h.3jtnz0s)

[Figure 9 Sequence Diagram - Add returning type 16](#_heading=h.1yyy98l)

[Figure 10 Sequence Diagram - Edit returning type 17](#_heading=h.4iylrwe)

[Figure 11 Sequence Diagram - Remove returning type 17](#_heading=h.2y3w247)

[Figure 12 Sequence Diagram - Block User 18](#_heading=h.1d96cc0)

[Figure 13 Sequence Diagram - Unblock User 18](#_heading=h.3x8tuzt)

[Figure 14 Sequence Diagram - Change Variables 19](#_heading=h.2ce457m)

[Figure 15 Activity Diagram - Make Request](about:blank) [20](#_heading=h.21od6so)

[Figure 16 Activity Diagram – Edit Request 21](#_heading=h.3bj1y38)

[Figure 17 Activity Diagram - Cancel booking 22](#_heading=h.1qoc8b1)

[Figure 18 Activity Diagram - Rating](about:blank) [23](#_heading=h.gtnh0h)

[Figure 19 Activity Diagram - Accept booking 24](#_heading=h.4anzqyu)

[Figure 20 Activity Diagram - Reject booking](about:blank) [25](#_heading=h.30tazoa)

[Figure 21 State Machine Diagram - Booking 26](#_heading=h.14ykbeg)

[Figure 22 Database Design 38](#_heading=h.49gfa85)

[Figure 23 Flowchart - Multiple factors sorting 46](#_heading=h.393x0lu)

**List of Tables**

[Table 1 Package Diagram Description 12](#_heading=h.vx1227)

[Table 2 Package Diagram Description 14](#_heading=h.279ka65)

[Table 3 Class Diagram - User 27](#_heading=h.j8sehv)

[Table 4 Class Diagram - Role 27](#_heading=h.1idq7dh)

[Table 5 Class Diagram - ERole 28](#_heading=h.2hio093)

[Table 6 Class Diagram - Package 28](#_heading=h.3gnlt4p)

[Table 7 Class Diagram - Service 29](#_heading=h.4fsjm0b)

[Table 8 Class Diagram - Booking 29](#_heading=h.1a346fx)

[Table 9 Class Diagram - EBookingStatus 30](#_heading=h.2981zbj)

[Table 10 Class Diagram - Album 30](#_heading=h.38czs75)

[Table 11 Class Diagram - Image 31](#_heading=h.47hxl2r)

[Table 12 Class Diagram - Category 31](#_heading=h.11si5id)

[Table 13 Class Diagram - DayOfWeek 32](#_heading=h.20xfydz)

[Table 14 Class Diagram - Location 32](#_heading=h.302dr9l)

[Table 15 Class Diagram - BookingComment 33](#_heading=h.3z7bk57)

[Table 16 Class Diagram - BookingDetail 33](#_heading=h.thw4kt)

[Table 17 Class Diagram - BusyDay 34](#_heading=h.1smtxgf)

[Table 18 Class Diagram - Thread 34](#_heading=h.356xmb2)

[Table 19 Class Diagram - ThreadTopic 35](#_heading=h.44bvf6o)

[Table 20 Class Diagram - ThreadComment 35](#_heading=h.ymfzma)

[Table 21 Class Diagram - ReturningType 36](#_heading=h.1xrdshw)

[Table 22 Class Diagram - EReturningType 36](#_heading=h.2wwbldi)

[Table 23 Class Diagram – TimeLocationDetail 37](#_heading=h.3w19e94)

[Table 24 Class Diagram - Variable 37](#_heading=h.qbtyoq)

[Table 25 Database - users 39](#_heading=h.2olpkfy)

[Table 26 Database - roles 39](#_heading=h.3nqndbk)

[Table 27 Database - photographer\_packages 39](#_heading=h.i17xr6)

[Table 28 Database - packages\_services 39](#_heading=h.1h65qms)

[Table 29 Database - services 39](#_heading=h.2gb3jie)

[Table 30 Database - bookings 40](#_heading=h.3fg1ce0)

[Table 31 Database - albums 40](#_heading=h.4ekz59m)

[Table 32 Database - locations 41](#_heading=h.18vjpp8)

[Table 33 Database - categories 41](#_heading=h.280hiku)

[Table 34 Database - album\_images 41](#_heading=h.375fbgg)

[Table 35 Database - images 41](#_heading=h.46ad4c2)

[Table 36 Database - customer\_albums 41](#_heading=h.10kxoro)

[Table 37 Database - working\_day\_of\_week 42](#_heading=h.1zpvhna)

[Table 38 Database - busy\_days 42](#_heading=h.2yutaiw)

[Table 39 Database - returning\_types 42](#_heading=h.3xzr3ei)

[Table 40 Database - booking\_details 42](#_heading=h.sabnu4)

[Table 41 Database - time\_location\_details 43](#_heading=h.1rf9gpq)

[Table 42 Database - booking\_comments 43](#_heading=h.2qk79lc)

[Table 43 Database - threads 43](#_heading=h.3pp52gy)

[Table 44 Database - thread\_comments 43](#_heading=h.jzpmwk)

[Table 45 Database - thread\_topics 43](#_heading=h.1j4nfs6)

[Table 46 Database - variables 44](#_heading=h.2i9l8ns)

[Table 47 Date File Design 44](#_heading=h.3hej1je)

# I. Project Report

## 1. Status Report

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Work Item** | **Status** | **Notes (Work Item in Details)** |
| 1 | Overall Description | Completed |  |
| 2 | System Architecture Design | Completed |  |
| 3 | System Detailed Design | Completed |  |
| 4 | Class Specifications | Completed |  |
| 5 | Data & Database Design | Completed |  |
| 6 | Algorithm | Completed |  |

## 2. Team Involvements

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Task** | **Member** | **Notes (Task Details, etc.)** |
| 1 | Overall Description | Kiendst |  |
| 2 | System Architecture Design | Thaott, Datbc |  |
| 3 | System Detailed Design | Thaott, Datbc |  |
| 4 | Class Specifications | Thaott |  |
| 5 | Data & Database Design | Thaott |  |
| 6 | Algorithm | Thaott |  |

## 3. Issues/Suggestions

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Issue** | **Status** | **Notes (Solution, Suggestion, etc.)** |
|  |  |  |  |

## 4. 状況報告

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **課題項目** | **状況** | **注意** |
| 1 | 概要説明 | 完成 |  |
| 2 | システム建築設計 | 完成 |  |
| 3 | システム詳細設計 | 完成 |  |
| 4 | クラス仕様 | 完成 |  |
| 5 | データ＆データベース設計 | 完成 |  |
| 6 | アルゴリズム | 完成 |  |

## 5. チーム-関与事項

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **課題** | **メンバー** | **注意** |
| 1 | 概要説明 | Kiendst |  |
| 2 | システム建築設計 | Thaott, Datbc |  |
| 3 | システム詳細設計 | Thaott, Datbc |  |
| 4 | クラス仕様 | Thaott |  |
| 5 | データ＆データベース設計 | Thaott |  |
| 6 | アルゴリズム | Thaott |  |

## 6. 課題/提案

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **課題** | **状況** | **注意** |
|  |  |  |  |

# II. Software Design Document

## 1. Overall Description

### 1.1 Assumptions

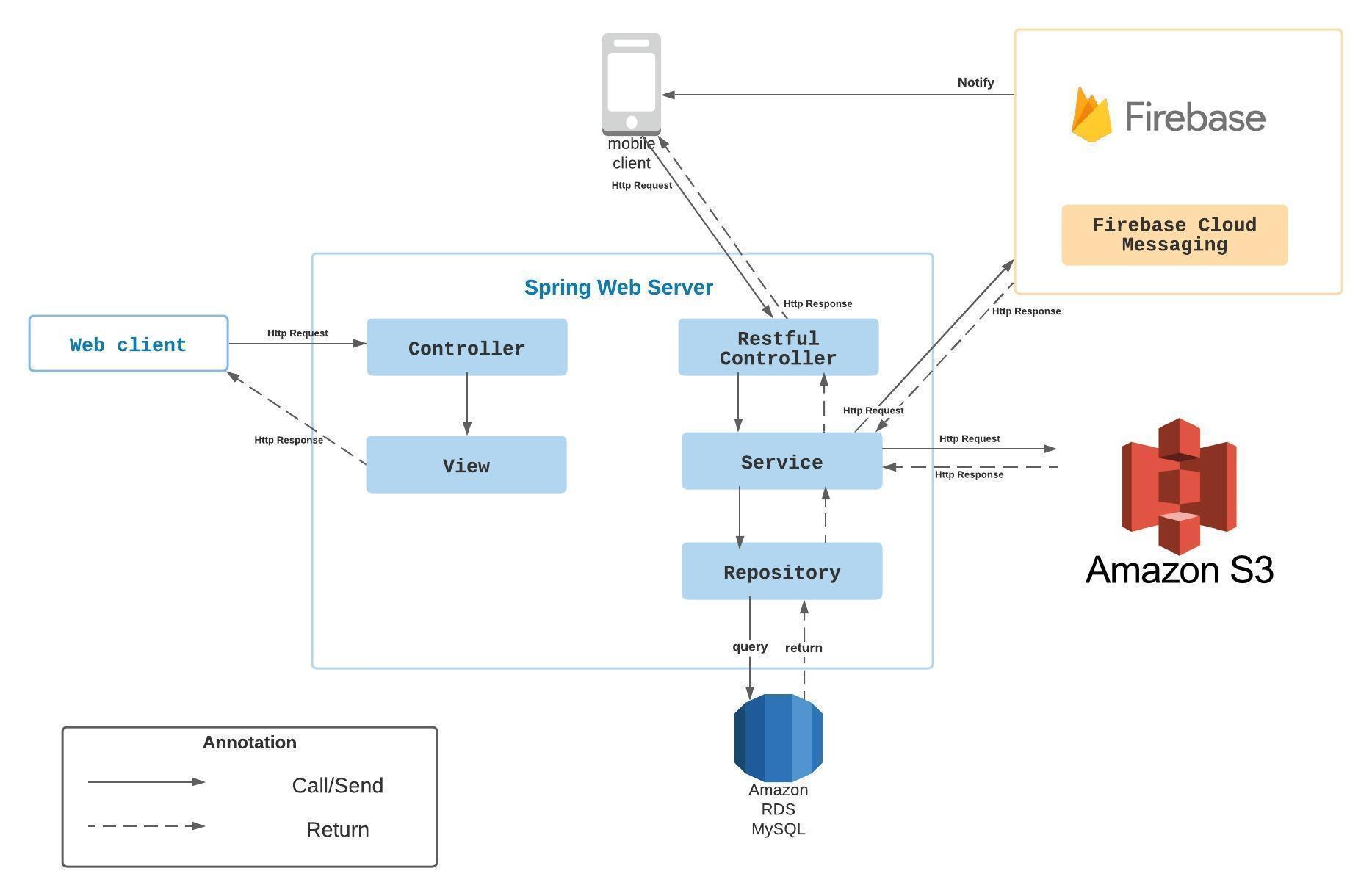
Assuming all the users for this system have basic mobile knowledge and knows how to operate the system smoothly.

### 1.2 Design Constraints

Hard drive space: The system will need at least 8GB storage space and minimum 1GB Storage space in order to run the booking system.

## 2. System Architecture Design

### 2.1 Overall Architecture



*Figure 1 Overall Architecture*

PBS system consists of three components, the spring boot web server provides APIs, two mobile applications for customers and photographers, and the web application for admin roles.

We also use several third-party services such as Aws S3 for image storage, Firebase cloud messaging for push notification functions and Amazon RDS for MySQL instances.

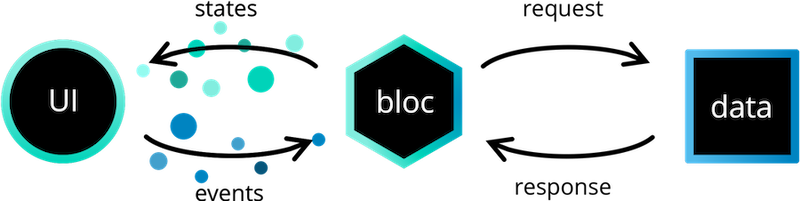
### 2.2 System Architecture

**2.2.1 Web server architecture**

We build our web server following n-tier architecture. We decided to use this pattern for several reasons:

* It has been proven to be a scalable architecture model.
* Provide reusability, maintainability and fault tolerance.
* Helps our members to create reusable and flexible functionality.

**2.2.2 Mobile app architecture**



*Figure 2 Mobile App Architecture*

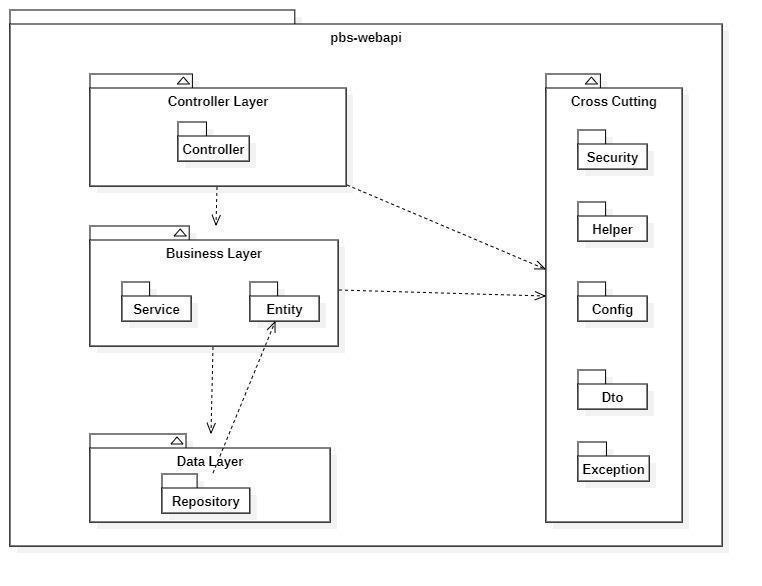
*Reference:* [*https://bloclibrary.dev/#/architecture*](https://bloclibrary.dev/#/architecture)

Bloc pattern allow us to separate our mobile app into 3 layers:

* Presentation: UI
* Business Logic Component: bloc (where we can manage events – click, tap, etc. - send a request to data layer, then retrieve data from data layer and response to input from UI through states).
* Data Layer: Repository (where we can retrieve data from sources).

### 2.3 Package Diagram

**2.3.1 Webapi**

****

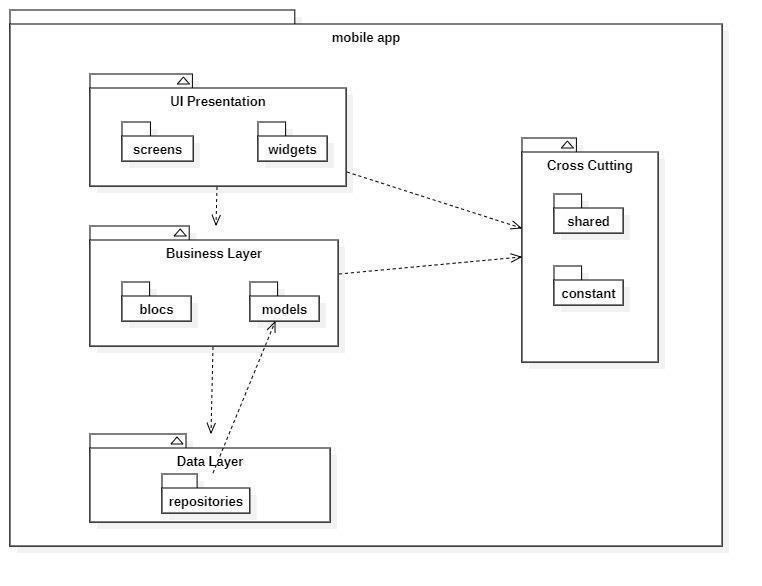
*Figure 3 Webapi Package Diagram*

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 01 | Controller | Package contains classes responsible for handling Http request and return responses with values |
| 02 | Service | Package contains classes responsible for handling business logic, validation and communicate with repository |
| 03 | Entity | Package contains classes represents the business logic entities, also responsible for code first approach with hibernate |
| 04 | Repository | Package contains classes responsible for executing query and communicate with database |
| 05 | Security | Package contains classes responsible for performing security includes jwt handling, authentication and authorization |
| 06 | Helper | Package contains classes responsible for utilizing like map helper, datetime helper, String helper |
| 07 | Config | Package contains classes responsible for configuring spring boot app, from security, application properties to firebase notification |
| 08 | Exception | Package contains classes responsible for handling exception, including status code mapping or error messages returning |
| 09 | Dto | Package contains classes responsible for sharping the data to return to clients |

*Table 1 Package Diagram Description*

**2.3.2 Mobile App**

Both mobile apps share a same project structuring, so we use one diagram to represents both of them here:



*Figure 4 Package Diagram – Mobile app*

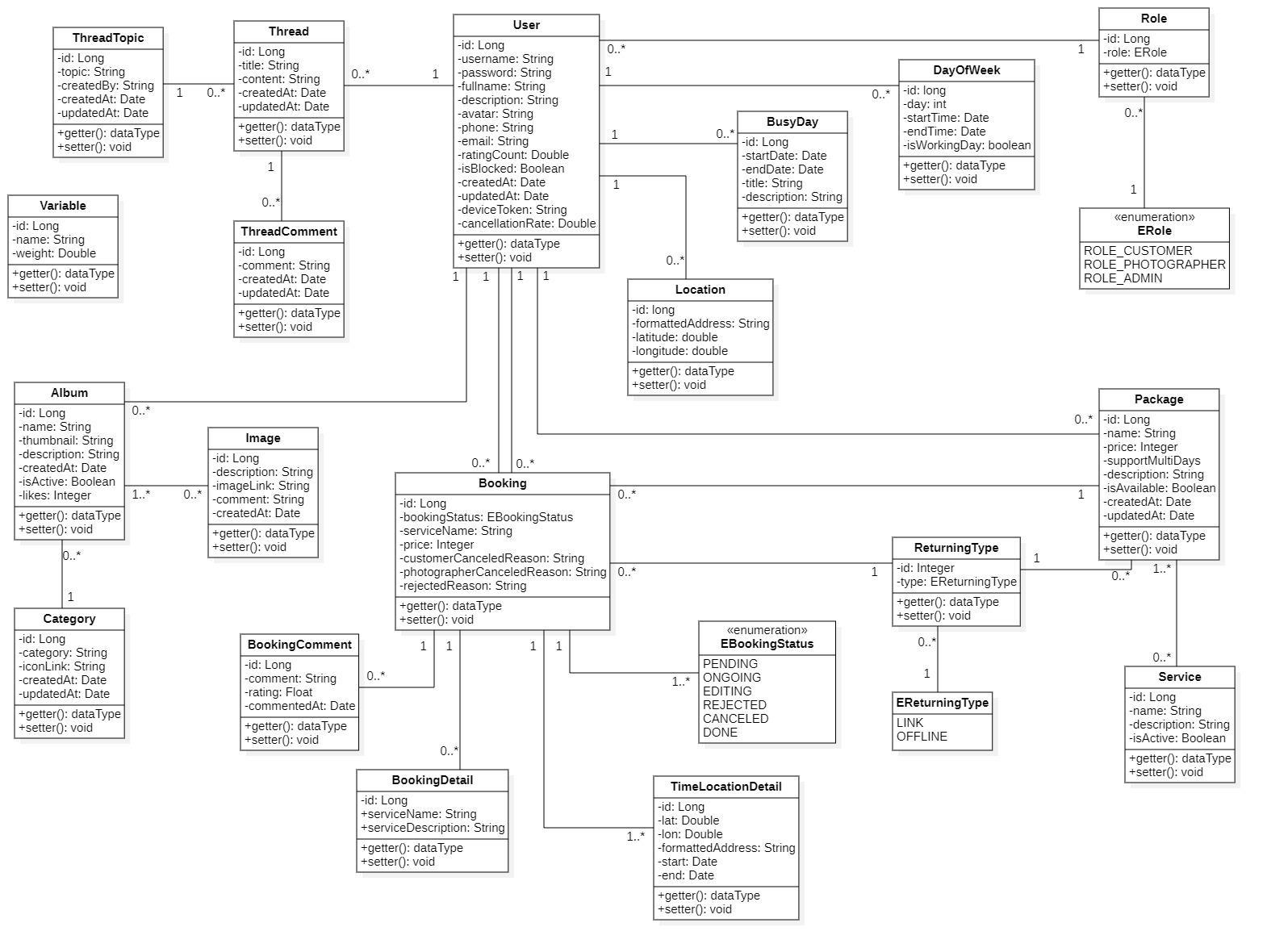
|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 01 | screens | Package contains classes for UI representation |
| 02 | widgets | Package contains classes to form the widgets used by screens |
| 03 | blocs | Package contains classes to implement the bloc architecture including bloc’s state, bloc’s event |
| 04 | models | Package contains classes to form an object extracted from api |
| 05 | repositories | Package contains classes responsible for communicating with api server |
| 06 | shared | Package contains classes responsible for utilizing and environment variables |
| 07 | constant | Package contains classes represents the constants used by application |

*Table 2 Package Diagram Description*

## 3. System Detailed Design

### 3.1 Overall

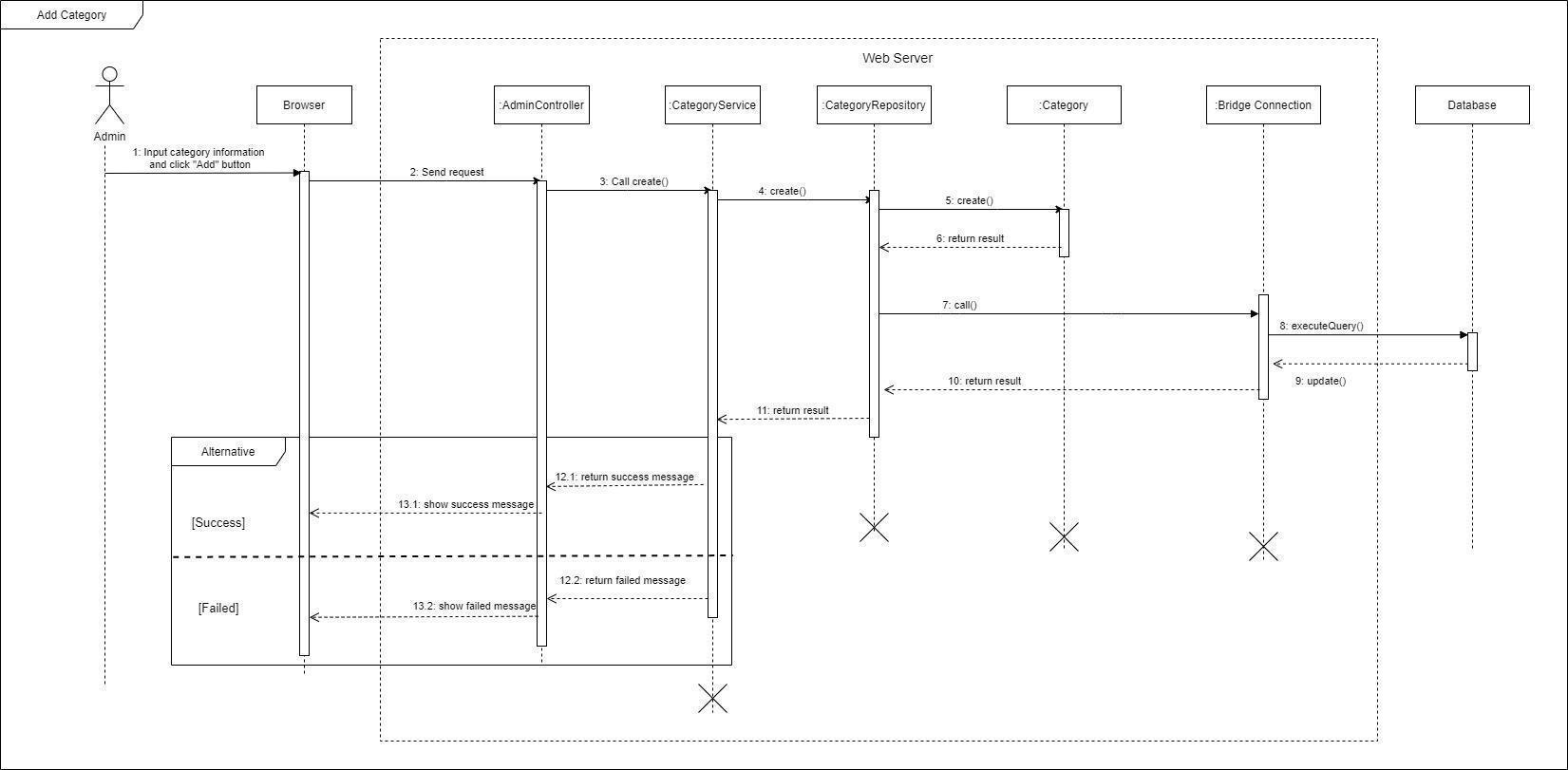
#### a. Class Diagram



*Figure 5 Class Diagram*

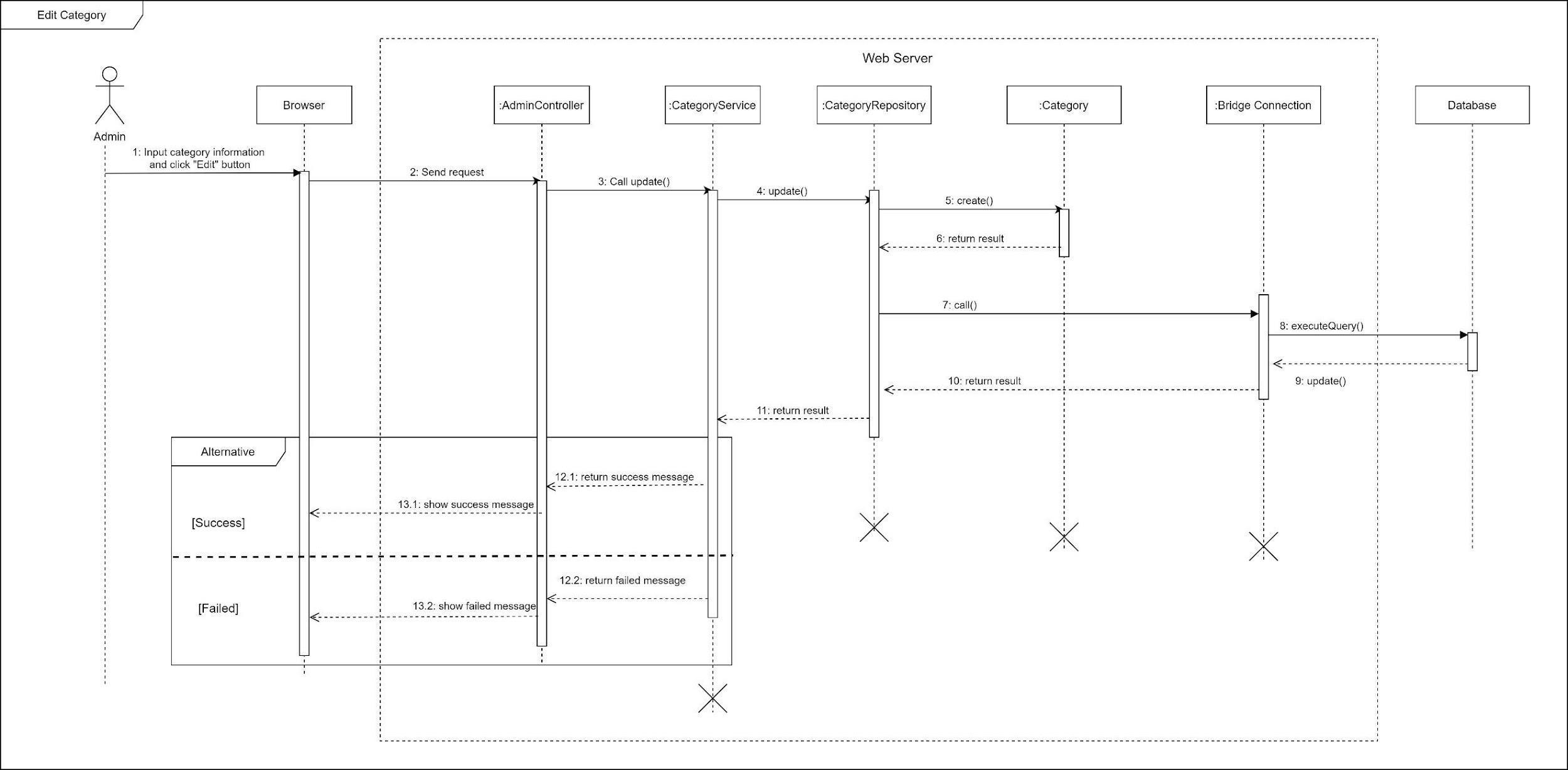
#### b. Sequence Diagram(s)

1. Add Category



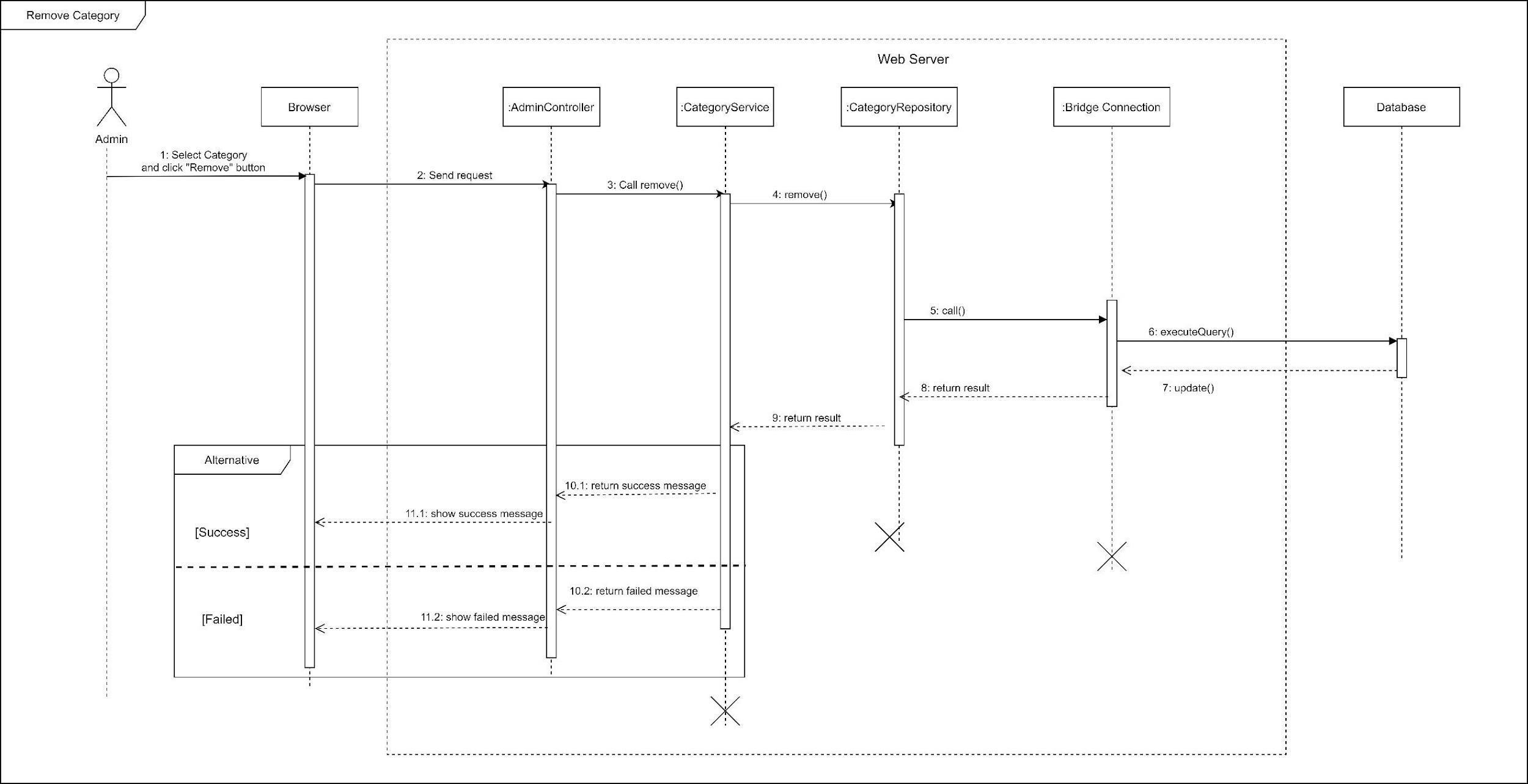
*Figure 6 Sequence Diagram - Add Category*

1. Edit Category



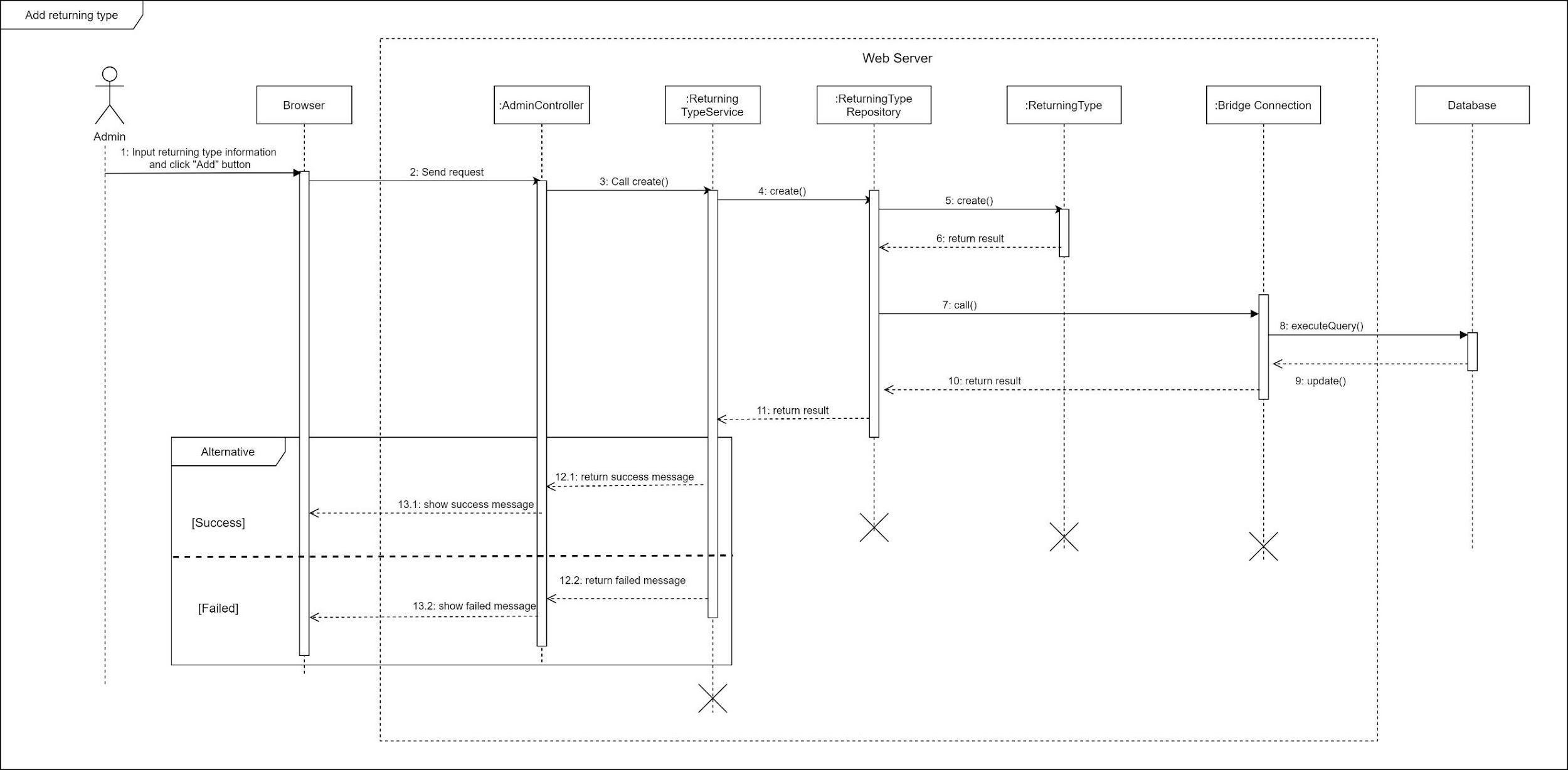
*Figure 7 Sequence Diagram - Edit Category*

1. Remove category



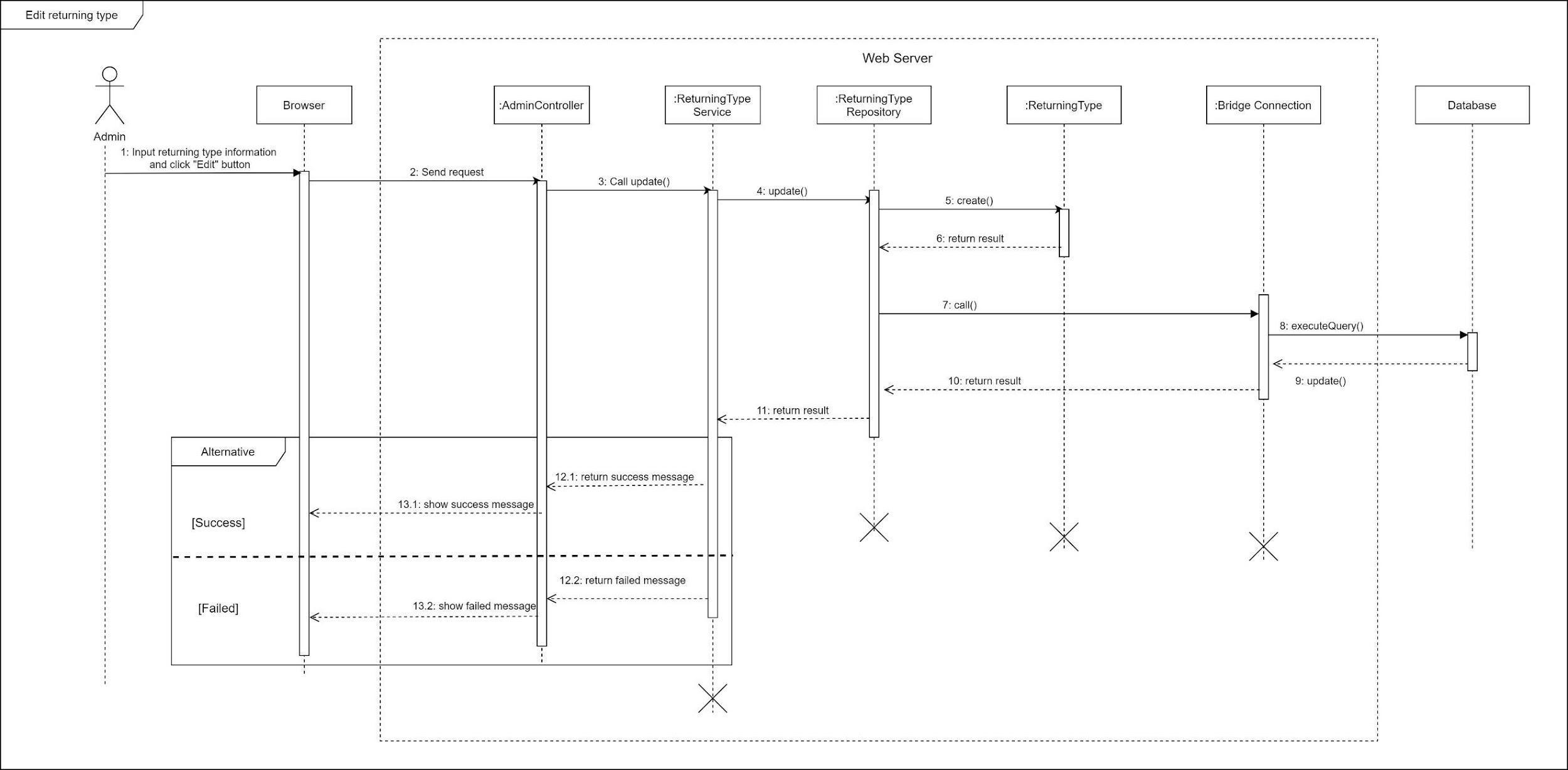
*Figure 8 Sequence Diagram - Remove Category*

1. Add returning type



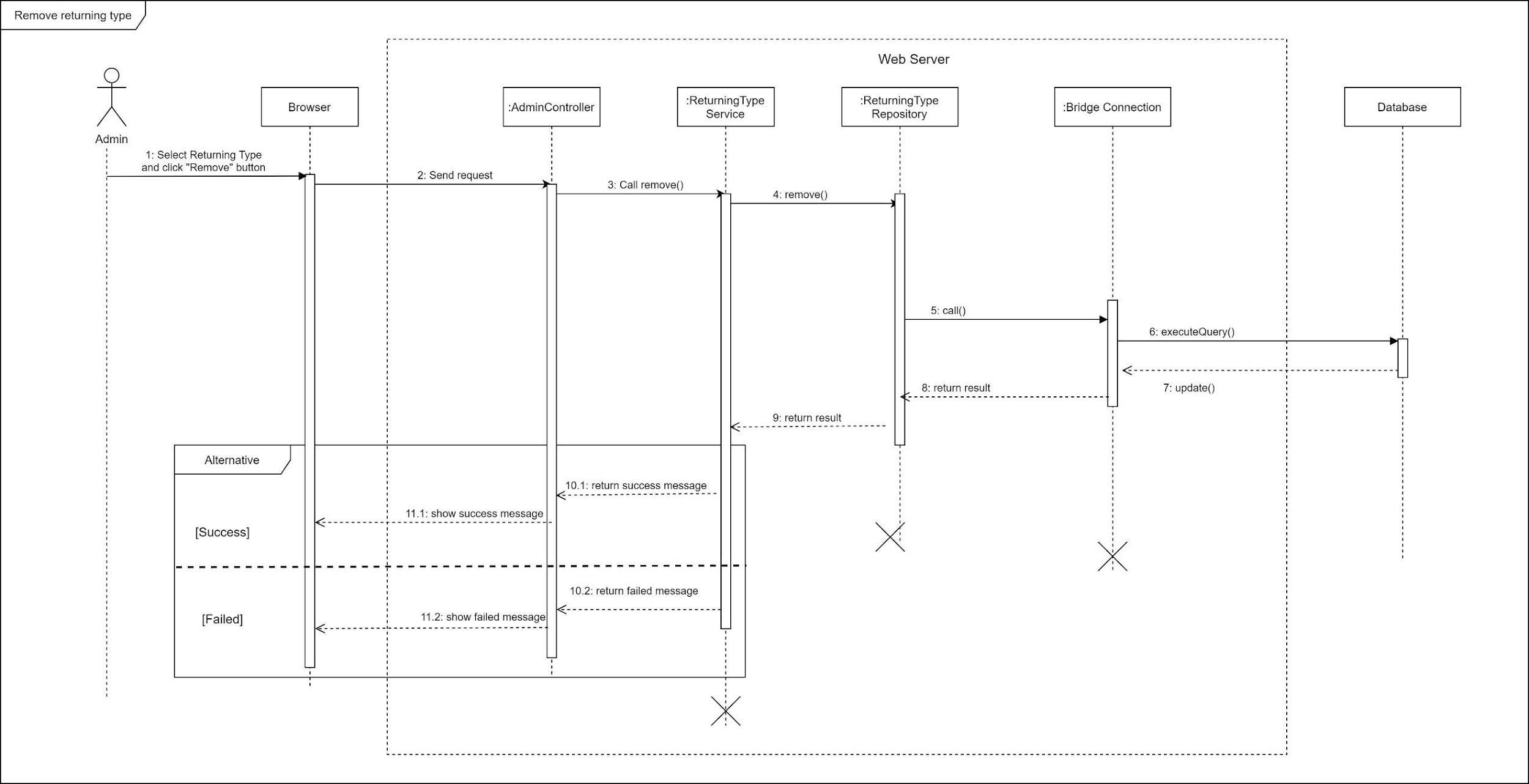
*Figure 9 Sequence Diagram - Add returning type*

1. Edit returning type



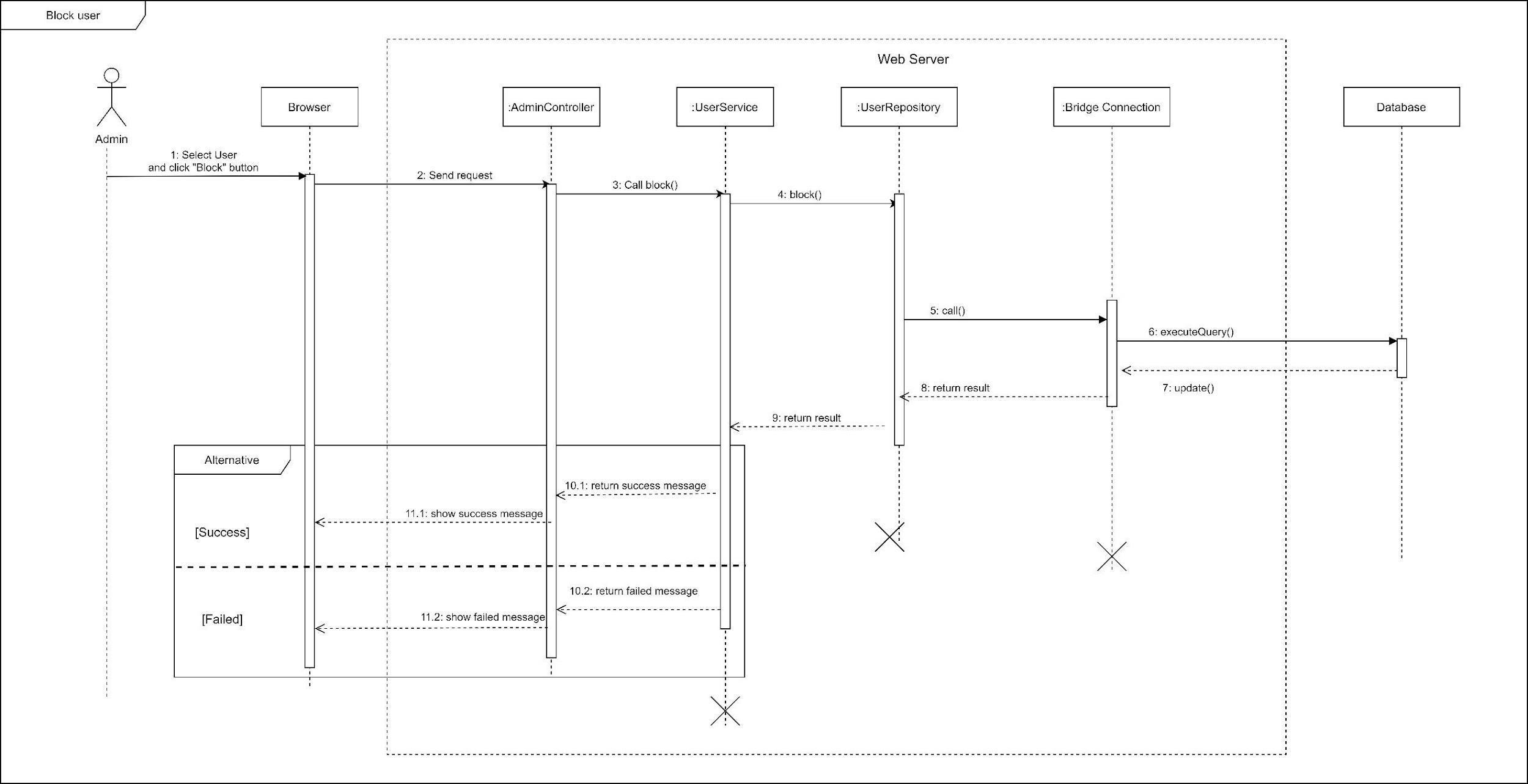
*Figure 10 Sequence Diagram - Edit returning type*

1. Remove returning type



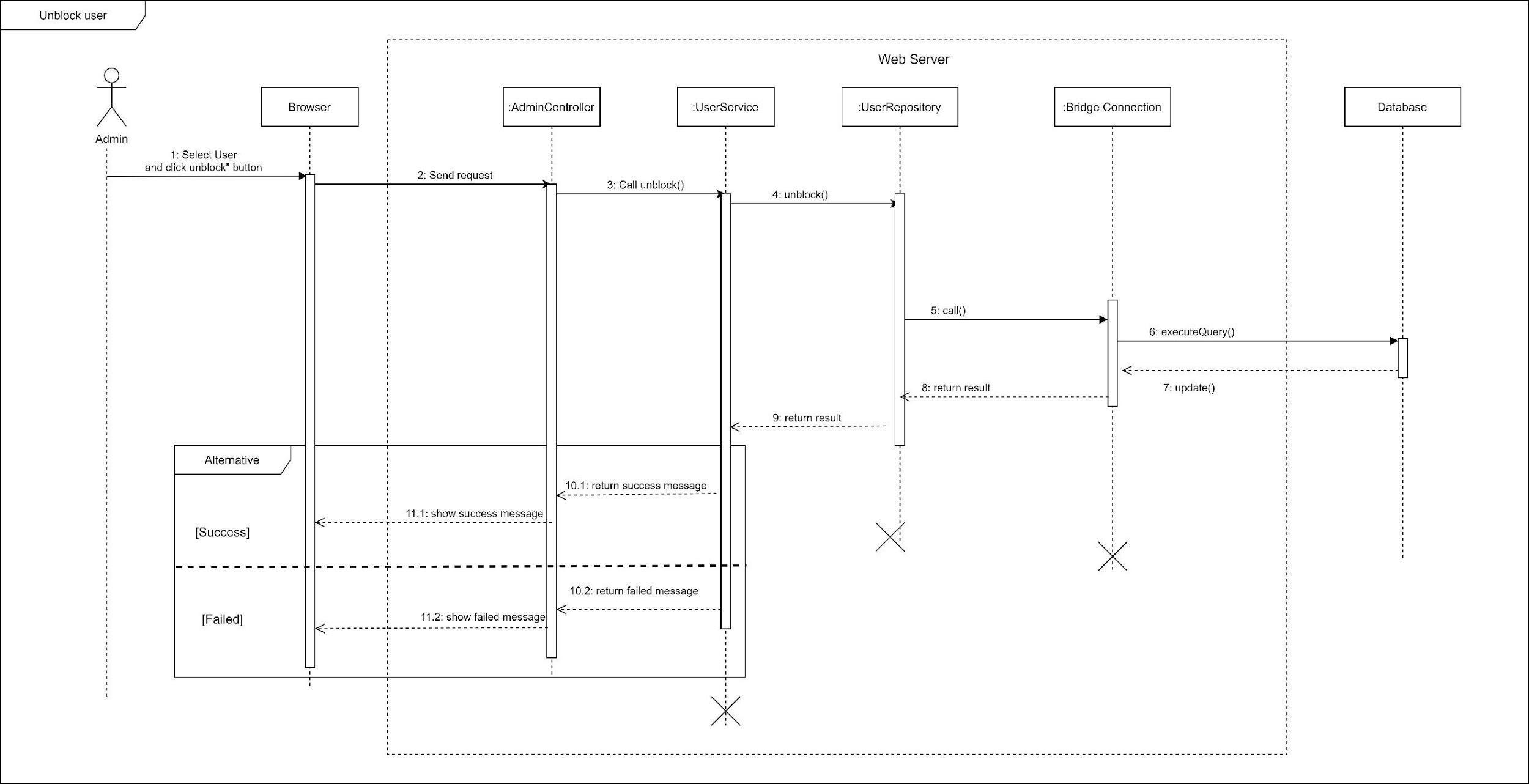
*Figure 11 Sequence Diagram - Remove returning type*

1. Block user



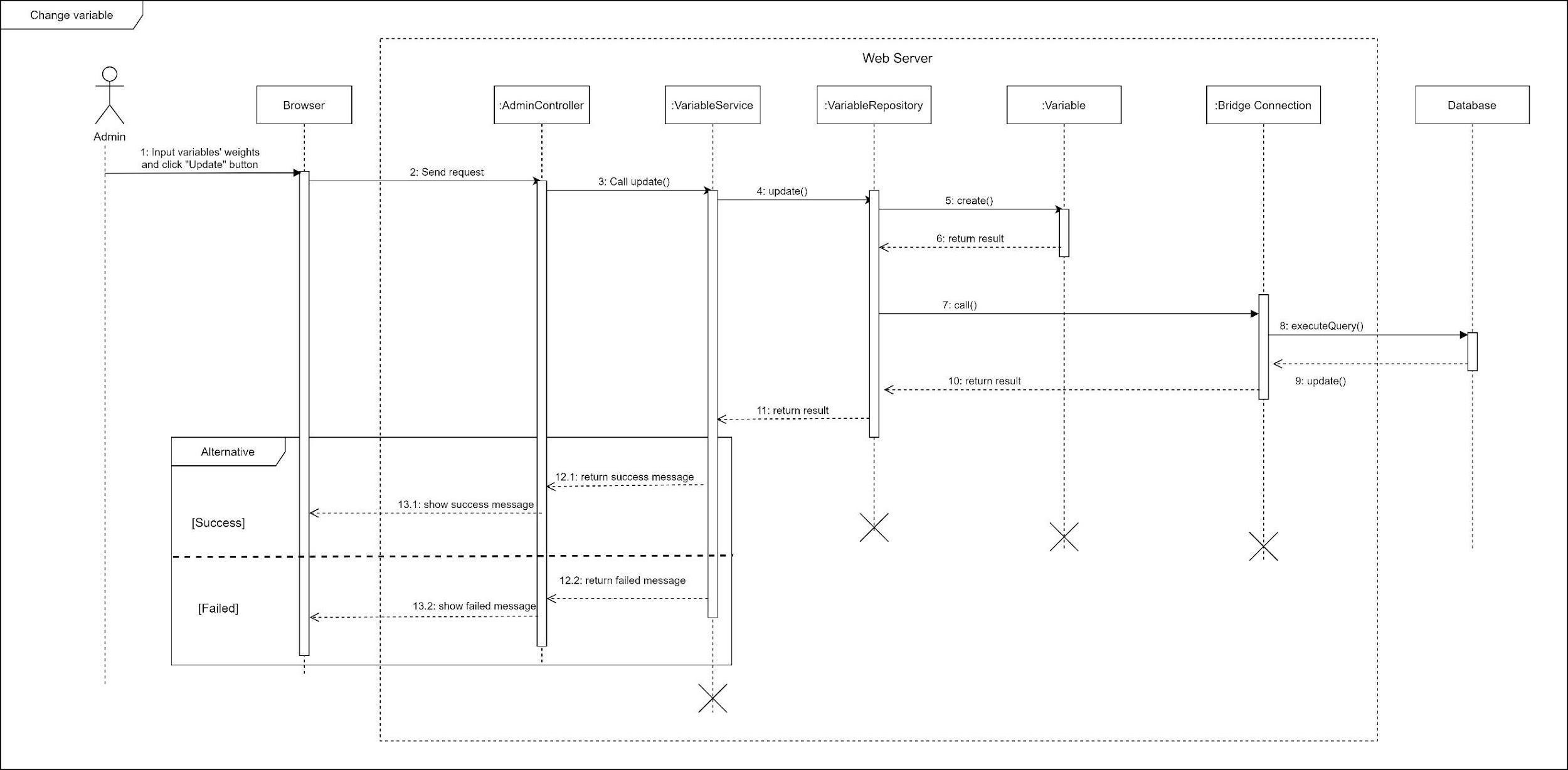
*Figure 12 Sequence Diagram - Block User*

1. Unblock user



*Figure 13 Sequence Diagram - Unblock User*

1. Change variable

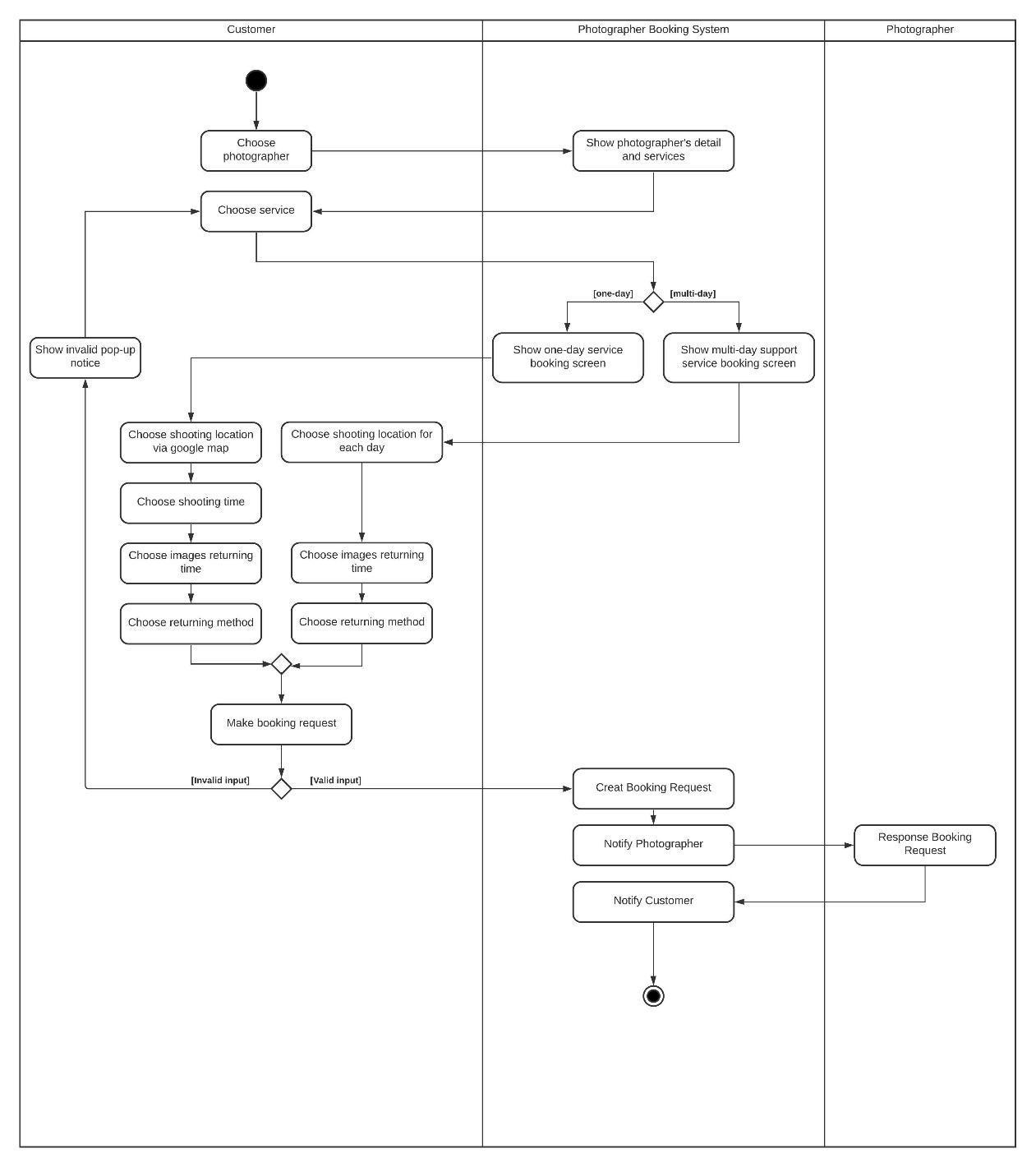


*Figure 14 Sequence Diagram - Change Variables*

#### c. Activity Diagram(s)

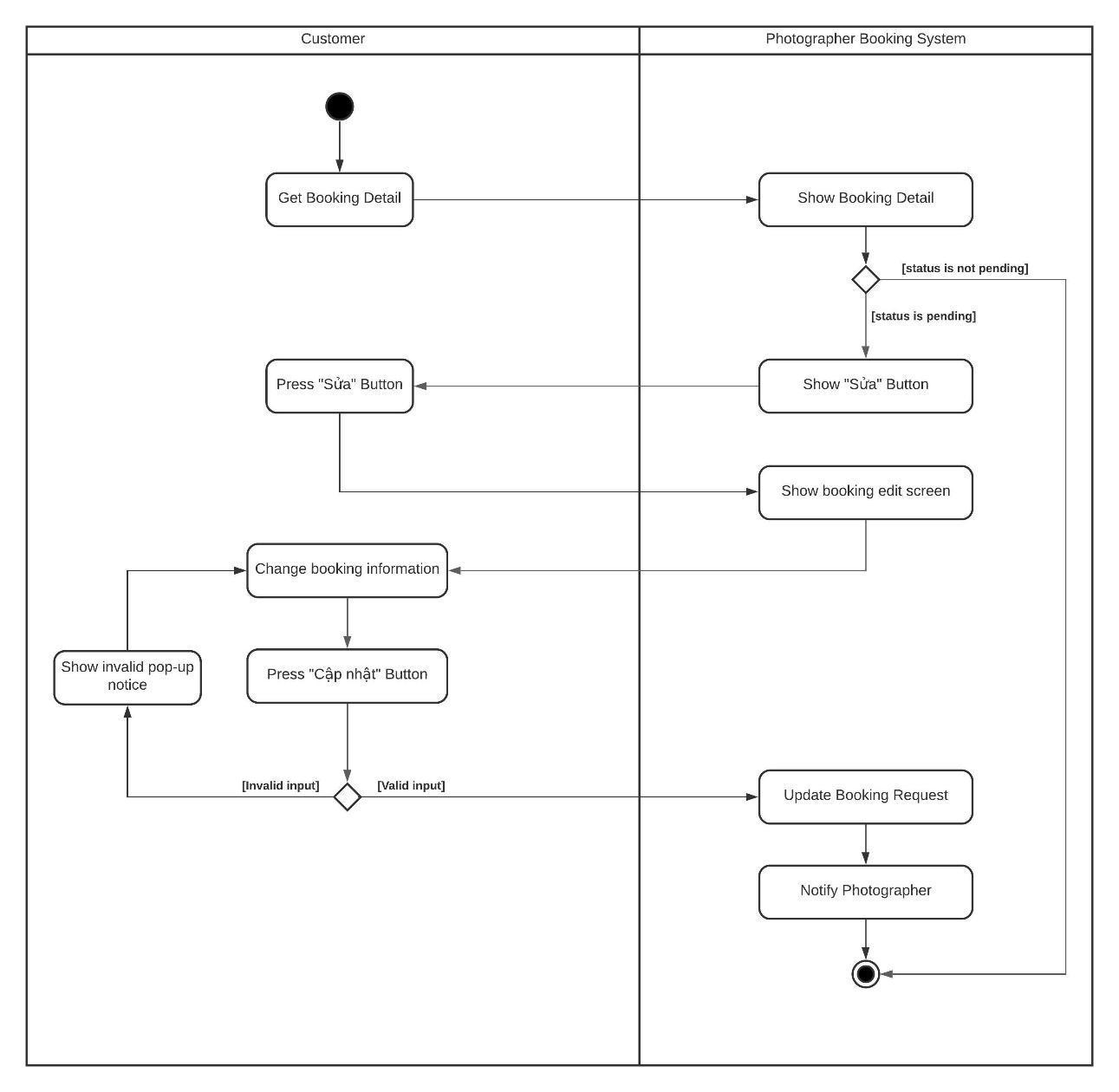
1. **<Customer>** **Make request**

Summary: This diagram shows the process when a customer makes booking request.



1. **<Customer> Edit request**

Summary: This diagram shows the process when a customer edits a request.

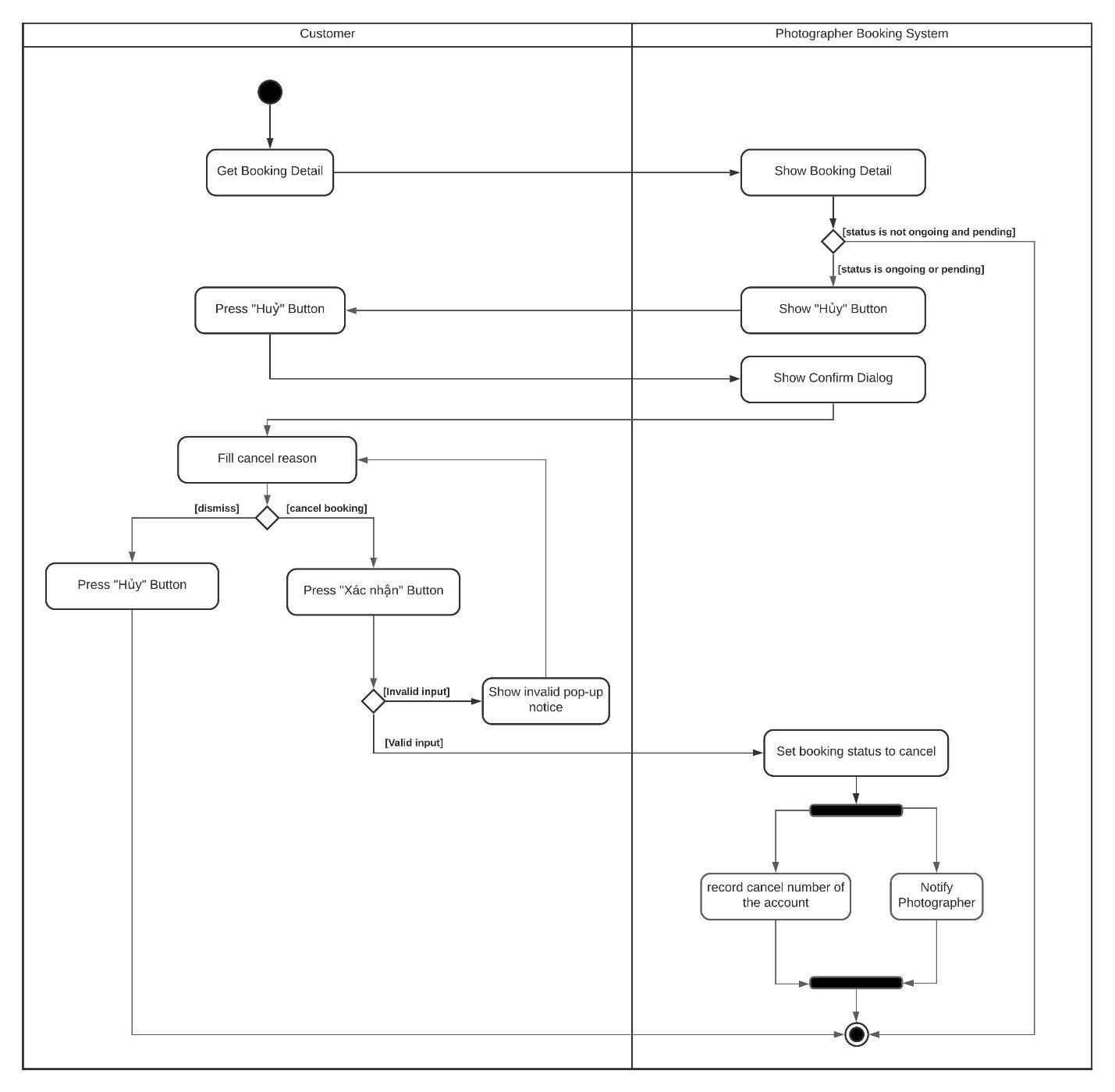


*Figure 16 Activity Diagram – Edit Request*



1. **<Customer> Cancel booking**

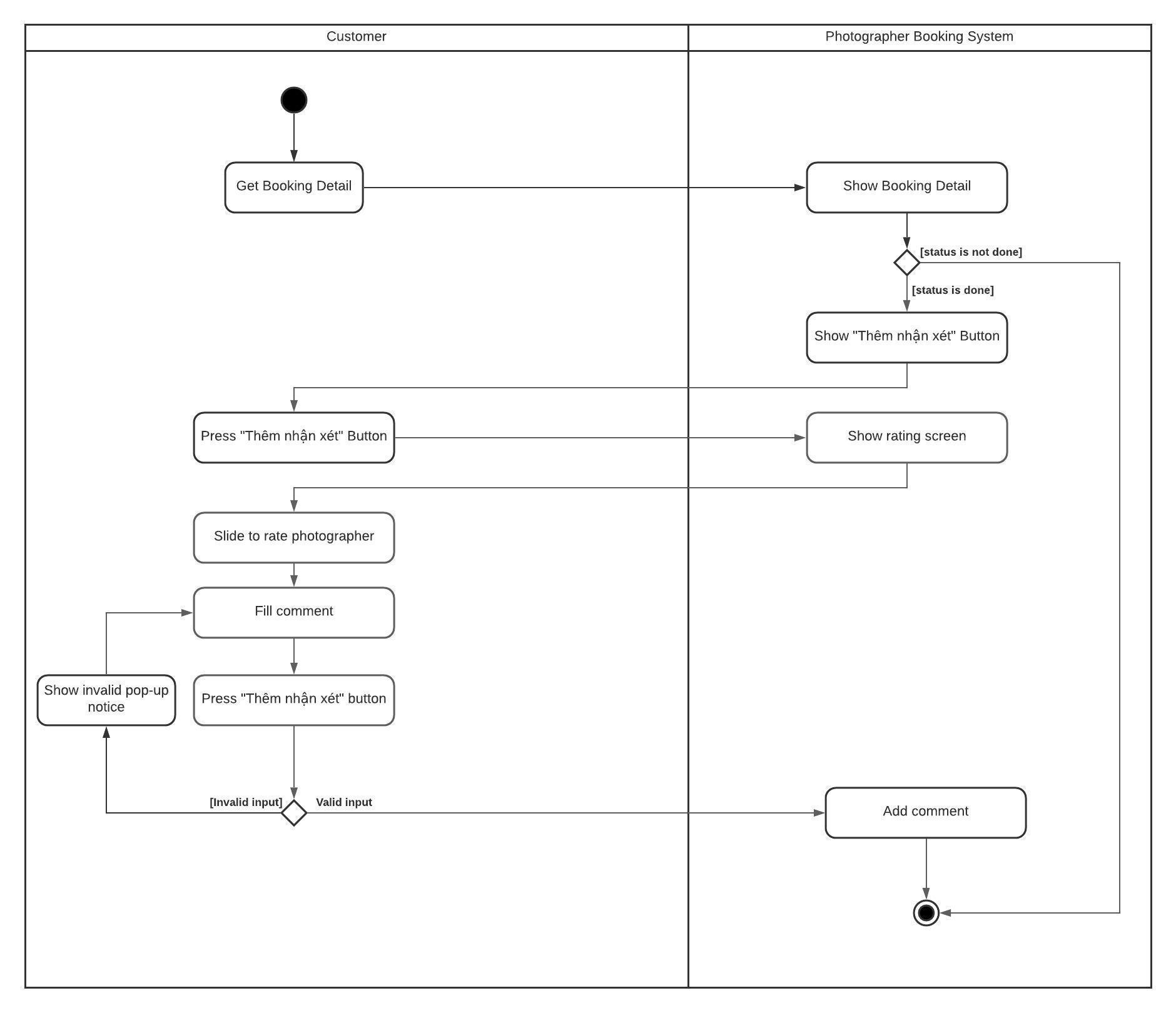
Summary: This diagram shows the process when a customer cancels a request.



*Figure 17 Activity Diagram - Cancel booking*

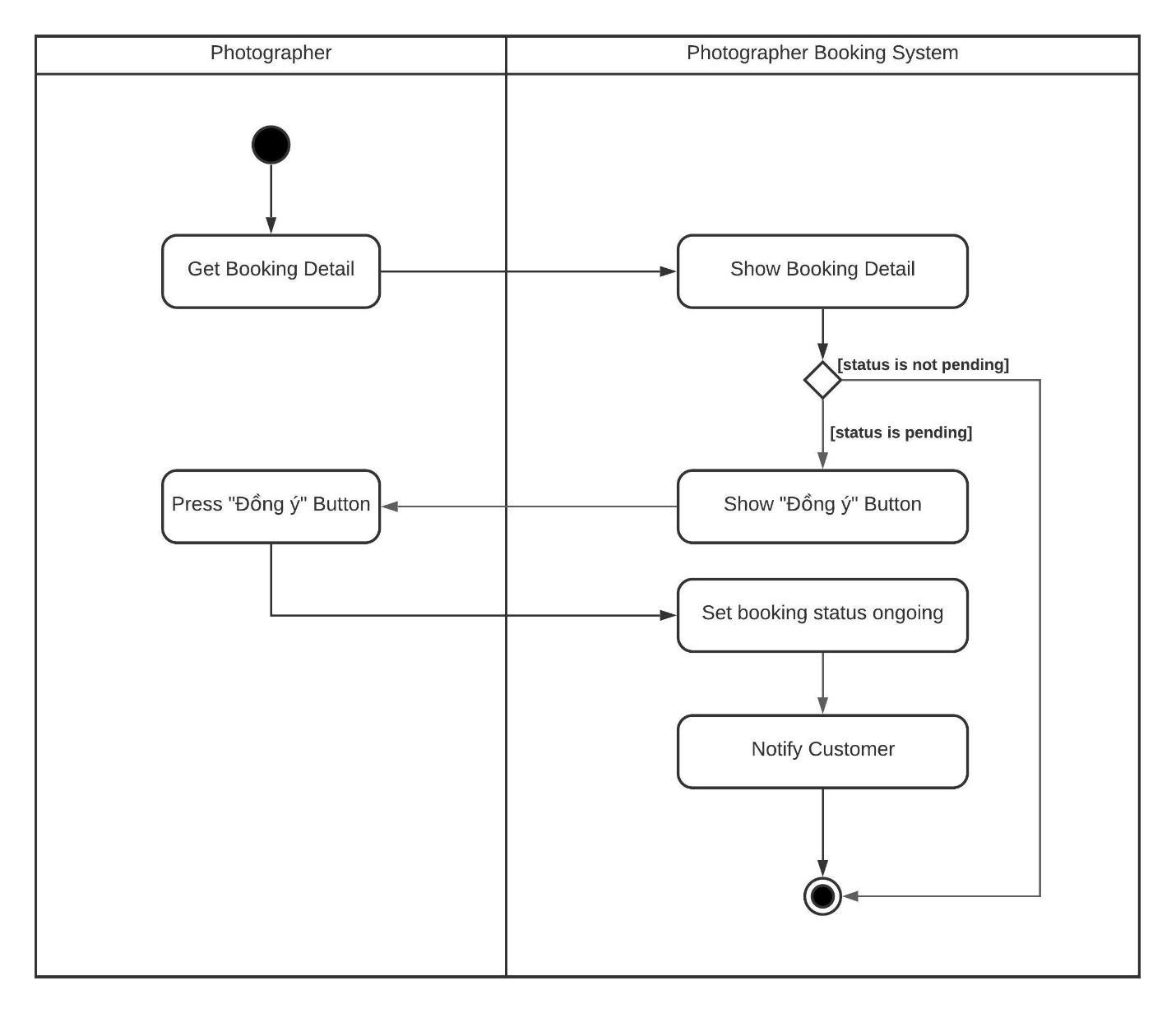
1. **<Customer> Rating**

Summary: This diagram shows the process when a customer rates a photographer after booking finishes.



1. **<Photographer> Accept booking**

Summary: This diagram shows the process when a photographer accepts a request.

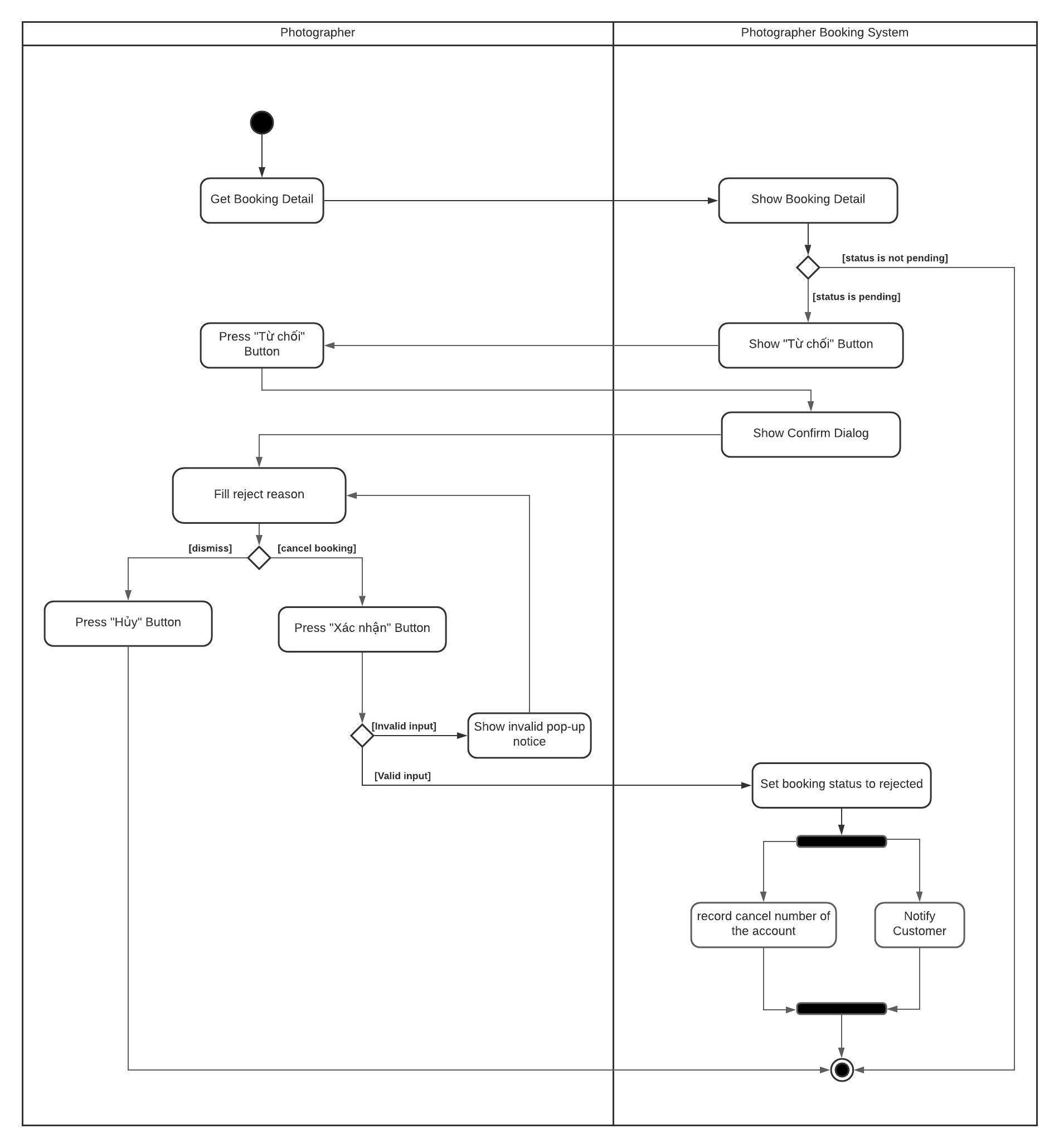


*Figure 19 Activity Diagram - Accept booking*



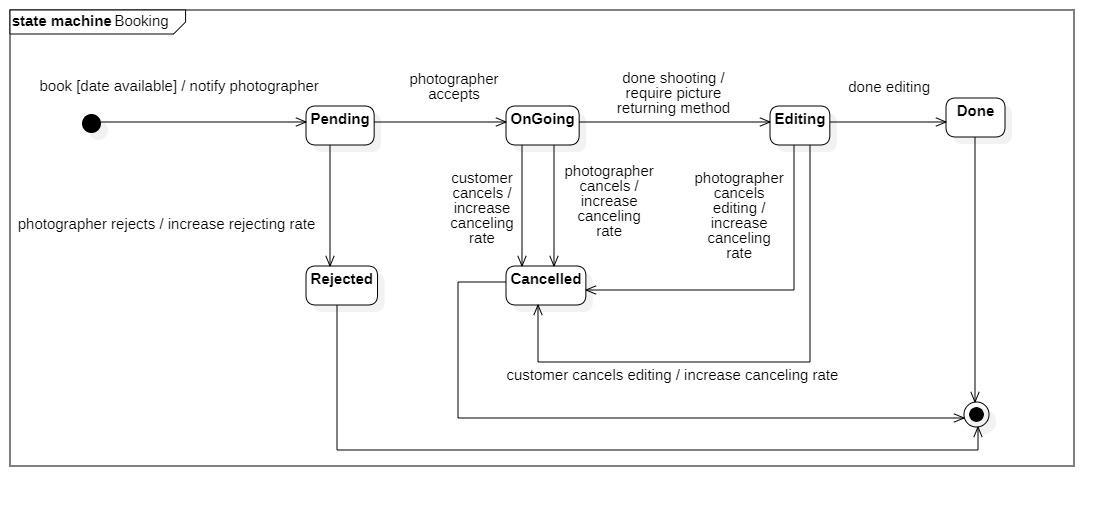
1. **<Photographer> Reject booking**

Summary: This diagram shows the process when a photographer rejects a request.



#### d. State Machine Diagram(s)

This state machine diagram describes states of a booking from start to done.



*Figure 21 State Machine Diagram - Booking*

## 4. Class Specifications

### 4.1 User

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of a user |
| username | String | private | The attribute represents username of a user |
| password | String | private | The attribute represents password of a user |
| fullname | String | private | The attribute represents full name of a user |
| description | String | private | The attribute represents user self-description |
| avatar | String | private | The attribute represents avatar link of a user |
| phone | String | private | The attribute represents phone of a user |
| email | String | private | The attribute represents email of an user |
| ratingCount | Float | private | The attribute represents the rating count of a user whose role is photographer |
| isBlocked | Boolean | private | The attribute represents if user has been blocked |
| createdAt | Date | private | The attribute represents created date of a user object |
| updatedAt | Date | private | The attribute represents updated date of a user object |
| deviceToken | String | private | The attribute represents user’s firebase device token |
| cancellationRate | Double | private | The attribute represents user’s cancellation rate |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 3 Class Diagram - User*

### 4.2 Role

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of a role |
| role | ERole | private | The attribute represents role name |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 4 Class Diagram - Role*

### 4.3 ERole

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ROLE\_CUSTOMER |  |  | Enum attribute |
| ROLE\_PHOTOGRAPHER |  |  | Enum attribute |
| ROLE\_ADMIN |  |  | Enum attribute |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 5 Class Diagram - ERole*

### 4.4 Package

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | private | Unique identifier of a package |
| name | String | private | The attribute represents the package name |
| price | Integer | private | The attribute represents the package price |
| supportMultiDays | Boolean | private | The attribute represents if this package support multi days deal |
| description | String | private | The attribute represents the package description |
| isAvailable | Boolean | private | The attribute represents if the package still available |
| createdAt | Date | private | The attribute represents created date of a package object |
| updatedAt | Date | private | The attribute represents updated date of a package object |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 6 Class Diagram - Package*

### 4.5 Service

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of a package |
| name | String | private | The attribute represents the name of a service |
| description | String | private | The attribute represents the service’s description |
| isActive | Boolean | private | The attribute represents if the service is still available |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 7 Class Diagram - Service*

### 4.6 Booking

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of a booking |
| bookingStatus | EBookingStatus | private | The attribute represents the status of a booking |
| serviceName | String | private | The attribute represents the service of a booking |
| price | Integer | private | The attribute represents the service price of a booking |
| customerCancelledReason | String | private | The attribute represents the reason if customer decided to cancel a booking |
| photographerCancelledReason | String | private | The attribute represents the reason if photographer decided to cancel a booking |
| rejectedReason | String | private | The attribute represents the reason if photographer decided to reject a booking |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 8 Class Diagram - Booking*

### 4.7 EBookingStatus

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Pending |  |  | Enum attribute |
| OnGoing |  |  | Enum attribute |
| Editing |  |  | Enum attribute |
| Rejected |  |  | Enum attribute |
| Cancelled |  |  | Enum attribute |
| Done |  |  | Enum attribute |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 9 Class Diagram - EBookingStatus*

### 4.8 Album

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of album |
| name | String | private | The attribute represents the name album |
| thumbnail | String | private | The attribute represents the album’s thumbnail link |
| description | String | private | The attribute represents the album’s description |
| createdAt | Date | private | The attribute represents the created date of album object |
| isActive | Boolean | private | The attribute represents if album still active |
| likes | Integer | private | The attribute represents the album like count |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 10 Class Diagram - Album*

### 4.9 Image

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of image |
| description | String | private | The attribute represents the description of album |
| imageLink | String | private | The attribute represents the album’s image link |
| comment | String | private | The attribute represents the album’s comment |
| createdAt | Date | private | The attribute represents the created date of image object |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 11 Class Diagram - Image*

### 4.10 Category

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of category |
| category | String | private | The attribute represents the category name |
| iconLink | String | private | The attribute represents the category icon link |
| createdAt | Date | private | The attribute represents the created date of category object |
| updatedAt | Date | private | The attribute represents the updated date of category object |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 12 Class Diagram - Category*

### 4.11 DayOfWeek

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of day of week |
| day | Int | private | The attribute represents the day of the week |
| startTime | Date | private | The attribute represents the start time of photographer’s working time |
| endTime | Date | private | The attribute represents the end time of photographer’s working time |
| isWorkingDay | Boolean | private | The attribute represents if this day of week is working day |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 13 Class Diagram - DayOfWeek*

### 4.12 Location

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of location |
| formattedAddress | String | private | The attribute represents the location full address |
| longitude | String | private | The attribute represents the longitude of a location |
| latitude | Double | private | The attribute represents the latitude of a location |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 14 Class Diagram - Location*

### 4.13 BookingComment

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of booking comment |
| comment | String | private | The attribute represents the comment of user |
| rating | Float | private | The attribute represents the rating of user |
| commentedAt | Date | private | The attribute represents the commented date of a booking |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 15 Class Diagram - BookingComment*

### 4.14 BookingDetail

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of booking detail |
| serviceName | String | private | The attribute represents the booked service’s name |
| serviceDescription | String | private | The attribute represents the booked service’s description |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 16 Class Diagram - BookingDetail*

### 4.15 BusyDay

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of busy day |
| startDate | Date | private | The attribute represents the start date of photographer’s busy event |
| endDate | Date | private | The attribute represents the end date of photographer’s busy event |
| title | String | private | The attribute represents the title of photographer’s busy event |
| description | String | private | The attribute represents the description of this busy event |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 17 Class Diagram - BusyDay*

### 4.16 Thread

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of thread |
| title | String | private | The attribute represents the title of thread |
| content | String | private | The attribute represents the content of a thread |
| createdAt | Date | private | The attribute represents the created date of a thread |
| updatedAt | Date | private | The attribute represents the updated date of a thread |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 18 Class Diagram - Thread*

### 4.17 ThreadTopic

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of thread topic |
| topic | String | private | The attribute represents the topic name of a thread |
| createdBy | String | private | The attribute represents the name of the person who created this topic |
| createdAt | Date | private | The attribute represents the created date of a thread |
| updateAt | Date | private | The attribute represents the updated date of a thread |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 19 Class Diagram - ThreadTopic*

### 4.18 ThreadComment

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of thread comment |
| comment | String | private | The attribute represents the comment of a thread |
| createdAt | Date | private | The attribute represents the created date of this record |
| updateAt | Date | private | The attribute represents the updated date of this record |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 20 Class Diagram - ThreadComment*

### 4.19 ReturningType

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of returning type |
| type | EReturningType | private | The attribute represents the type |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 21 Class Diagram - ReturningType*

### 4.20 EReturningType

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| LINK |  |  | Enum attribute |
| OFFLINE |  |  | Enum attribute |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 22 Class Diagram - EReturningType*

### 4.21 TimeLocationDetail

This table used for saving time and location details of a booking

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of time location detail |
| lat | Double | private | The attribute represents the latitude of a user |
| lon | Double | private | The attribute represents the longitude of a user |
| formattedAddress | String | private | The attribute represents the formatted address of a user |
| start | Date | private | The attribute represents start time of a booking |
| end | Date | private | The attribute represents end time of a booking |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 23 Class Diagram – TimeLocationDetail*

### 4.22 Variable

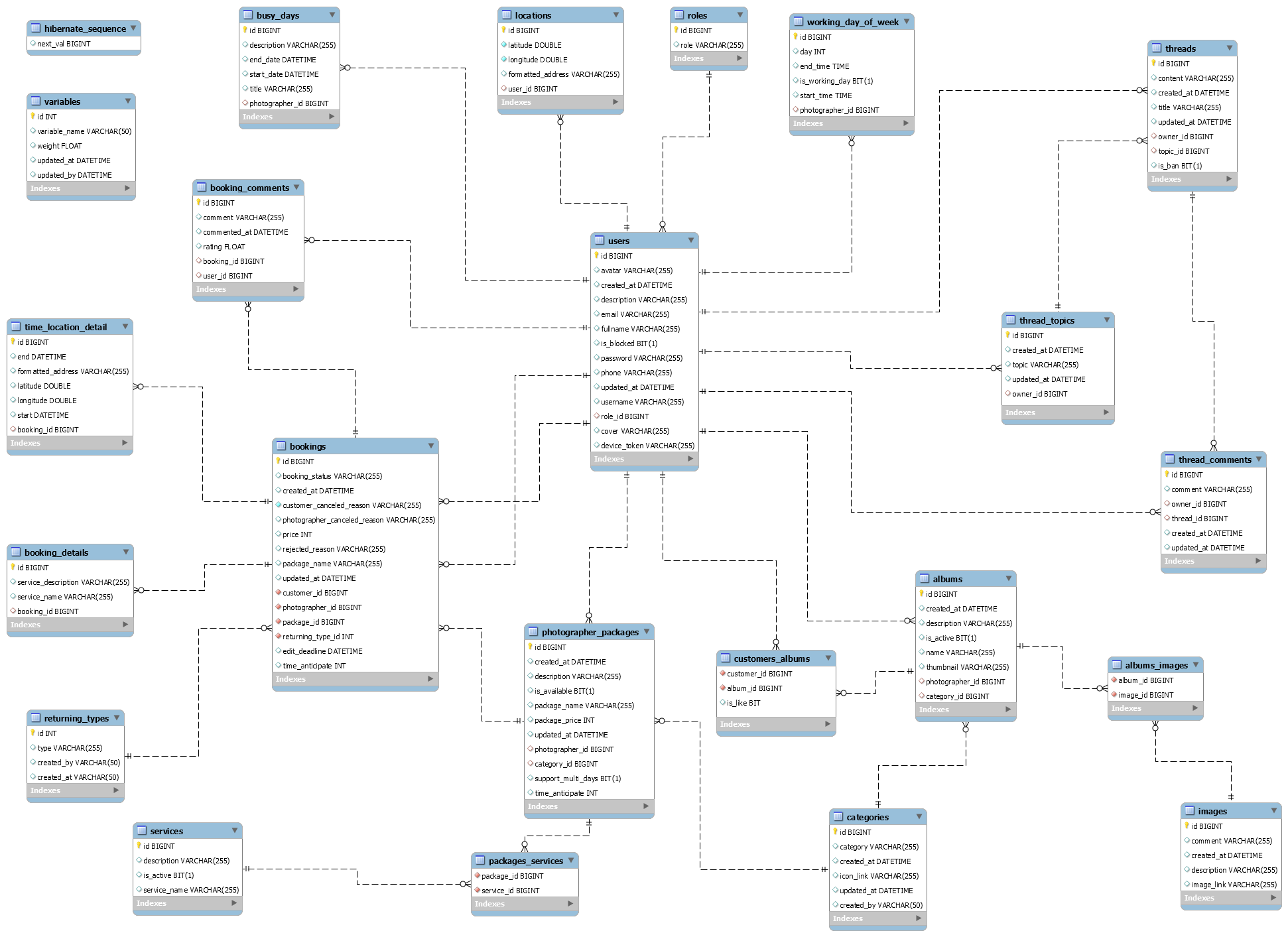
|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | private | Unique identifier of variable |
| name | String | private | The attribute represents the name of the variable |
| weight | Double | private | The attribute represents the weight of the variable |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Method** | **Visibility** | **Return Type** | **Description** |
| 1 | getter | public | Attribute Type | get attribute value |
| 2 | setter | public | void | set attribute value |

*Table 24 Class Diagram - Variable*

## 5. Data & Database Design

### 5.1 Database Design



*Figure 22 Database Design*

#### users

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | username | varchar | 255 | X | X |  |  |
| 3 | password | varchar | 255 |  | X |  |  |
| 4 | fullname | varchar | 255 |  | X |  |  |
| 5 | description | varchar | 255 |  |  |  |  |
| 6 | avatar | varchar | 255 |  |  |  | Avatar image link of user |
| 7 | phone | varchar | 255 |  |  |  |  |
| 8 | email | varchar | 255 |  |  |  |  |
| 9 | device\_token | varchar | 255 |  |  |  | The firebase token of device, using for firebase messaging. |
| 9 | is\_blocked | bit |  |  |  |  |  |
| 10 | is\_active | bit |  |  |  |  |  |
| 11 | created\_at | datetime |  |  |  |  |  |
| 12 | updated\_at | datetime |  |  |  |  |  |
| 13 | role\_id | bigint |  |  | X | X |  |
| 14 | cover | varchar | 255 |  |  |  |  |

*Table 25 Database - users*

#### roles

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | int |  | X | X | X |  |
| 2 | role | varchar | 255 |  | X |  |  |

*Table 26 Database - roles*

#### photographer\_packages

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | package\_name | varchar | 255 |  | X |  |  |
| 3 | package\_price | int |  |  | X |  |  |
| 4 | is\_available | bit | 1 |  |  |  |  |
| 5 | description | varchar | 255 |  |  |  |  |
| 6 | photographer \_id | bigint |  |  | X | X |  |
| 7 | category\_id | bigint |  |  | X | X |  |
| 8 | support\_multiple\_days | bit | 1 |  |  |  | This field represents whether or not a package is supporting multiple day booking. |
| 9 | time\_anticipate | int |  |  |  |  | The shooting time of a package (in seconds). |
| 10 | created\_at | datetime |  |  |  |  |  |
| 11 | updated\_at | datetime |  |  |  |  |  |

*Table 27 Database - photographer\_packages*

#### packages\_services

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | package\_id | Int |  |  | X | X |  |
| 2 | service\_id | Int |  |  | X | X |  |

*Table 28 Database - packages\_services*

#### services

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | name | varchar | 255 |  | X |  |  |
| 3 | description | varchar | 255 |  |  |  |  |
| 4 | is\_active | bit | 1 |  |  |  |  |

*Table 29 Database - services*

#### bookings

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | booking\_status | varchar | 255 |  | X |  | There will be 6 status: Pending, Ongoing, Editing, Done, Rejected, Canceled |
| 3 | package\_name | varchar | 255 |  | X |  | Package’s name |
| 4 | price | int |  |  | X |  | Package’s price |
| 5 | customer  \_canceled  \_reason | varchar | 255 |  |  |  | In case customers cancel and provide their reason |
| 6 | photographer \_canceled \_reason | varchar | 255 |  |  |  | In case photographers cancel and provide their reason |
| 7 | rejected\_reason | varchar | 255 |  |  |  | In case photographers reject the request and provide reason |
| 8 | customer\_id | bigint |  |  | X | X |  |
| 9 | photographer \_id | bigint |  |  | X | X |  |
| 10 | returning\_type\_id | int |  |  | X | X |  |
| 11 | package\_id | bigint |  |  | X | X |  |
| 12 | edit\_deadline | datetime |  |  |  |  | The datetime customer wants to have their pictures done. |
| 13 | time\_anticipate | int |  |  |  |  | Package’s shooting time (in seconds) |
| 14 | created\_at | datetime |  |  |  |  |  |
| 15 | updated\_at | datetime |  |  |  |  |  |

*Table 30 Database - bookings*

#### albums

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | name | varchar | 255 |  | X |  |  |
| 3 | description | varchar | 255 |  |  |  |  |
| 4 | created\_at | datetime |  |  |  |  |  |
| 5 | thumbnail | varchar | 255 |  |  |  | The thumbnail link of the album. |
| 6 | is\_active | bit | 1 |  |  |  |  |
| 7 | photographer \_id | bigint |  |  | X | X |  |
| 8 | category\_id | bigint |  |  |  | X |  |

*Table 31 Database - albums*

#### locations

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | formatted\_address | varchar | 255 |  | X |  | Location full address. |
| 3 | longitude | double |  |  | X |  |  |
| 4 | latitude | double |  |  | X |  |  |
| 5 | user\_id | bigint |  |  | X | X |  |

*Table 32 Database - locations*

#### categories

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | category | varchar | 255 |  | X |  |  |
| 3 | icon\_link | varchar | 255 |  | X |  | Link for the icon of category |
| 4 | created\_by | varchar | 255 |  |  |  | Name of the admin who created the category |
| 5 | created\_at | datetime |  |  |  |  |  |
| 6 | updated\_at | datetime |  |  |  |  |  |

*Table 33 Database - categories*

#### album\_images

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | album\_id | bigint |  |  | X | X |  |
| 2 | image\_id | bigint |  |  | X | X |  |

*Table 34 Database - album\_images*

#### images

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | image\_link | varchar | 255 |  | X |  | Link of the image. |
| 3 | description | varchar | 255 |  |  |  |  |
| 4 | created\_at | datetime |  |  |  |  |  |
| 5 | comment | varchar | 255 |  | X | X |  |

*Table 35 Database - images*

#### customers\_albums

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | is\_like | bit |  |  |  |  | represent if customer has liked the album |
| 2 | album\_id | bigint |  |  | X | X |  |
| 3 | customer\_id | bigint |  |  | X | X |  |

*Table 36 Database - customer\_albums*

#### working\_day\_of\_week

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | day | int |  |  |  |  | Day represents week day from Monday to Sunday.  1 is Sunday and 7 is Saturday. |
| 3 | start\_time | time |  |  |  |  |  |
| 4 | end\_time | time |  |  |  |  |  |
| 5 | is\_working\_day | bit | 1 |  |  |  |  |
| 6 | photographer \_id | bigint |  |  | X | X |  |

*Table 37 Database - working\_day\_of\_week*

#### busy\_days

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | description | varchar | 255 |  |  |  |  |
| 3 | start\_date | datetime |  |  |  |  |  |
| 4 | end\_date | datetime |  |  |  |  |  |
| 5 | title | varchar | 255 |  |  |  |  |
| 6 | photographer \_id | bigint |  |  | X | X |  |

*Table 38 Database - busy\_days*

#### returning\_types

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | int |  | X | X | X |  |
| 2 | type | varchar | 255 |  | X |  |  |
| 3 | created\_at | datetime |  |  |  |  |  |
| 4 | created\_by | varchar | 255 |  | X |  | Name of the admin who created the type. |

*Table 39 Database - returning\_types*

#### booking\_details

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | service\_name | varchar | 255 |  | X |  |  |
| 3 | service\_description | varchar | 255 |  | X |  |  |
| 4 | booking\_id | bigint |  |  | X | X |  |

*Table 40 Database - booking\_details*

#### time\_location\_details

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | formatted\_address | varchar | 255 |  |  |  | Booking location full address. |
| 3 | latitude | double |  |  |  |  |  |
| 4 | longitude | double |  |  |  |  |  |
| 5 | start | datetime |  |  |  |  | Start time of the booking. |
| 6 | end | datetime |  |  |  |  | End time of the booking. |
| 7 | booking\_id | bigint |  |  | X | X |  |

*Table 41 Database - time\_location\_details*

#### booking\_comments

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | comment | varchar | 255 |  |  |  |  |
| 3 | commented\_at | datetime |  |  |  |  |  |
| 4 | rating | float |  |  |  |  | Rating from 0 to 5. |
| 5 | user\_id | bigint |  |  |  |  |  |
| 6 | booking\_id | bigint |  |  |  |  |  |

*Table 42 Database - booking\_comments*

#### threads

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | title | varchar | 255 |  |  |  |  |
| 3 | content | varchar | 255 |  |  |  |  |
| 4 | update\_at | datetime |  |  |  |  |  |
| 5 | created\_at | datetime |  |  |  |  |  |
| 6 | topic\_id | bigint |  |  | X | X |  |
| 7 | owner\_id | bigint |  |  | X | X |  |
| 8 | is\_ban | bit | 1 |  |  |  |  |

*Table 43 Database - threads*

#### thread\_comments

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | comment | varchar | 255 |  |  |  |  |
| 3 | update\_at | datetime |  |  |  |  |  |
| 4 | created\_at | datetime |  |  |  |  |  |
| 5 | thread\_id | bigint |  |  |  | X |  |
| 6 | owner\_id | bigint |  |  |  | X |  |

*Table 44 Database - thread\_comments*

#### thread\_topics

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | topic | varchar | 255 |  |  |  |  |
| 3 | owner\_id | bigint |  |  | X | X |  |
| 3 | created\_by | varchar | 255 |  |  |  |  |
| 4 | update\_at | datetime |  |  |  |  |  |
| 5 | created\_at | datetime |  |  |  |  |  |

*Table 45 Database - thread\_topics*

#### variables

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | int |  | X | X | X |  |
| 2 | variable\_name | varchar | 255 |  |  |  |  |
| 3 | weight | float |  |  |  |  | Weight from 0 to 1. |
| 4 | updated\_at | datetime |  |  |  |  |  |
| 5 | updated\_by | datetime |  |  |  |  | Name of the admin who changed the variable. |

*Table 46 Database - variables*

### 5.2 Data File Design

We use Aws S3 to store files and images.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **File Name** | **Type** | **Notes** |
| 1 | User avatar | Image | url: “{baseUrl}/api/users/{userId}/{userId}-avatar” |
| 2 | Photographer album thumbnail | Image | url: “{baseUrl}/api/photographers/{ptgId}/{albumid}/{albumId}-thumbnail” |
| 3 | Photographer album image | Image | url: “{baseUrl}/api/photographers /{ptgId}/{albumId}/{imageId}-image” |
| 4 | Photographer cover image | Image | url: “{baseUrl}/api/photographers/{ptgId}/{ptgId}-cover” |
| 5 | Category icon image | Image | url: ““{baseUrl}/api/categories/{categoryId}-category” |

*Table 47 Date File Design*

## 6. Algorithm

### 6.1 Multiple-factors weighted sorting

#### 6.1.1 Definition

Multiple-factors weighted ranking is considered to be a simple yet effective algorithm for recommendation purpose.

#### 6.1.2 Define problem

A suggestion method is always needed in every kind of commercial application. Since it gives the customers an objective way to decide whether a photographer is trustable or not.

It also boosts the photographers’ motivation to give the best of them in order to connect to more customers.

#### 6.1.3 Solution

**Step 1: Choose suitable factors**

We will rank the photographers based on the following factors:

* Customers’ ratings.
* Photographer’s average service price.
* Distance (from photographer’s working location to the booked location).

**Step 2: Specify weight for each factor**

(Admin can change the weights)

|  |  |
| --- | --- |
| **Factor** | **Weight** |
| Rating | 0.3 |
| Distance | 0.3 |
| Price | 0.4 |

**Step 3: Set a common scale for factors**

Common scale = from 0 (worst) to 100 (best)

**Step 4: Transform each object’s factors score to the common scale**

* Rating: 1 rating point on scale of 5 will equal 20 points on scale of 100
* Distance:

Calculate total distances of all photographers in the list

Calculate score of a photographer’s distance factor = (1 – distance / sum) \* max scale

(photographers with higher distance will get lower score)

* Photographer’s average service price

Calculate total prices of all photographers in the list

Calculate score of a photographer’s price factor = (1 – price / sum) \* max scale

(Photographer with higher price will get lower score)

**Step 5: Sum the scores**

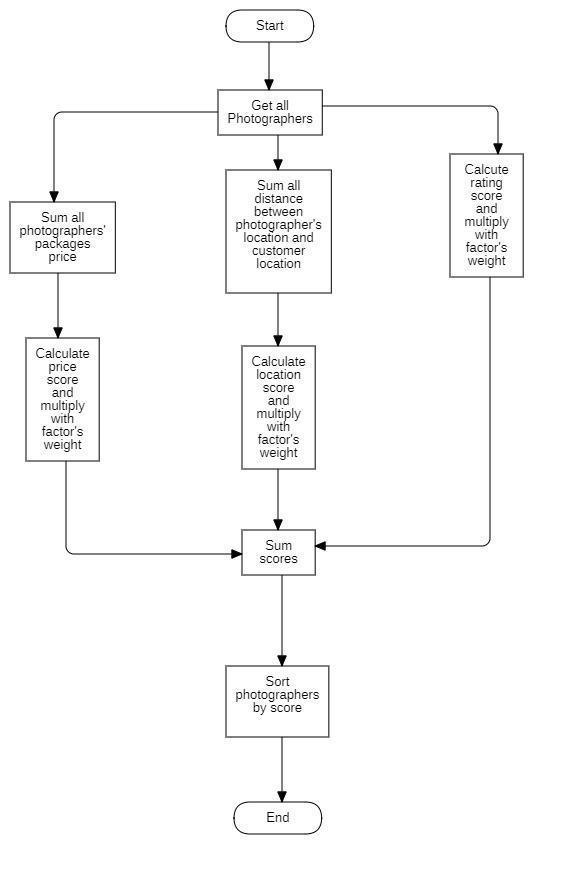
Score of a photographer = 0.3 \* rating score + 0.3 \* distance score + 0.4 \* price score

**Step 6: Sort photographers based on the score.**

#### 6.1.4 Complexity

The complexity of this algorithm is O(n) for finding n photographers.

#### 6.1.5 Flowchart



*Figure 23 Flowchart - Multiple factors sorting*

### 6.2 Naive Bayes Classifier

#### 6.2.1 Definition

Naive bayes classifiers is a simple technique mostly used for building classifiers based on Bayes’ theorem from probability theory and statistics.

It is best suited to solve problems which has categorizable variables Some application of naive bayes includes: Spam email filtering, sentiment prediction, or in our case, weather prediction.

#### 6.2.2 Define problem

Our application domain is photography, which has highly dependence on weather in order to make the most out of a shooting session. So, the problem here is we need a tool which could warn customer or photographer on bad weather days.

#### 6.2.3 Solution

Some definitions:

Bayes’ Theorem:

P(A|B) - Posterior: The probability of “A” being true given that “B” is already true.

P(B|A) - Likelihood: The probability of “B” being true given that “A” is already true.

P(A) - Prior: The probability of “A” being true.

P(B) - Evidence: The probability of “B” being true.

Input:

* A set of features x = {x1, x2… xn}, in our case x = {Outlook, Temperature, Humidity, Windy}
* A fixed set of classes C = {c1, c2… cn}, in our case C = {Yes, No}

Output

* A predicted class c from C.

For documenting purpose, we use a fictional example dataset below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Outlook** | **Temperature** | **Humidity** | **Windy** | **Shooting** |
| Sunny | High | High | Strong | No |
| Overcast | Normal | Normal | Normal | Yes |
| Rainy | High | Normal | Strong | No |
| Sunny | Normal | Normal | Normal | Yes |
| Rainy | High | Normal | Strong | No |
| Overcast | Normal | High | Normal | Yes |
| Rainy | High | High | Strong | No |

We will convert this to Frequency tables:

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Outlook** | | |
| Sunny | Overcast | Rainy |
| Yes | 1 | 2 | 0 |
| No | 1 | 0 | 3 |

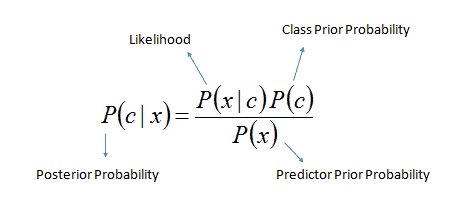
|  |  |  |
| --- | --- | --- |
| **Class** | **Temperature** | |
| High | Normal |
| Yes | 0 | 4 |
| No | 4 | 0 |

|  |  |  |
| --- | --- | --- |
| **Class** | **Humidity** | |
| High | Normal |
| Yes | 1 | 2 |
| No | 2 | 1 |

|  |  |  |
| --- | --- | --- |
| **Class** | **Windy** | |
| Strong | Normal |
| Yes | 0 | 4 |
| No | 4 | 0 |

For example, if we want to know if a set of features {Sunny, High Temperature, High Humidity, Strong Windy} is suitable for photographing or not.

According to Bayes theorem we need to calculate posterior probability.



Or we can simply calculate in expanded form:

posterior_probability_expanded

We calculate for each class (Yes and No) and then compare it to find out which gives higher score.

Since we want to classify sunny outlook, high temperature, high humidity and strong wind, we need to calculate following probabilities:

P (Yes) =

P (No) =

P (Sunny | Yes) =

P (Sunny | No) =

P (Temperature = High | Yes) =

In this case, to avoid the multiplication of probabilities become 0, we will add 1 to the value.

-> P (Temperature = High | Yes) =

P (Temperature = High | No) =

P (Humidity = High | Yes) =

P (Humidity = High | No) =

P (Windy = Strong | Yes) =

Same as the case above, we will add to 1 to the value.

-> P (Windy = Strong | Yes) =

P (Windy = Normal | No) =

Final step, we will find two posterior probabilities in expanded form (Without divide it for the evidence):

* P (Yes | X):

* P (No | X):

As 0.0714 > 0.00529 so we can predict that the weather is not suitable for shooting.

#### 6.2.4 Complexity

The complexity of Naive Bayes is O(nK), where n is number of features and K is number of classes.