

**CAPSTONE PROJECT REPORT**

**Report 4 – Software Design Document**

– HoChiMinh, September 2020 –

**Table of Contents**

[I. Project Report 6](#_heading=h.1ksv4uv)

[1. Status Report 6](#_heading=h.44sinio)

[2. Team Involvements 6](#_heading=h.2jxsxqh)

[3. Issues/Suggestions 6](#_heading=h.z337ya)

[II. Software Design Document 7](#_heading=h.3j2qqm3)

[1. Overall Description 7](#_heading=h.1y810tw)

[1.1 Assumptions 7](#_heading=h.4i7ojhp)

[1.2 Design Constraints 7](#_heading=h.2xcytpi)

[2. System Architecture Design 7](#_heading=h.1ci93xb)

[2.1 Overall Architecture 7](#_heading=h.3whwml4)

[2.2 System Architecture 8](#_heading=h.2bn6wsx)

[2.3 Package Diagram 8](#_heading=h.qsh70q)

[3. System Detailed Design 12](#_heading=h.ihv636)

[3.1 Overall 12](#_heading=h.32hioqz)

[a. Class Diagram 12](#_heading=h.1hmsyys)

[b. Sequence Diagram(s) 12](#_heading=h.2grqrue)

[c. Activity Diagram(s) 14](#_heading=h.4f1mdlm)

[d. State Machine Diagram(s) 21](#_heading=h.2u6wntf)

[4. Class Specifications 22](#_heading=h.3tbugp1)

[4.1 User 22](#_heading=h.28h4qwu)

[4.2 Role 23](#_heading=h.nmf14n)

[4.3 ERole 23](#_heading=h.37m2jsg)

[4.4 Package 24](#_heading=h.1mrcu09)

[4.5 Service 24](#_heading=h.46r0co2)

[4.6 Booking 25](#_heading=h.2lwamvv)

[4.7 EBookingStatus 25](#_heading=h.111kx3o)

[4.8 Album 26](#_heading=h.3l18frh)

[4.9 Image 27](#_heading=h.206ipza)

[4.10 Category 27](#_heading=h.4k668n3)

[4.11 DayOfWeek 28](#_heading=h.2zbgiuw)

[4.12 Location 28](#_heading=h.1egqt2p)

[4.13 BookingComment 28](#_heading=h.3ygebqi)

[4.14 BookingDetail 29](#_heading=h.2dlolyb)

[4.15 BusyDay 29](#_heading=h.sqyw64)

[4.16 Thread 30](#_heading=h.3cqmetx)

[4.17 ThreadTopic 30](#_heading=h.1rvwp1q)

[4.18 ThreadComment 31](#_heading=h.4bvk7pj)

[4.19 ReturningType 31](#_heading=h.2r0uhxc)

[4.20 EReturningType 32](#_heading=h.1664s55)

[4.21 TimeLocationDetail 32](#_heading=h.3q5sasy)

[5. Data & Database Design 33](#_heading=h.25b2l0r)

[5.1 Database Design 33](#_heading=h.kgcv8k)

[5.1.1 users 33](#_heading=h.1jlao46)

[5.1.2 roles 34](#_heading=h.43ky6rz)

[5.1.3 photographer\_packages 34](#_heading=h.2iq8gzs)

[5.1.4 packages\_services 34](#_heading=h.xvir7l)

[5.1.5 services 34](#_heading=h.3hv69ve)

[5.1.6 bookings 35](#_heading=h.1x0gk37)

[5.1.7 albums 35](#_heading=h.4h042r0)

[5.1.8 locations 36](#_heading=h.2w5ecyt)

[5.1.9 categories 36](#_heading=h.1baon6m)

[5.1.10 album\_images 36](#_heading=h.3vac5uf)

[5.1.11 images 36](#_heading=h.2afmg28)

[5.1.12 customers\_albums 36](#_heading=h.pkwqa1)

[5.1.13 working\_day\_of\_week 37](#_heading=h.39kk8xu)

[5.1.14 busy\_days 37](#_heading=h.1opuj5n)

[5.1.15 returning\_types 37](#_heading=h.48pi1tg)

[5.1.16 booking\_details 37](#_heading=h.2nusc19)

[5.1.17 time\_location\_details 38](#_heading=h.1302m92)

[5.1.18 booking\_comments 38](#_heading=h.3mzq4wv)

[5.1.19 threads 38](#_heading=h.2250f4o)

[5.1.20 thread\_comments 38](#_heading=h.haapch)

[5.1.21 thread\_topics 39](#_heading=h.319y80a)

[5.2 Data File Design 39](#_heading=h.1gf8i83)

[6. Algorithm 39](#_heading=h.40ew0vw)

[6.1 Multiple-factors weighted sorting 39](#_heading=h.2fk6b3p)

[6.1.1 Definition 39](#_heading=h.upglbi)

[6.1.2 Define problem 39](#_heading=h.3ep43zb)

[6.1.3 Solution 39](#_heading=h.1tuee74)

[6.1.4 Complexity 40](#_heading=h.4du1wux)

[6.1.5 Flowchart 41](#_heading=h.2szc72q)

[Figure 1 Overall Architecture](about:blank) 6

[Figure 2 Webapi Package Diagram 8](#_heading=h.3as4poj)

[Figure 3 Mobile Package Diagram 10](#_heading=h.23ckvvd)

[Figure 4 Class Diagram 11](#_heading=h.41mghml)

[Figure 5 Sequence Diagram - Add Category 12](#_heading=h.vx1227)

[Figure 6 Sequence Diagram - Add Delivery Method 12](#_heading=h.3fwokq0)

[Figure 7 Sequence Diagram - Block User 13](#_heading=h.1v1yuxt)

[Figure 8 Activity Diagram - Make Booking Request](about:blank) 14

[Figure 9 Activity Diagram - Edit Request](about:blank) 15

[Figure 10 Activity Diagram- Cancel Booking](about:blank) 16

[Figure 11 Activity Diagram - Rate Booking](about:blank) 17

[Figure 12 Activity Diagram - Accept Booking](about:blank) 18

[Figure 13 Activity Diagram - Reject Booking](about:blank) 19

[Figure 14 Activity Diagram - Cancel Booking](about:blank) 20

[Figure 15 State Machine Diagram of Booking 21](#_heading=h.19c6y18)

[Figure 16 Database Design 32](#_heading=h.34g0dwd)

[Figure 17 Flowchart - Multiple factors sorting 40](#_heading=h.184mhaj)

# I. Project Report

## 1. Status Report

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Work Item** | **Status** | **Notes (Work Item in Details)** |
| 1 |  | Pending |  |
| 2 |  | In Progress |  |
| 3 |  | Completed |  |

## 2. Team Involvements

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Task** | **Member** | **Notes (Task Details, etc.)** |
| 1 |  | Thaott |  |
| 2 |  | Datbc |  |
| 3 |  | Kiendst |  |
| 4 |  | Mytn |  |

## 3. Issues/Suggestions

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Issue** | **Status** | **Notes (Solution, Suggestion, etc.)** |
| 1 |  | Pending |  |
| 2 |  | In Progress |  |
| 3 |  | Completed |  |

# 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.

The process and algorithm for calculating available schedules are assumed to be given prior.

### 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

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**



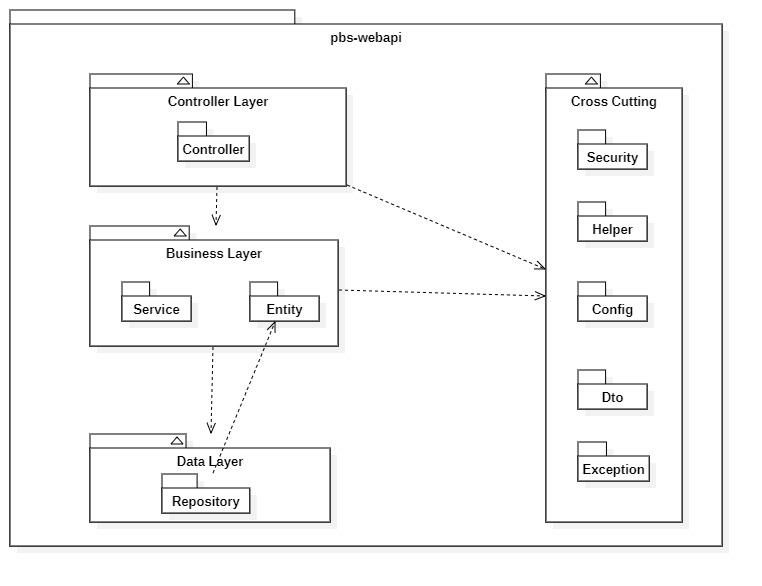


Bloc pattern allow us to separate our mobile app into 3 layers:

* 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**

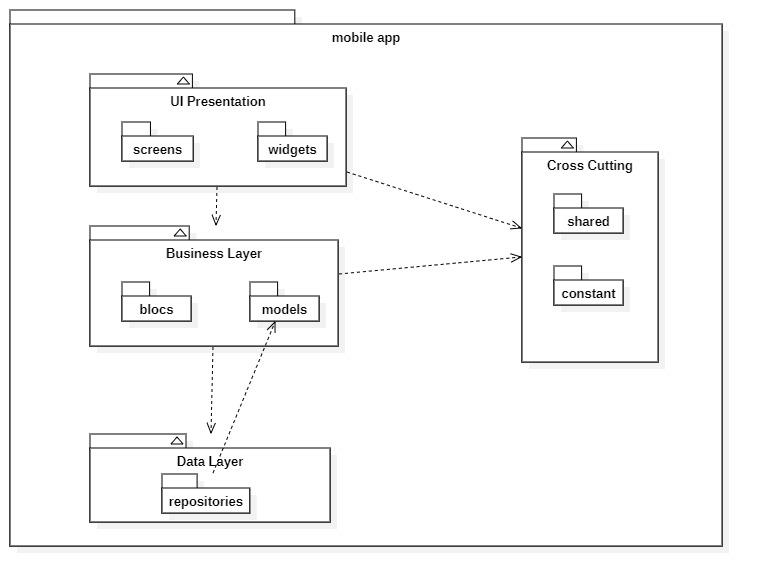
****

*Figure 2 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 |

**2.3.2 Mobile App**

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



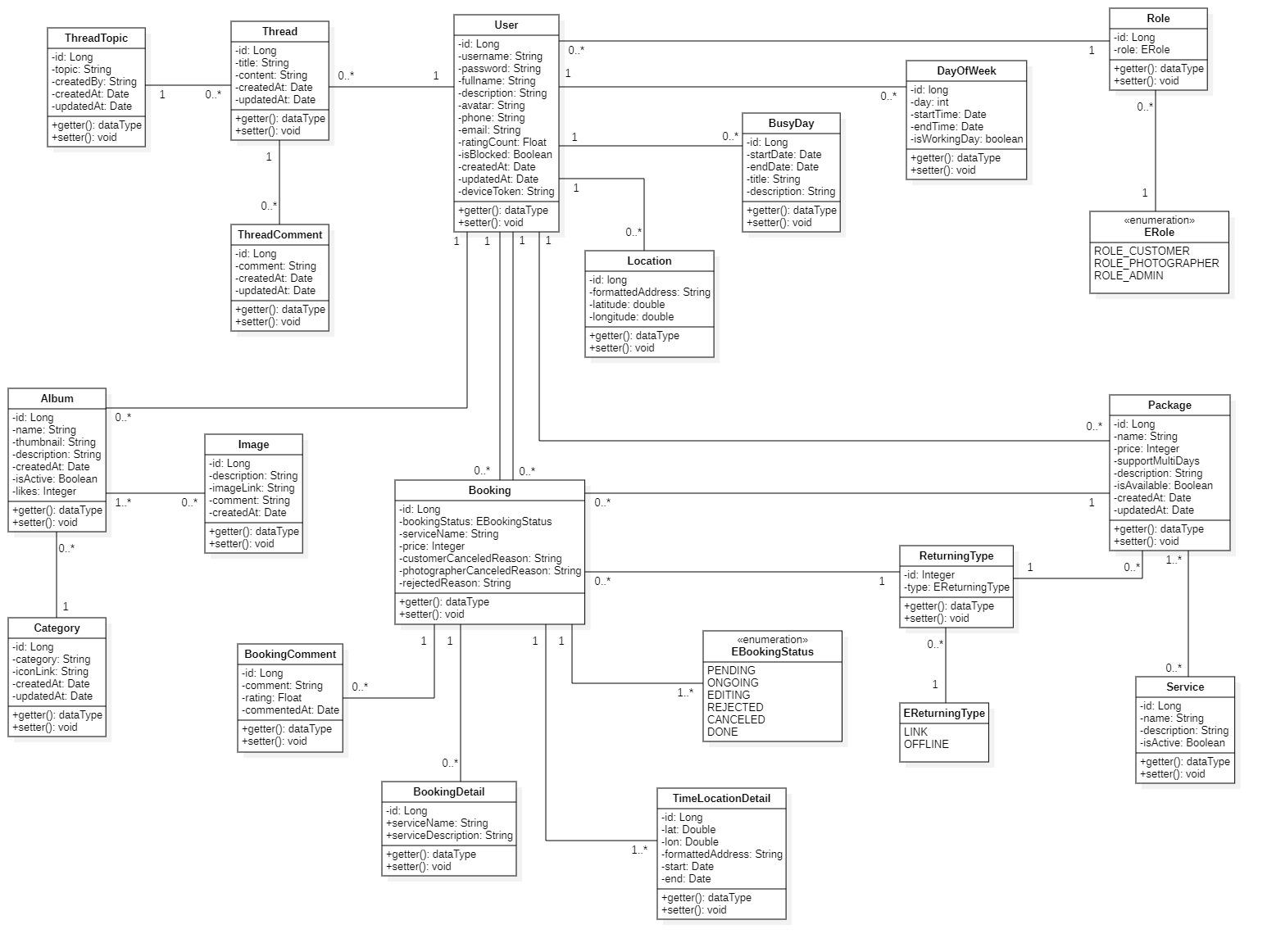
*Figure 3 Mobile Package Diagram*

|  |  |  |
| --- | --- | --- |
| **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 |

## 3. System Detailed Design

### 3.1 Overall

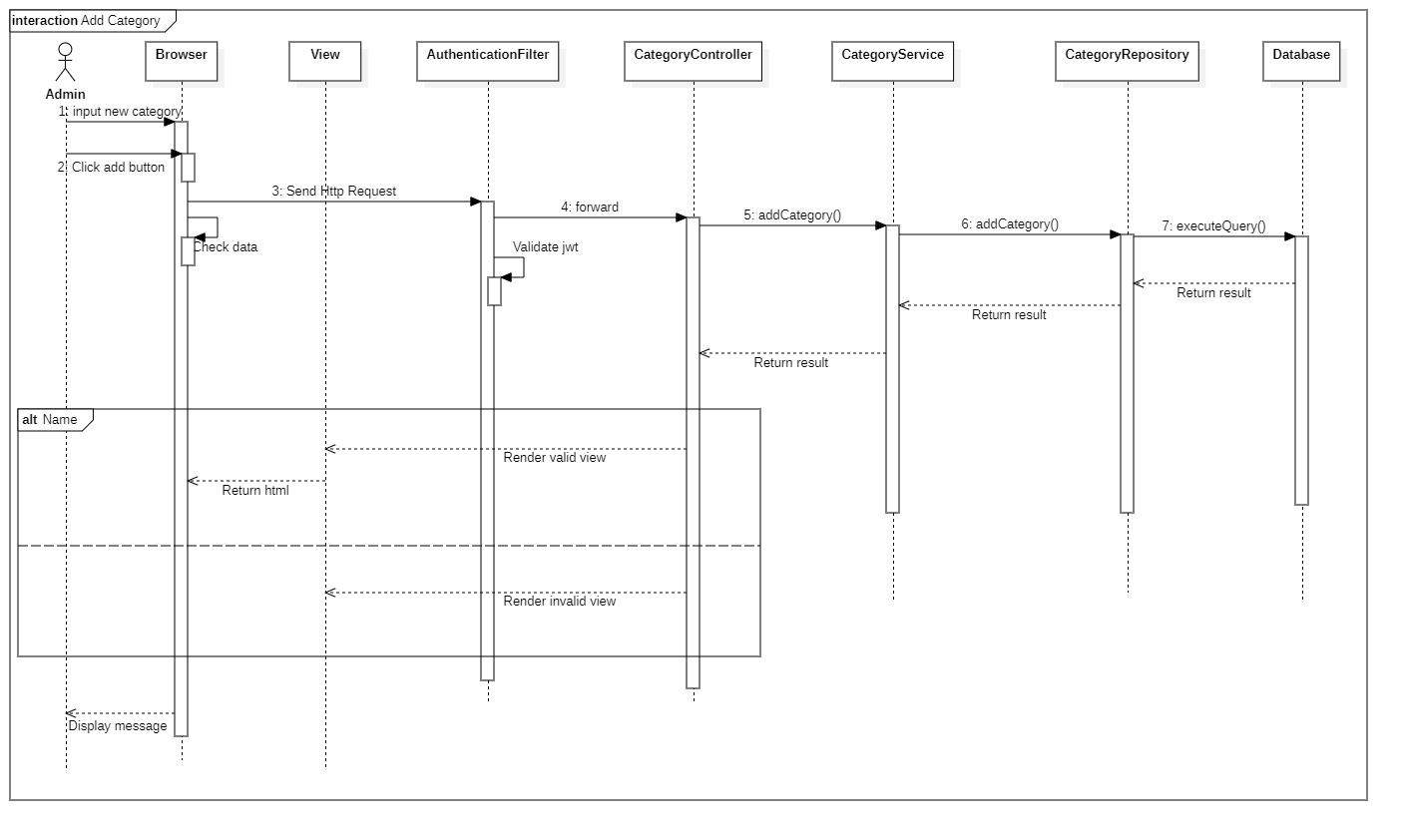
#### a. Class Diagram



*Figure 4 Class Diagram*

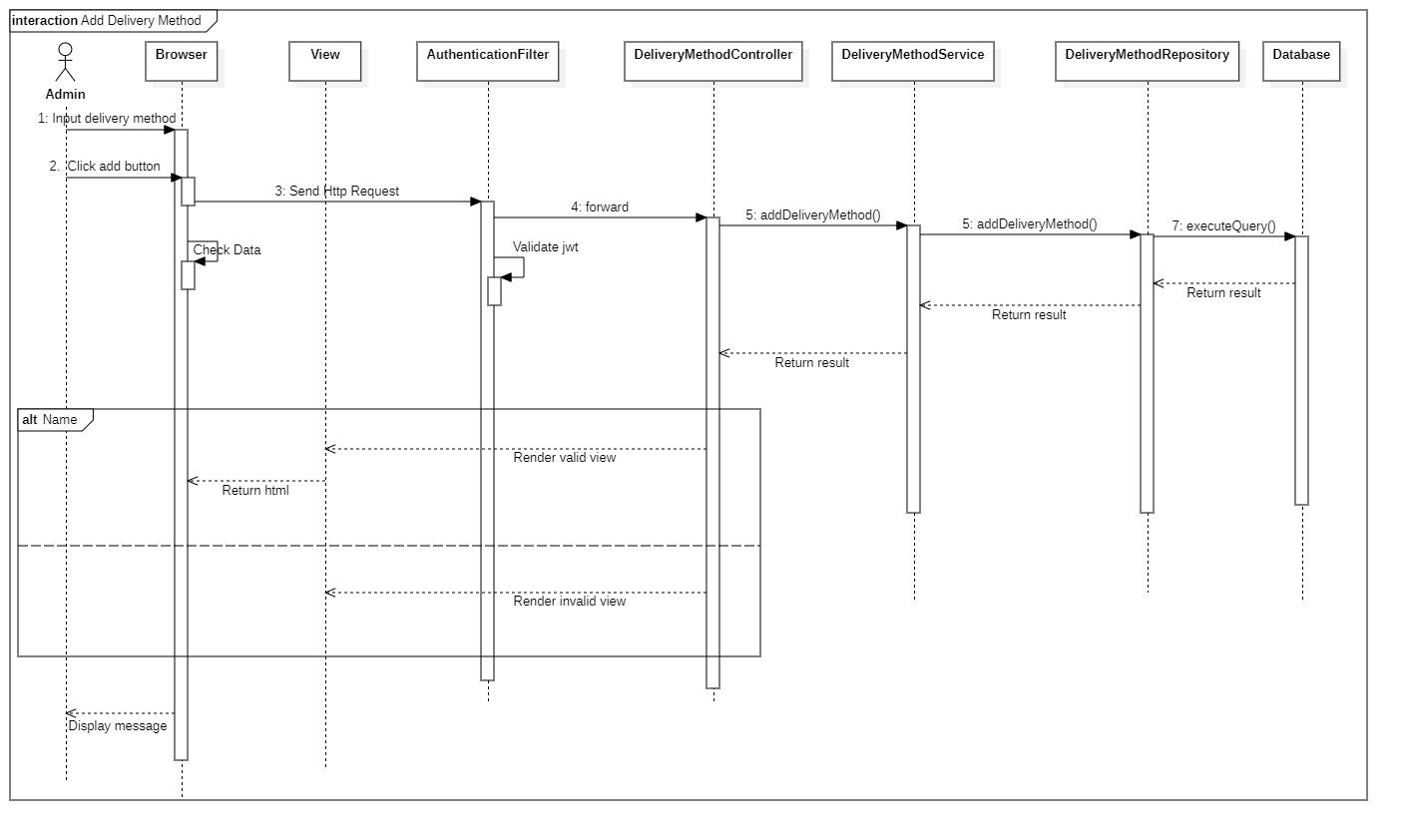
#### b. Sequence Diagram(s)

1. Add Category



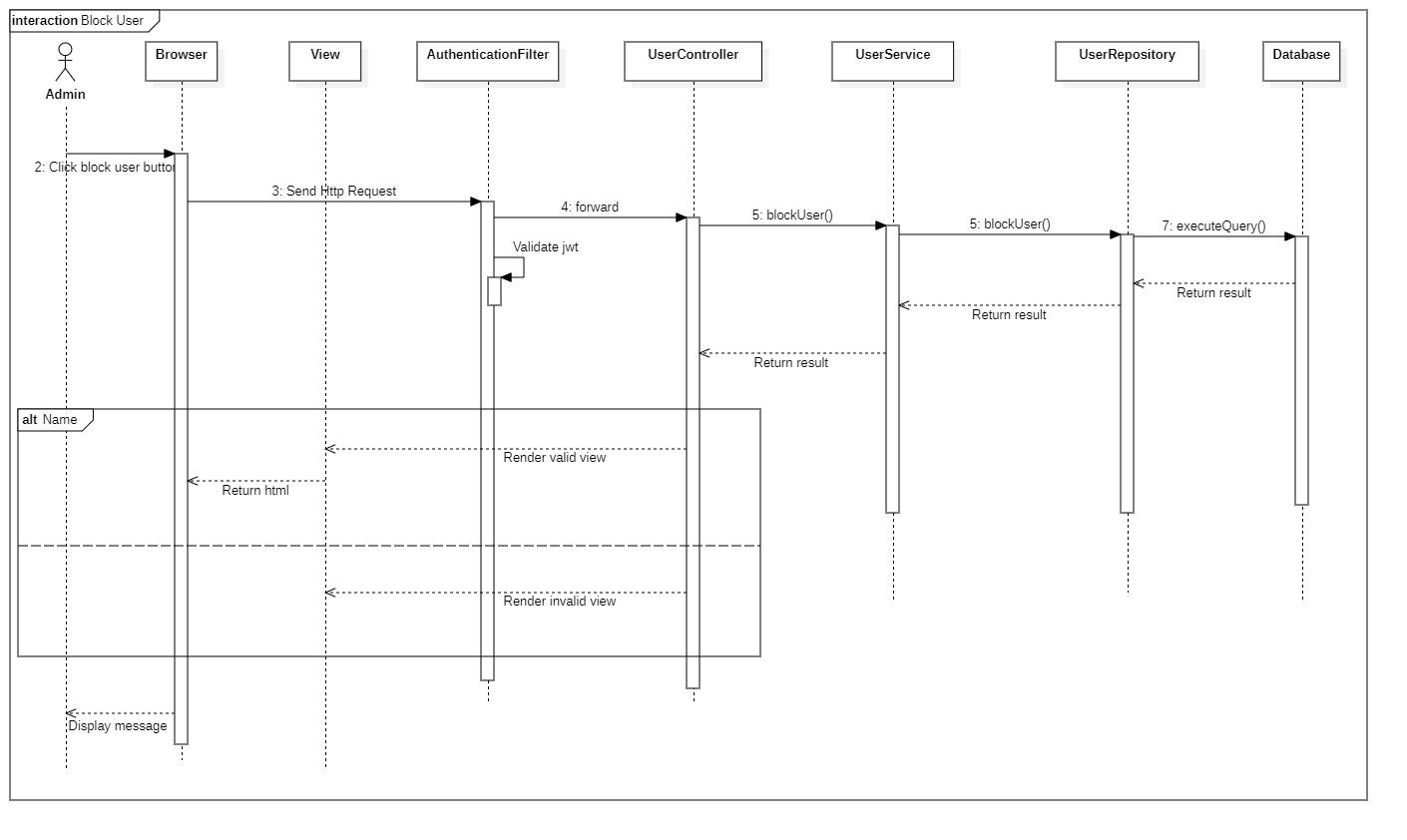
*Figure 5 Sequence Diagram - Add Category*

1. Add delivery method



*Figure 6 Sequence Diagram - Add Delivery Method*

1. Block user

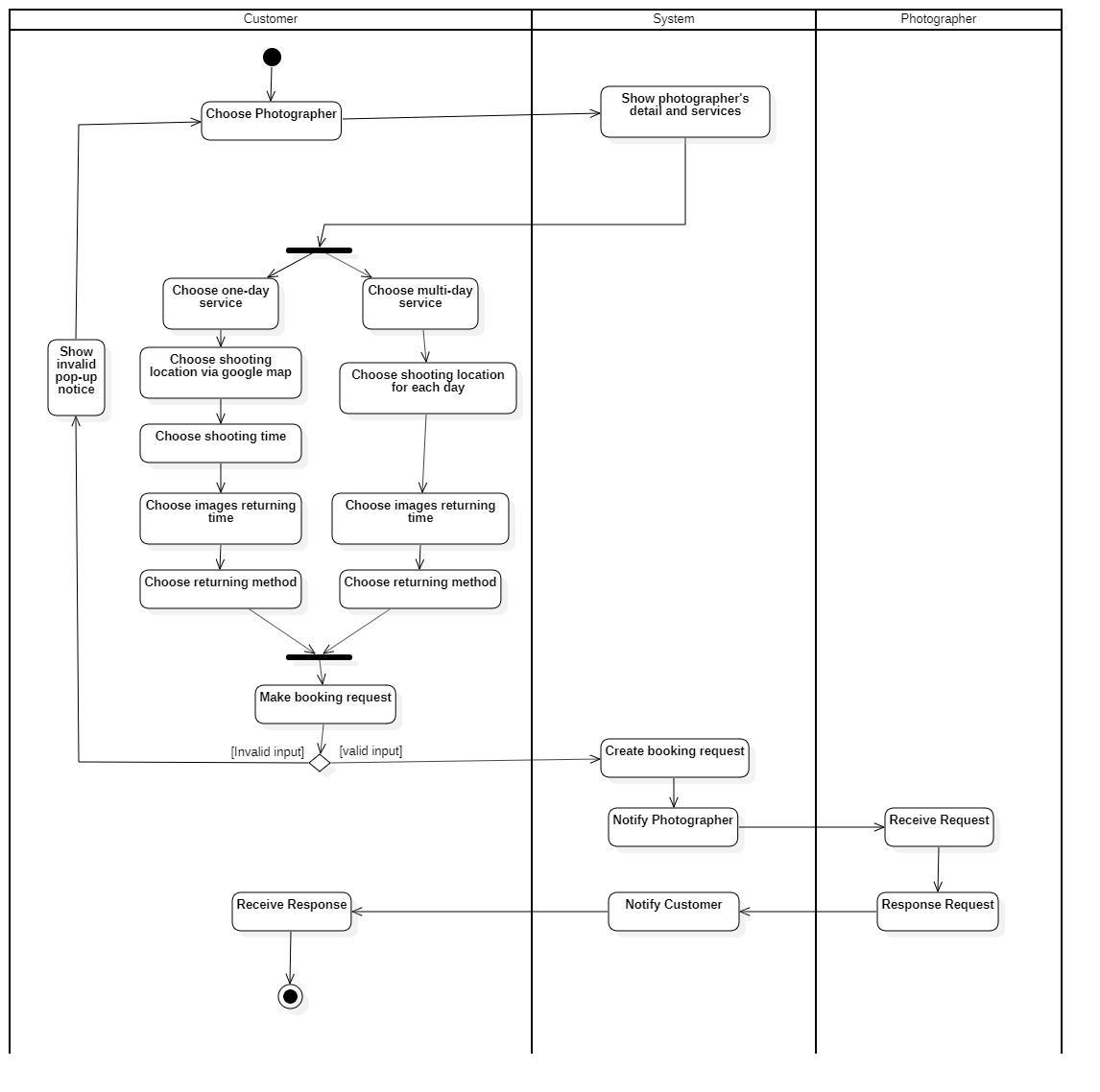


*Figure 7 Sequence Diagram - Block User*

#### c. Activity Diagram(s)

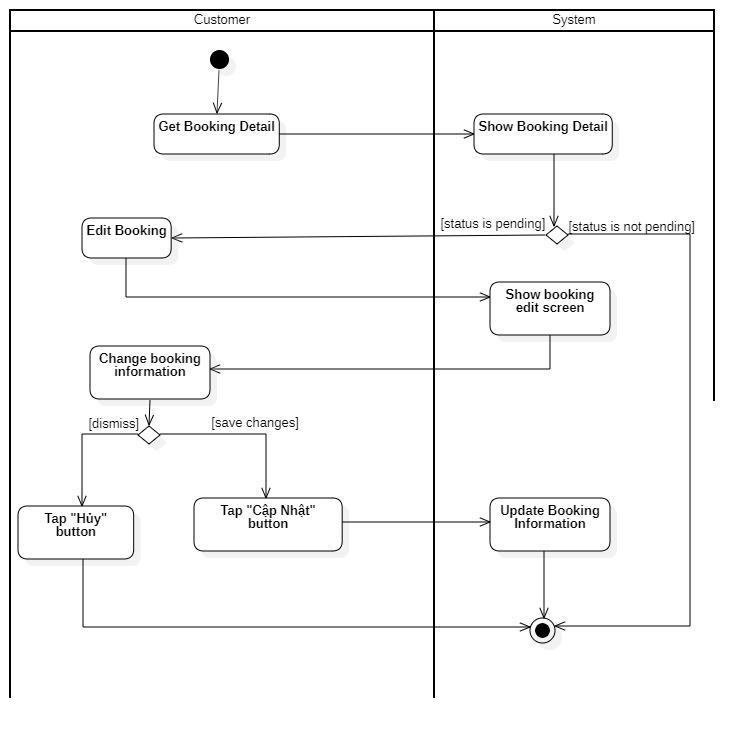
1. **<Customer>** **Make request**

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



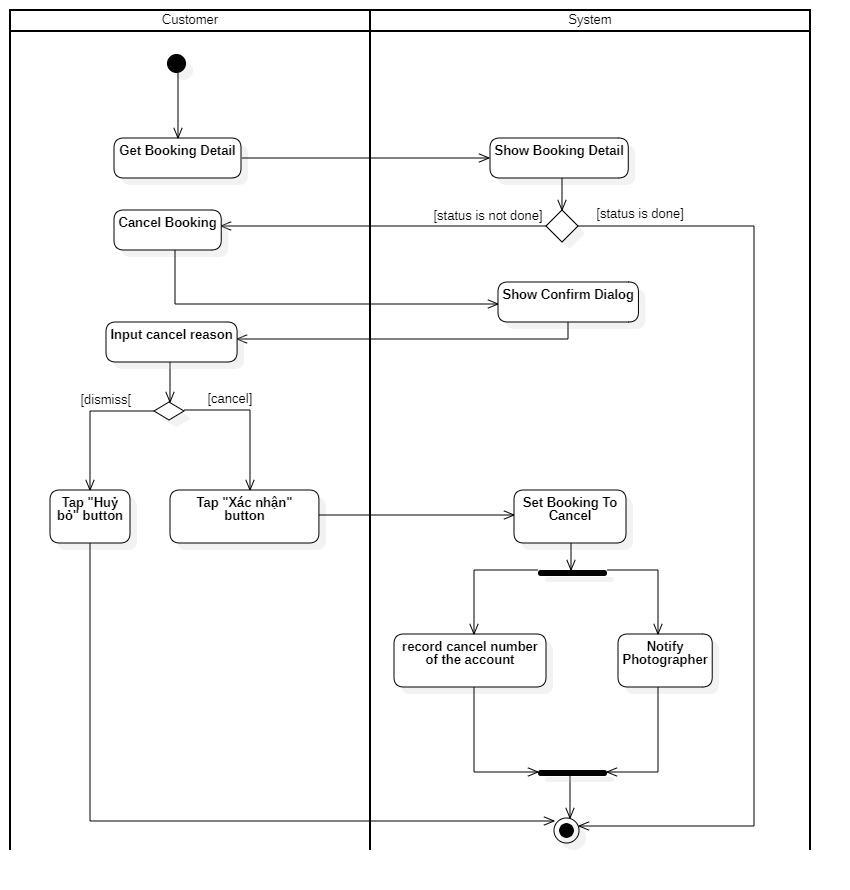
1. **<Customer> Edit request**

Summary: This diagram shows the process when customer edit request.



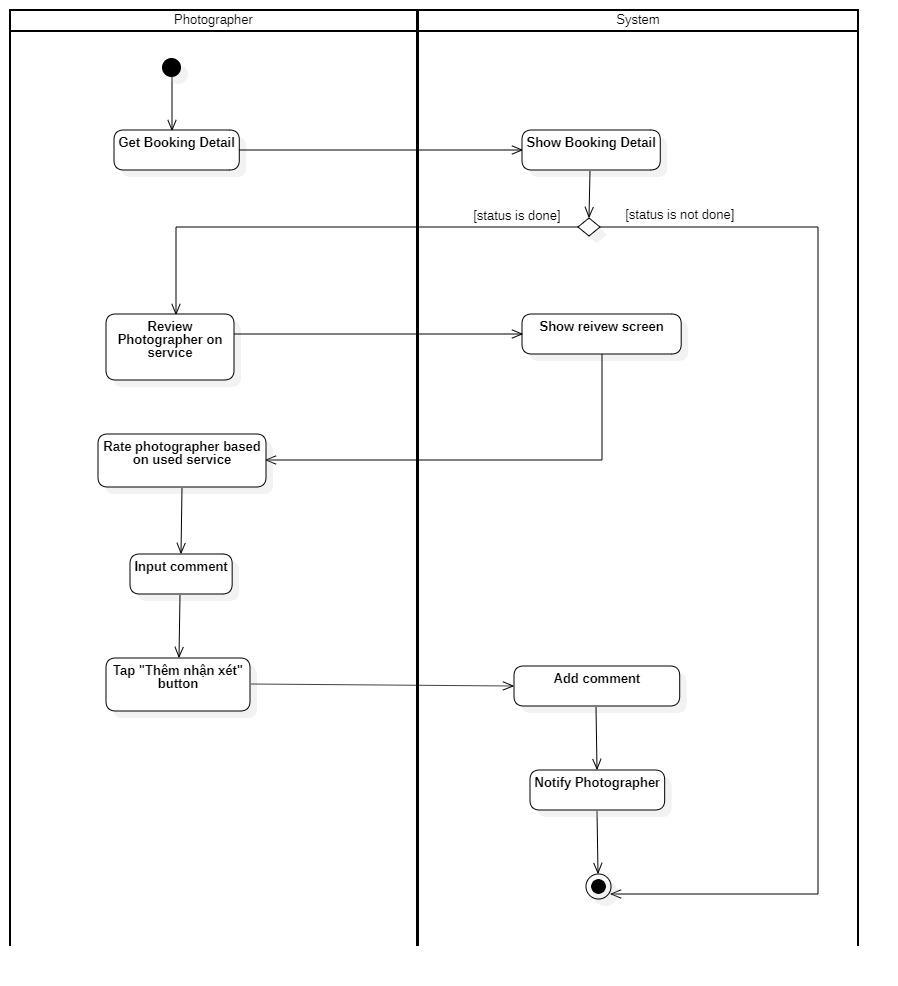
1. **<Customer> Cancel booking**

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



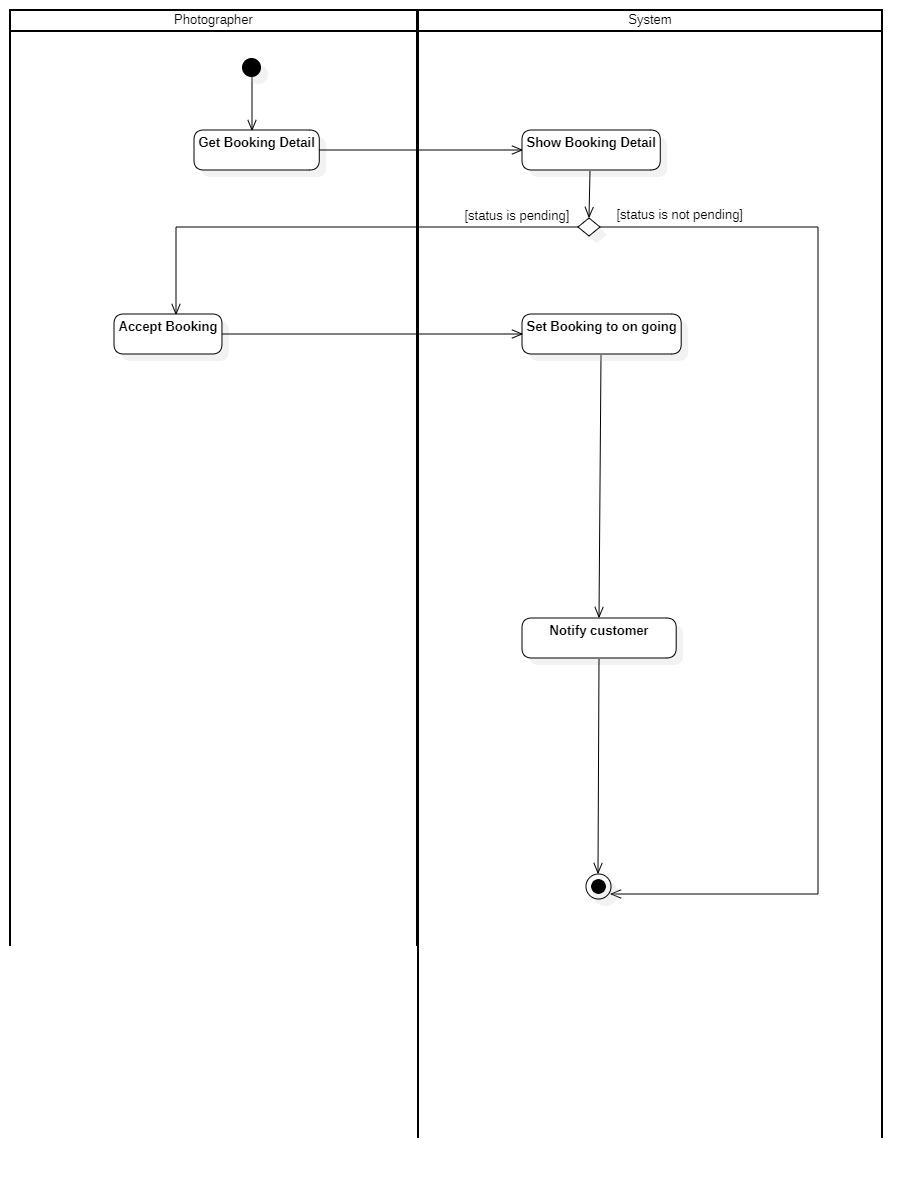
1. **<Customer> Rating**

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



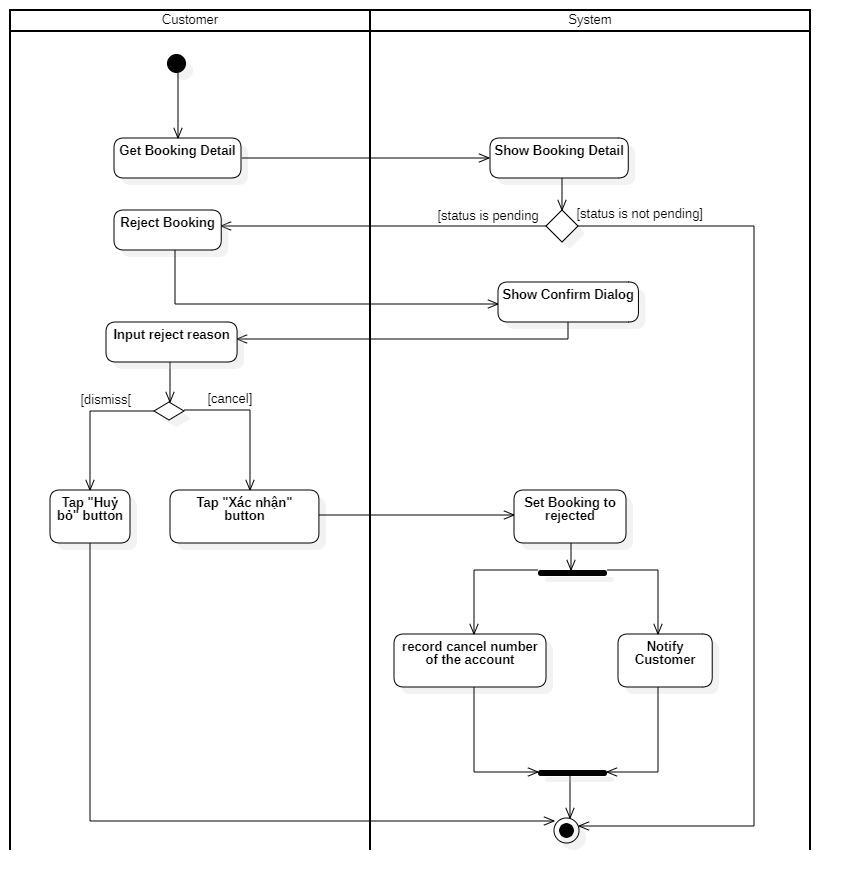
1. **<Photographer> Accept booking**

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



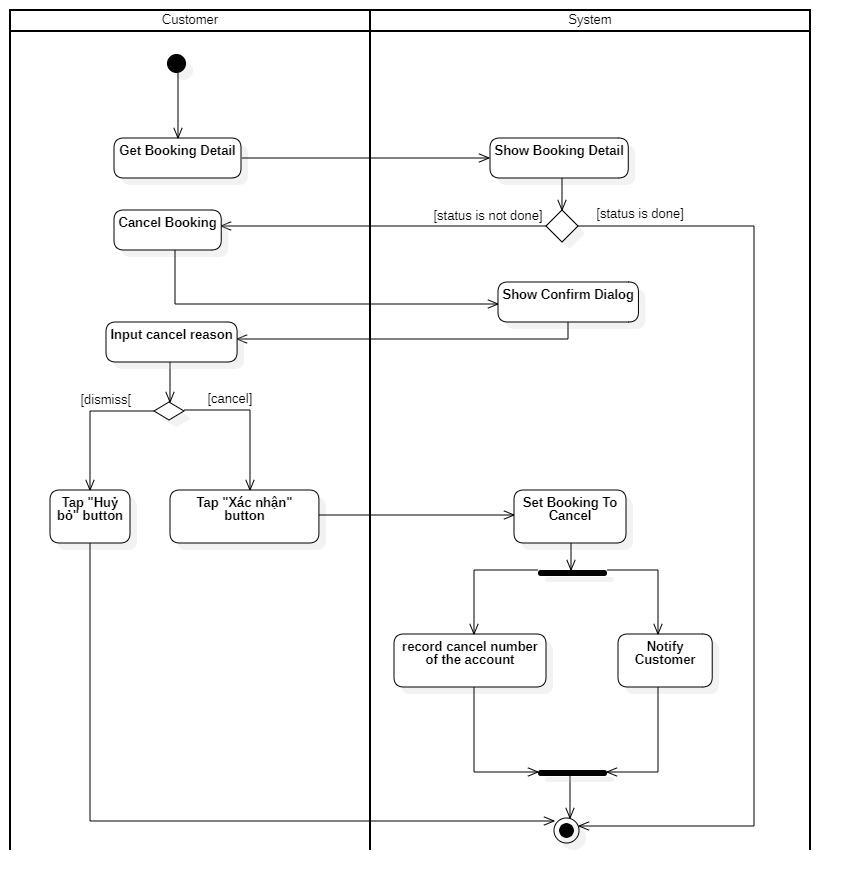
1. **<Photographer> Reject booking**

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



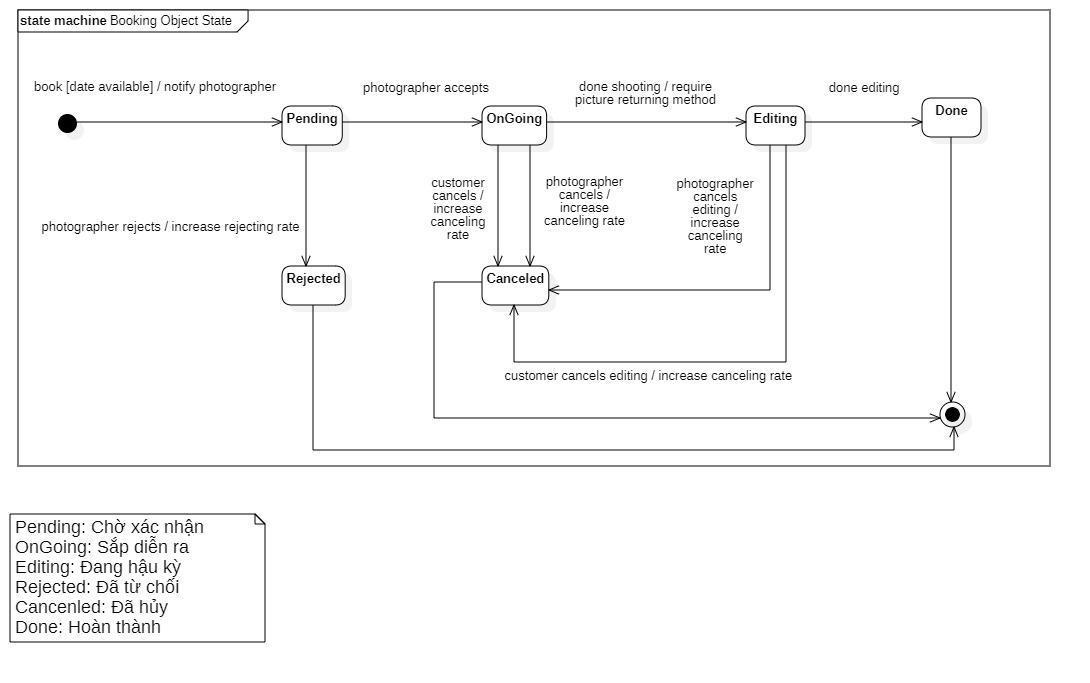
1. **<Photographer> Cancel booking**

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



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

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



*Figure 15 State Machine Diagram of 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 |

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

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 4.21 TimeLocationDetail

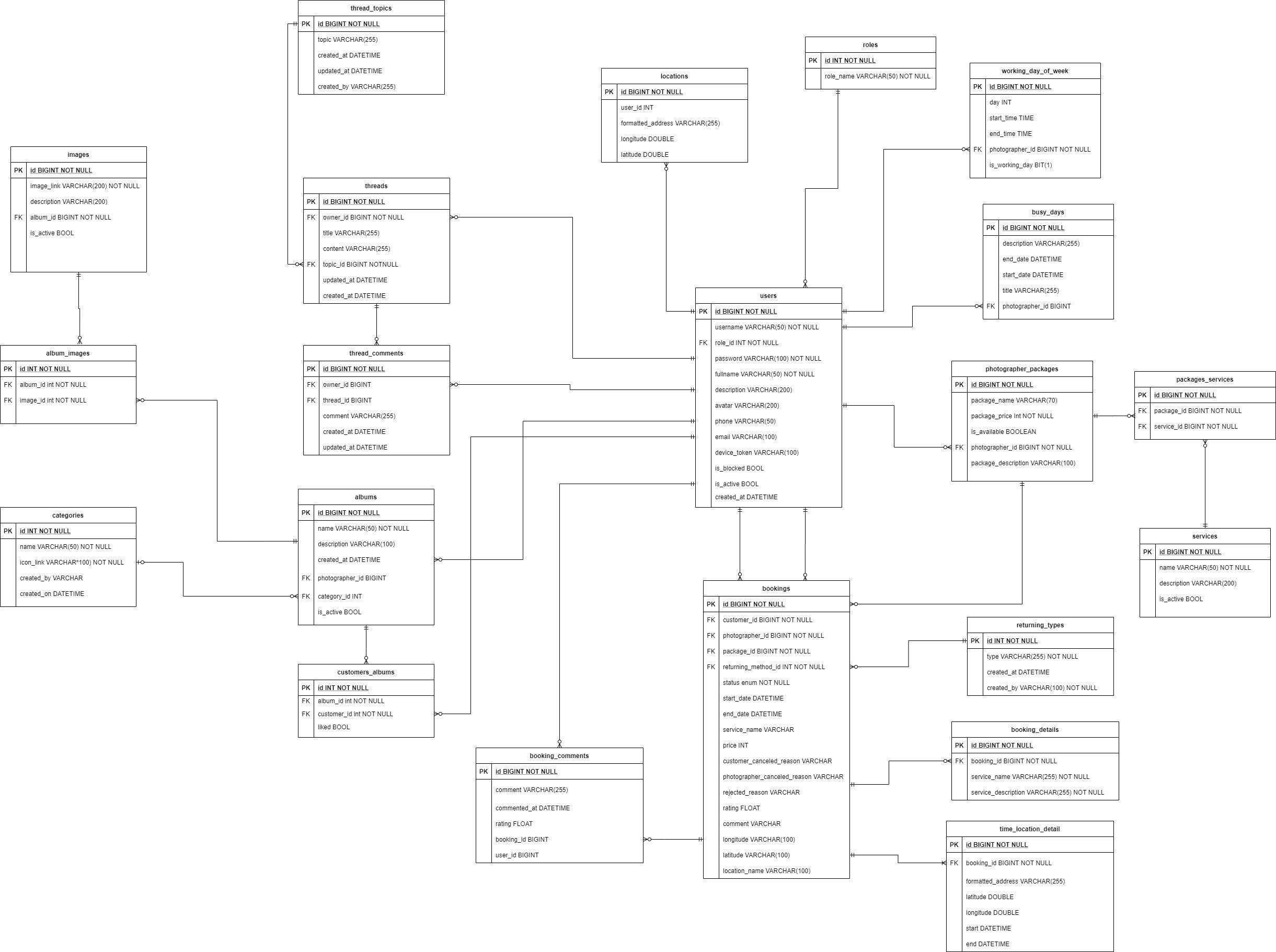
This table used for saving time and location details of a booking

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

## 5. Data & Database Design

### 5.1 Database Design



*Figure 16 Database Design*

#### 5.1.1 users

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | username | varchar | 50 | X | X |  |  |
| 3 | password | varchar | 100 |  | X |  |  |
| 4 | fullname | varchar | 50 |  | X |  |  |
| 5 | description | varchar | 200 |  |  |  |  |
| 6 | avatar | varchar | 200 |  |  |  | Avatar image link of user |
| 7 | phone | varchar | 50 |  |  |  |  |
| 8 | email | varchar | 100 |  |  |  |  |
| 9 | device\_token | varchar | 100 |  |  |  | The firebase token of device, using for firebase messaging. |
| 9 | is\_blocked | boolean |  |  |  |  |  |
| 10 | is\_active | boolean |  |  |  |  |  |
| 11 | created\_at | datetime |  |  |  |  |  |
| 12 | role\_id | int |  |  | X | X |  |

#### roles

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | int |  | X | X | X |  |
| 2 | role\_name | varchar | 50 |  | X |  |  |

#### photographer\_packages

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | package\_name | varchar | 70 |  | X |  |  |
| 3 | package\_price | int |  |  | X |  |  |
| 4 | is\_available | boolean |  |  |  |  |  |
| 5 | package \_description | varchar | 100 |  |  |  |  |
| 6 | photographer \_id | int |  |  | X | X |  |

#### packages\_services

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | package\_id | int |  |  | X | X |  |
| 3 | service\_id | int |  |  | X | X |  |

#### services

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | name | varchar | 50 |  | X |  |  |
| 3 | description | varchar | 200 |  |  |  |  |
| 4 | is\_active | boolean |  |  |  |  |  |

#### bookings

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | status | enum |  |  | X |  | There will be 6 status: Pending, Ongoing, Editing, Done, Rejected, Canceled |
| 3 | start\_date | datetime |  |  | X |  |  |
| 4 | end\_date | datetime |  |  | X |  |  |
| 5 | service\_name | varchar | 100 |  | X |  | Package’s name |
| 6 | price | int |  |  | X |  | Package’s price |
| 7 | customer  \_canceled  \_reason | varchar | 100 |  |  |  | In case customers cancel and provide their reason |
| 8 | photographer \_canceled \_reason | varchar | 100 |  |  |  | In case photographers cancel and provide their reason |
| 9 | rejected\_reason | varchar | 100 |  |  |  | In case photographers reject the request and provide reason |
| 10 | rating | float |  |  |  |  | After booking is done, customer can rate the booking |
| 11 | comment | varchar | 100 |  |  |  | After booking is done, customer can comment on the booking |
| 12 | longitude | varchar | 100 |  |  |  |  |
| 13 | latitude | varchar | 100 |  |  |  |  |
| 14 | location\_name | varchar | 100 |  |  |  | The shooting location picked by customer |
| 15 | customer\_id | int |  |  | X | X |  |
| 16 | photographer \_id | int |  |  | X | X |  |
| 17 | package\_id | int |  |  | X | X |  |

#### albums

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | name | varchar | 50 |  | X |  |  |
| 3 | description | varchar | 100 |  |  |  |  |
| 4 | created\_at | datetime |  |  |  |  |  |
| 5 | is\_active | boolean |  |  |  |  |  |
| 6 | photographer \_id | int |  |  | X | X |  |
| 7 | category\_id | int |  |  |  | X |  |

#### locations

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | Formatted\_address | varchar | 255 |  | X |  |  |
| 3 | longitude | double | 100 |  | X |  |  |
| 4 | latitude | double | 100 |  | X |  |  |
| 5 | user\_id | int |  |  | X | X |  |

#### categories

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | int |  | X | X | X |  |
| 2 | name | varchar | 50 |  | X |  |  |
| 3 | icon\_link | varchar | 100 |  | X |  | Link for the icon of category |
| 4 | created\_by | varchar | 100 |  |  |  | Name of the admin who created the category |
| 5 | Created\_on | datetime |  |  |  |  |  |

#### album\_images

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | int |  | X | X | X |  |
| 2 | album\_id | int |  |  | X | X |  |
| 3 | image\_id | int |  |  | X | X |  |

#### images

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | image\_link | varchar | 200 |  | X |  |  |
| 3 | description | varchar | 200 |  |  |  |  |
| 4 | is\_active | boolean |  |  |  |  |  |
| 5 | album\_id | int |  |  | X | X |  |

#### customers\_albums

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | int |  | X | X | X |  |
| 2 | liked | boolean |  |  |  |  | represent if customer has liked the album |
| 3 | album\_id | int |  |  | X | X |  |
| 4 | customer\_id | int |  |  | X | X |  |

#### working\_day\_of\_week

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | day | int |  |  |  |  |  |
| 3 | start\_time | datetime |  |  |  |  |  |
| 4 | end\_time | datetime |  |  |  |  |  |
| 5 | is\_working\_day | bit |  |  |  |  |  |
| 6 | photographer \_id | bigint |  |  | X | X |  |

#### 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 |  |

#### 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 | 100 |  | X |  |  |

#### 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 |  |

#### time\_location\_details

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | bigint |  | X | X | X |  |
| 2 | formatted\_address | varchar | 255 |  |  |  |  |
| 3 | laitude | double |  |  |  |  |  |
| 4 | longitude | double |  |  |  |  |  |
| 5 | start | datetime |  |  |  |  |  |
| 6 | end | datetime |  |  |  |  |  |
| 7 | booking\_id | bigint |  |  | X | X |  |

#### 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 |  |  |  |  |  |
| 5 | user\_id | bigint |  |  |  |  |  |
| 6 | booking\_id | bigint | 100 |  |  |  |  |

#### 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 |  |

#### 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 |  |

#### thread\_topics

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

### 5.2 Data File Design

We use Aws S3 to store files and images.

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

## 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.

#### 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 + 0.3 \* 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 17 Flowchart - Multiple factors sorting*