

**Bird training system**

**Software Design Specification**

– Ho Chi Minh city, June 2023 –

Table of Contents

[I. Overview 10](#_Toc140001039)

[1. Code Packages 10](#_Toc140001040)

[2. Database Design 11](#_Toc140001041)

[a. Database Schema 11](#_Toc140001042)

[II. Code Designs 11](#_Toc140001043)

[1. <Public/View Workshop Detail> 11](#_Toc140001044)

[a. Class Diagram 11](#_Toc140001045)

[b. Class Specifications 11](#_Toc140001046)

[c. Sequence Diagram(s) 12](#_Toc140001047)

[d. Database Queries 12](#_Toc140001048)

[2. <Public/View Online Course Detail> 13](#_Toc140001049)

[a. Class Diagram 13](#_Toc140001050)

[b. Class Specifications 13](#_Toc140001051)

[c. Sequence Diagram(s) 13](#_Toc140001052)

[d. Database Queries 13](#_Toc140001053)

[3. <Customer/Payment for Online Course> 14](#_Toc140001054)

[a. Class Diagram 14](#_Toc140001055)

[b. Class Specifications 14](#_Toc140001056)

[c. Sequence Diagram(s) 14](#_Toc140001057)

[d. Database Queries 14](#_Toc140001058)

[4. <Customer/Payment for Workshop> 15](#_Toc140001059)

[a. Class Diagram 15](#_Toc140001060)

[b. Class Specifications 15](#_Toc140001061)

[c. Sequence Diagram(s) 16](#_Toc140001062)

[d. Database Queries 16](#_Toc140001063)

[5. <Common/Login> 17](#_Toc140001064)

[a. Class Diagram 17](#_Toc140001065)

[b. Class Specifications 17](#_Toc140001066)

[c. Sequence Diagram(s) 18](#_Toc140001067)

[d. Database Queries 18](#_Toc140001068)

[6. <Customer/Form> 19](#_Toc140001069)

[a. Class Diagram 19](#_Toc140001070)

[b. Class Specifications 19](#_Toc140001071)

[c. Sequence Diagram(s) 20](#_Toc140001072)

[d. Database Queries 20](#_Toc140001073)

[7. <Public/View Bird Course List> 20](#_Toc140001074)

[a. Class Diagram 20](#_Toc140001075)

[b. Class Specifications 20](#_Toc140001076)

[c. Sequence Diagram(s) 21](#_Toc140001077)

[d. Database queries 21](#_Toc140001078)

[8. <Public/View bird course detail> 22](#_Toc140001079)

[a. Class Diagram 22](#_Toc140001080)

[b. Class Specifications 22](#_Toc140001081)

[c. Sequence Diagram(s) 23](#_Toc140001082)

[d. Database Queries 23](#_Toc140001083)

[9. <Customer/Add a booking> 24](#_Toc140001084)

[a. Class Diagram 24](#_Toc140001085)

[b. Class Specifications 24](#_Toc140001086)

[c. Sequence Diagram(s) 26](#_Toc140001087)

[d. Database Queries 26](#_Toc140001088)

[10. <Staff/View list of pending bookings> 27](#_Toc140001089)

[a. Class Diagram 27](#_Toc140001090)

[b. Class Specifications 27](#_Toc140001091)

[c. Sequence Diagram(s) 28](#_Toc140001092)

[d. Database Queries 28](#_Toc140001093)

[11. <Staff/Pending Form> 29](#_Toc140001094)

[a. Class Diagram 29](#_Toc140001095)

[b. Class specification 29](#_Toc140001096)

[c. Sequence Diagram(s) 30](#_Toc140001097)

[d. Database Queries 30](#_Toc140001098)

[12. Staff/Processing Form 31](#_Toc140001099)

[a. Class Diagram 31](#_Toc140001100)

[b. Class specification 31](#_Toc140001101)

[c. Sequence Diagram(s) 32](#_Toc140001102)

[d. Database Queries 32](#_Toc140001103)

[13. Staff / Complete Form 33](#_Toc140001104)

[a. Class Diagram 33](#_Toc140001105)

[b. Class specification 33](#_Toc140001106)

[c. Sequence Diagram(s) 33](#_Toc140001107)

[d. Database Queries 34](#_Toc140001108)

[14. Staff / History Form 35](#_Toc140001109)

[a. Class Diagram 35](#_Toc140001110)

[b. Class specification 35](#_Toc140001111)

[c. Sequence Diagram(s) 36](#_Toc140001112)

[d. `Database Queries 36](#_Toc140001113)

[15. Staff / Detail Form 37](#_Toc140001114)

[a. Class Diagram 37](#_Toc140001115)

[b. Class specification 37](#_Toc140001116)

[c. Sequence Diagram(s) 38](#_Toc140001117)

[d. `Database Queries 38](#_Toc140001118)

[16. Trainer/ Payment Process 39](#_Toc140001119)

[a. Class Diagram 39](#_Toc140001120)

[b. Class specification 39](#_Toc140001121)

[c. Sequence Diagram(s) 40](#_Toc140001122)

[d. `Database Queries 40](#_Toc140001123)

[17. Trainer/ Detail Form 41](#_Toc140001124)

[a. Class Diagram 41](#_Toc140001125)

[b. Class specification 41](#_Toc140001126)

[c. Sequence Diagram(s) 42](#_Toc140001127)

[d. Database Queries 42](#_Toc140001128)

[18. Trainer/ List Form 43](#_Toc140001129)

[a. Class Diagram 43](#_Toc140001130)

[b. Class specification 43](#_Toc140001131)

[c. Sequence Diagram(s) 44](#_Toc140001132)

[d. `Database Queries 44](#_Toc140001133)

[19. Public / Blog List 45](#_Toc140001134)

[a. Class Diagram 45](#_Toc140001135)

[b. Class specification 45](#_Toc140001136)

[c. Sequence Diagram(s) 46](#_Toc140001137)

[d. `Database Queries 46](#_Toc140001138)

[20. Public / Blog Detail 46](#_Toc140001139)

[a. Class Diagram 46](#_Toc140001140)

[b. Class specification 47](#_Toc140001141)

[c. Sequence Diagram(s) 47](#_Toc140001142)

[d. `Database Queries 47](#_Toc140001143)

[21. Staff/ Manage Blog 48](#_Toc140001144)

[a. Class Diagram 48](#_Toc140001145)

[b. Class specification 48](#_Toc140001146)

[c. Sequence Diagram(s) 49](#_Toc140001147)

[d. `Database Queries 49](#_Toc140001148)

[22. Public / Register 50](#_Toc140001149)

[a. Class Diagram 50](#_Toc140001150)

[b. Class specification 50](#_Toc140001151)

[c. Sequence Diagram(s) 51](#_Toc140001152)

[d. Database Queries 51](#_Toc140001153)

[23. Customer / Change Password 52](#_Toc140001154)

[a. Class Diagram 52](#_Toc140001155)

[b. Class specification 52](#_Toc140001156)

[c. Sequence Diagram(s) 53](#_Toc140001157)

[d. Database Queries 53](#_Toc140001158)

[24. Public / Homepage 53](#_Toc140001159)

[a. Class Diagram 53](#_Toc140001160)

[b. Class specification 53](#_Toc140001161)

[c. Sequence Diagram(s) 53](#_Toc140001162)

[d. Database Queries 53](#_Toc140001163)

[25. Staff / Set trainer to a booking 53](#_Toc140001164)

[a. Class Diagram 53](#_Toc140001165)

[b. Class specification 53](#_Toc140001166)

[c. Sequence Diagram(s) 53](#_Toc140001167)

[d. Database Queries 53](#_Toc140001168)

[26. Staff / View detail of a pending booking 53](#_Toc140001169)

[a. Class Diagram 53](#_Toc140001170)

[b. Class specification 53](#_Toc140001171)

[c. Sequence Diagram(s) 53](#_Toc140001172)

[d. Database Queries 53](#_Toc140001173)

[27. Staff / View list of processing bookings 53](#_Toc140001174)

[a. Class Diagram 53](#_Toc140001175)

[b. Class specification 53](#_Toc140001176)

[c. Sequence Diagram(s) 53](#_Toc140001177)

[d. Database Queries 53](#_Toc140001178)

[28. Staff / View list of completed bookings 54](#_Toc140001179)

[a. Class Diagram 54](#_Toc140001180)

[b. Class specification 54](#_Toc140001181)

[c. Sequence Diagram(s) 54](#_Toc140001182)

[d. Database Queries 55](#_Toc140001183)

[29. Staff / Checkout a booking 55](#_Toc140001184)

[a. Class Diagram 55](#_Toc140001185)

[b. Class specification 55](#_Toc140001186)

[c. Sequence Diagram(s) 56](#_Toc140001187)

[d. Database Queries 56](#_Toc140001188)

[30. Trainer / Track a training progress 56](#_Toc140001189)

[a. Class Diagram 56](#_Toc140001190)

[b. Class specification 56](#_Toc140001191)

[c. Sequence Diagram(s) 57](#_Toc140001192)

[d. Database Queries 57](#_Toc140001193)

[31. Customer / View tracking progress of the booking 58](#_Toc140001194)

[a. Class Diagram 58](#_Toc140001195)

[b. Class specification 58](#_Toc140001196)

[c. Sequence Diagram(s) 58](#_Toc140001197)

[d. Database Queries 58](#_Toc140001198)

[32. Staff / View list of all courses 59](#_Toc140001199)

[a. Class Diagram 59](#_Toc140001200)

[b. Class specification 59](#_Toc140001201)

[c. Sequence Diagram(s) 59](#_Toc140001202)

[d. Database Queries 60](#_Toc140001203)

[33. Staff / View detail of a course 60](#_Toc140001204)

[a. Class Diagram 60](#_Toc140001205)

[b. Class specification 60](#_Toc140001206)

[c. Sequence Diagram(s) 60](#_Toc140001207)

[d. Database Queries 61](#_Toc140001208)

[34. Staff / Create a new course 61](#_Toc140001209)

[a. Class Diagram 61](#_Toc140001210)

[b. Class specification 61](#_Toc140001211)

[c. Sequence Diagram(s) 62](#_Toc140001212)

d[. Database Queries 62](#_Toc140001213)

[35. Staff / Update a course 63](#_Toc140001214)

[a. Class Diagram 63](#_Toc140001215)

[b. Class specification 63](#_Toc140001216)

[c. Sequence Diagram(s) 63](#_Toc140001217)

[d. Database Queries 64](#_Toc140001218)

[36. Staff / Delete a course 64](#_Toc140001219)

[a. Class Diagram 64](#_Toc140001220)

[b. Class specification 64](#_Toc140001221)

[c. Sequence Diagram(s) 65](#_Toc140001222)

[d. Database Queries 65](#_Toc140001223)

[37. Admin / View list of all account 65](#_Toc140001224)

[a. Class Diagram 65](#_Toc140001225)

[b. Class specification 65](#_Toc140001226)

[c. Sequence Diagram(s) 65](#_Toc140001227)

[d. Database Queries 65](#_Toc140001228)

[38. Admin / View detail of an account 65](#_Toc140001229)

[a. Class Diagram 65](#_Toc140001230)

[b. Class specification 65](#_Toc140001231)

[c. Sequence Diagram(s) 66](#_Toc140001232)

[d. Database Queries 66](#_Toc140001233)

[39. Admin / Create a new account 67](#_Toc140001234)

[a. Class Diagram 67](#_Toc140001235)

[b. Class specification 67](#_Toc140001236)

[c. Sequence Diagram(s) 68](#_Toc140001237)

[d. Database Queries 68](#_Toc140001238)

[40. Admin / Update an account Class Diagram 69](#_Toc140001239)

[a. Class specification 69](#_Toc140001240)

[b. Sequence Diagram(s) 69](#_Toc140001241)

[c. Database Queries 70](#_Toc140001242)

[41. Admin / Delete an account 70](#_Toc140001243)

[a. Class Diagram 70](#_Toc140001244)

[b. Class specification 70](#_Toc140001245)

[c. Sequence Diagram(s) 70](#_Toc140001246)

[d. Database Queries 71](#_Toc140001247)

[42. Staff / Dashboard to statistic revenue 71](#_Toc140001248)

[a. Class Diagram 71](#_Toc140001249)

[b. Class specification 71](#_Toc140001250)

[c. Sequence Diagram(s) 72](#_Toc140001251)

[d. Database Queries 72](#_Toc140001252)

# I. Overview

## 1. Code Packages

A picture containing diagram, line, rectangle, square

Description automatically generated

***Package descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 01 | Controller | Handle back-end, drive flow of code |
| 02 | Dal | Handle database |
| 03 | Model | Create object suitable database |
| 04 | Util | Support date |

## 2. Database Design

### a. Database Schema

A picture containing text, diagram, plan, technical drawing

Description automatically generated

# II. Code Designs

## 1. <Public/View Workshop Detail>

### a. Class Diagram

A picture containing text, line, plot, number

Description automatically generated

### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| **1** | **getWorkshopDetail** | **Use to take all the information of workshop which user click on.** |
|  |  |  |

### c. Sequence Diagram(s)

*A picture containing text, parallel, diagram, number

Description automatically generated*

### d. Database Queries

SELECT tbl\_course.course\_id , tbl\_course.trainer\_id, tbl\_course.staff\_id , tbl\_course.content , tbl\_course.category , tbl\_course.title,tbl\_course.price, tbl\_courseImg.img

FROM tbl\_course

JOIN tbl\_courseImg ON tbl\_course.course\_id = tbl\_courseImg.course\_id

WHERE tbl\_course.category = 'workshop'

AND tbl\_course.course\_id = ?

## 2. <Public/View Online Course Detail>

### a. Class Diagram

A picture containing text, line, plot, font

Description automatically generated

### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| **1** | **getCourseBYID** | **Use to collect detail information of online course when user click on.** |
|  |  |  |

### c. Sequence Diagram(s)

A picture containing text, parallel, number, diagram

Description automatically generated

### d. Database Queries

SELECT tbl\_course.course\_id , tbl\_course.trainer\_id, tbl\_course.staff\_id , tbl\_course.content , tbl\_course.category , tbl\_course.title,tbl\_course.price, tbl\_courseImg.img

FROM tbl\_course

JOIN tbl\_courseImg ON tbl\_course.course\_id = tbl\_courseImg.course\_id

WHERE tbl\_course.course\_id =?

## 3. <Customer/Payment for Online Course>

### a. Class Diagram

A screenshot of a computer program

Description automatically generated with low confidence

### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| **1** | **getCustomerInfo** | **Use to collect customer’s information by user\_id** |
| **2** | **getDate** | **Get the current date to update on Database** |
| **3** | **paymentOnlineCourse** | **Update new order on Database** |

### c. Sequence Diagram(s)

A picture containing text, parallel, diagram

Description automatically generated

### d. Database Queries

- SELECT customer\_id, user\_id, fullname, contact

FROM tbl\_customer

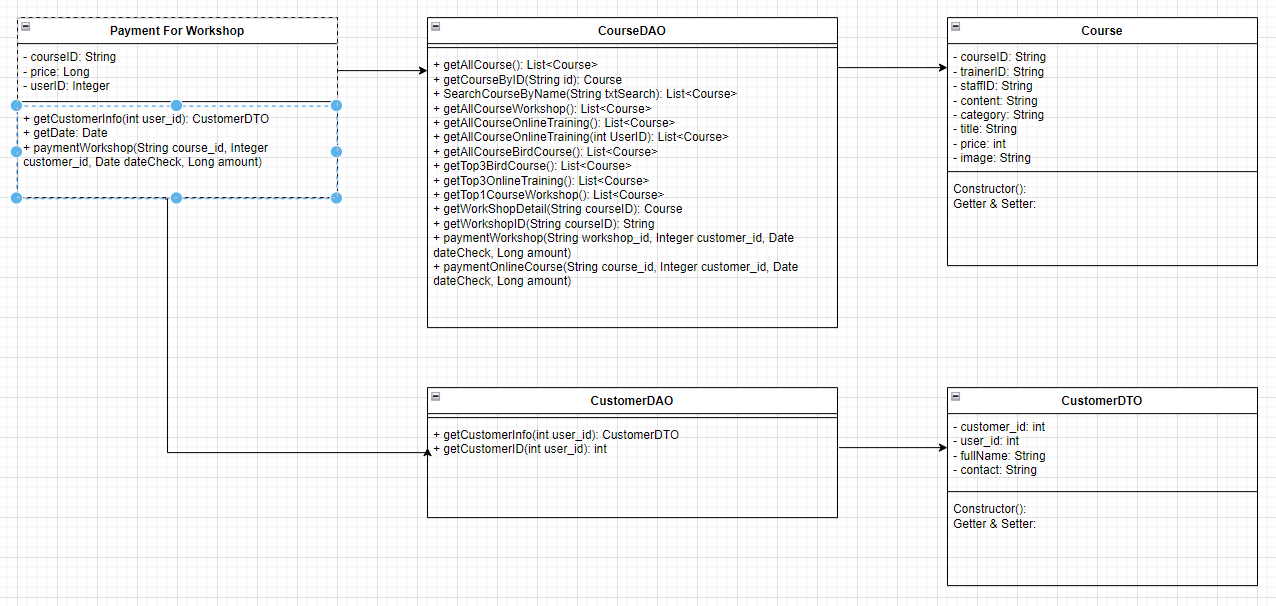
WHERE user\_id = ?

- INSERT INTO tbl\_courseBuy (course\_id, customer\_id, date, amount, payment\_id)

VALUES (?, ?, ?, ?, NULL)

## 4. <Customer/Payment for Workshop>

### a. Class Diagram



### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| **1** | **getCustomerInfo** | **Use to collect customer’s information by user\_id** |
| **2** | **getDate** | **Get the current date to update on Database** |
| **3** | **paymentForWorkshop** | **Update new order on Database** |

### c. Sequence Diagram(s)

A picture containing text, parallel, diagram, line

Description automatically generated

### d. Database Queries

- SELECT customer\_id, user\_id, fullname, contact

FROM tbl\_customer

WHERE user\_id = ?

- INSERT INTO tbl\_attendance (workshop\_id, customer\_id, dateCheck, amount, payment\_id)

VALUES (?, ?, ?, ?, NULL)

## 5. <Common/Login>

### a. Class Diagram

A screenshot of a computer

Description automatically generated with medium confidence

### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Login* | *Using to login into the system* |
|  | *Check Account Login* | *Using to check account in database* |

### c. Sequence Diagram(s)

### A picture containing text, diagram, screenshot, parallel Description automatically generatedd. Database Queries

String sql = "SELECT \* FROM users where email=? or username=?"

## 6. <Customer/Form>

### a. Class Diagram

A screenshot of a diagram

Description automatically generated with low confidence

### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *InputFormAppointment* | *Using to insert information that user input in form* |
| *02* | *GetTrainerUnavailable* | *Using to show all list of trainers that are not have requets in private consultation* |

### c. Sequence Diagram(s)

### A picture containing text, diagram, screenshot, line Description automatically generatedd. Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *InputFormAppointment* | *String sql = "insert into tbl\_appointment(consultation\_id , trainer\_id,customer\_id , date ,amount, payment\_id , note, address , type ,present\_price , status,tracking,Request\_trainer\_id)\n"*  *+ "values(?,NULL,?,?,NULL,4,? ,?,?,100,NULL,NULL,?) ";* |
| *02* | *GetTrainerUnavailable* | *String sql = "select \* from tbl\_trainer where status ='available'";* |

## 7. <Public/View Bird Course List>

### a. Class Diagram

A picture containing text, diagram, screenshot, line

Description automatically generated

### b. Class Specifications

* CourseDTO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CourseDTO | This class contains attribute of course like: courseID, trainerID, staffID, content, category, title, image, price, duration. It mapped to the tbl\_course of database |

* CourseDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getAllCourseBirdCourse | This method will access to database and get all courses that have category equals “bird training” |

### c. Sequence Diagram(s) A diagram of a course Description automatically generated with medium confidence

### d. Database queries

Insert into tbl\_Booking(trainer\_id, course\_id, customer\_id, bird\_id, start\_date, end\_date, status, amount, payment\_id, create\_date, preferred\_date, checkout\_date)

values(NULL, ?, ?, ?, NULL, NULL, NULL, ?, NULL, ?, ?, NULL)

## 8. <Public/View bird course detail>

### a. Class Diagram

A picture containing text, screenshot, diagram, line

Description automatically generated

### b. Class Specifications

* CourseDTO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CourseDTO | This class contains attribute of course like: courseID, trainerID, staffID, content, category, title, image, price, duration. It mapped to the tbl\_course of database |

* CourseDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 |  | This class will access to database and get all courses |

* DetailController

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | This class gets the ID of one’s course and call CourseDAO to query course that has that id |

### c. Sequence Diagram(s)

A diagram of a course

Description automatically generated with medium confidence

### d. Database Queries

select tbl\_course.course\_id , tbl\_course.trainer\_id, tbl\_course.staff\_id , tbl\_course.content , tbl\_course.category , tbl\_course.title, tbl\_course.price, tbl\_course.duration, tbl\_courseImg.img from tbl\_course

JOIN tbl\_courseImg ON tbl\_course.course\_id = tbl\_courseImg.course\_id

where tbl\_course.course\_id = ?

## 9. <Customer/Add a booking>

### a. Class Diagram

A screenshot of a computer

Description automatically generated with medium confidence

### b. Class Specifications

* FormCourseServlet class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | This class get information from the form user inputted. Then it will calls BirdDAO to create a new bird and BookingDAO to creat a new booking |

* BirdDTO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | BirdDTO | This class contains these attribute: bird\_id, name, type constructor, getter and setter. This class is mapped to tbl\_bird in the database |

* BirdDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | addNewBird | This method add new bird entity to database |
| 02 | generateBirdID | This method automatically generate an id |

* BookingDTO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | BookingDTO | This class contains attributes, constructor, getter setter. This class is mapped to tbl\_Booking in the databse |

* BookingDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | addNewBooking | This method create a new row in table tbl\_Booking with customer\_id, bird\_id, course\_id, amount and create\_date |

### c. Sequence Diagram(s)

A picture containing text, diagram, plan, parallel

Description automatically generated

### d. Database Queries

Insert into tbl\_Booking(trainer\_id, course\_id, customer\_id, bird\_id, start\_date, end\_date, status, amount, payment\_id, create\_date, preferred\_date, checkout\_date)

values(NULL, ?, ?, ?, NULL, NULL, NULL, ?, NULL, ?, ?, NULL)

## 10. <Staff/View list of pending bookings>

### a. Class Diagram

A screenshot of a computer

Description automatically generated with medium confidence

### b. Class Specifications

* BookingDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getPendingOrder | This method get all courses that have status equals NULL |

* BookingDTO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | BookingDTO | This class contains attributes, constructor, getter setter. This class is mapped to tbl\_Booking in the databse |

### c. Sequence Diagram(s)

A picture containing text, diagram, parallel, line

Description automatically generated

### d. Database Queries

SELECT c.fullname, b.bird\_id, b.type, co.title, bk.create\_date

FROM tbl\_Booking bk

INNER JOIN tbl\_customer c ON bk.customer\_id = c.customer\_id

INNER JOIN tbl\_bird b ON bk.bird\_id = b.bird\_id

INNER JOIN tbl\_course co ON bk.course\_id = co.course\_id

WHERE bk.status IS NULL

ORDER BY bk.create\_date ASC;

## 11. <Staff/Pending Form>

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getTrainerNameByRequestTrainerId()* | *Using to get TrainerName based on trainer\_id\_request* |
| *02* | *getAppointmentFormPending()* | *Using to show all list of appointment form pending which sent by customers* |

### Sequence Diagram(s)

A diagram of a project

Description automatically generated

### Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **SQL** |
| *01* | *getTrainerNameByRequestTrainerId()* | *String sql = " select a.Request\_trainer\_id ,t.fullName from tbl\_appointment as a \n"*  *+ " join tbl\_trainer as t ON a.Request\_trainer\_id = t.trainer\_id\n"*  *+ " where a.Request\_trainer\_id = ?";* |
| *02* | *getAppointmentFormPending()* | *String sql = "select customer.customer\_id , appointment.consultation\_id, customer.fullname as customer\_fullname, appointment.date, appointment.note, appointment.address, appointment.type, appointment.status ,appointment.Request\_trainer\_id,appointment.duration , appointment.DateSubmit,appointment.amount,appointment.fullname,appointment.gmail,appointment.contact\n"*  *+ " from tbl\_appointment appointment \n"*  *+ " join tbl\_customer customer on customer.customer\_id = appointment.customer\_id\n"*  *+ " WHERE appointment.status IS NULL ORDER BY appointment.consultation\_id DESC";* |

## 12. Staff/Processing Form

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getTrainerNameByRequestTrainerId()* | *Using to get TrainerName based on trainer\_id\_request* |
| *02* | *getAppointmentFormProcesing ()* | *Using to show all list of appointment form processing which are apprroved by staff* |

### Sequence Diagram(s)

A diagram with text and arrows

Description automatically generated

### Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **SQL** |
| *01* | *getTrainerNameByRequestTrainerId()* | *String sql = " select a.Request\_trainer\_id ,t.fullName from tbl\_appointment as a \n"*  *+ " join tbl\_trainer as t ON a.Request\_trainer\_id = t.trainer\_id\n"*  *+ " where a.Request\_trainer\_id = ?";* |
| *02* | *getAppointmentFormProcessing()* | *String sql = " SELECT appointment.customer\_id, appointment.consultation\_id, customer.fullname AS customer\_fullname, appointment.date, appointment.note, appointment.address, appointment.type, appointment.status,appointment.Request\_trainer\_id, appointment.duration , appointment.DateSubmit,appointment.amount,appointment.fullname,appointment.gmail,appointment.contact\n"*  *+ " FROM tbl\_appointment appointment\n"*  *+ " JOIN tbl\_customer customer ON customer.customer\_id = appointment.customer\_id\n"*  *+ " WHERE appointment.Request\_trainer\_id IS NOT NULL AND appointment.status IS NOT NULL AND trainer\_id IS NOT NULL and appointment.tracking IS NULL ORDER BY CONVERT(INT, appointment.consultation\_id) DESC;";* |

## 13. Staff / Complete Form

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getTrainerNameByRequestTrainerId()* | *Using to get TrainerName based on trainer\_id\_request* |
| *02* | *getAppointmentFormCompleted()* | *Using to show all list of appointment form completef which are completed by trainer* |

### Sequence Diagram(s)

A diagram of a diagram

Description automatically generated

### Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **SQL** |
| *01* | *getTrainerNameByRequestTrainerId()* | *String sql = " select a.Request\_trainer\_id ,t.fullName from tbl\_appointment as a \n"*  *+ " join tbl\_trainer as t ON a.Request\_trainer\_id = t.trainer\_id\n"*  *+ " where a.Request\_trainer\_id = ?";* |
| *02* | *getAppointmentFormProcessing()* | *String sql = " SELECT appointment.customer\_id, appointment.consultation\_id, customer.fullname AS customer\_fullname, appointment.date, appointment.note, appointment.address, appointment.type, appointment.status,appointment.Request\_trainer\_id, appointment.duration , appointment.DateSubmit,appointment.amount,appointment.fullname,appointment.gmail,appointment.contact\n"*  *+ " FROM tbl\_appointment appointment\n"*  *+ " JOIN tbl\_customer customer ON customer.customer\_id = appointment.customer\_id\n"*  *+ " WHERE appointment.tracking = 'complete' and appointment.history IS NULL ORDER BY CONVERT(INT, appointment.consultation\_id) DESC;";* |

## 14. Staff / History Form

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getTrainerNameByRequestTrainerId()* | *Using to get TrainerName based on trainer\_id\_request* |
| *02* | *getAppointmentFormHistory()* | *Using to show all list of appointment form completef which are completed by trainer* |

### Sequence Diagram(s)

A diagram of a project

Description automatically generated

### `Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **SQL** |
| *01* | *getTrainerNameByRequestTrainerId()* | *String sql = " select a.Request\_trainer\_id ,t.fullName from tbl\_appointment as a \n"*  *+ " join tbl\_trainer as t ON a.Request\_trainer\_id = t.trainer\_id\n"*  *+ " where a.Request\_trainer\_id = ?";* |
| *02* | *getAppointmentFormHiostory()* | *String sql = " SELECT appointment.customer\_id, appointment.consultation\_id, customer.fullname AS customer\_fullname, appointment.date, appointment.note, appointment.address, appointment.type, appointment.status,appointment.Request\_trainer\_id, appointment.duration , appointment.DateSubmit,appointment.amount,appointment.fullname,appointment.gmail,appointment.contact\n"*  *+ " FROM tbl\_appointment appointment\n"*  *+ " JOIN tbl\_customer customer ON customer.customer\_id = appointment.customer\_id\n"*  *+ " WHERE /\*appointment.consultation\_id = 1 AND\*/ appointment.history = 'yes' ORDER BY CONVERT(INT, appointment.consultation\_id) DESC;";* |

## 15. Staff / Detail Form

### Class Diagram

A diagram of a program

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAppointmentDetailCompleted()* | *Using to get appointment form detail* |
| *02* | *getTrainerByID(in trainerID)* | *Using to get trainer information by trainerID* |

### Sequence Diagram(s)

A diagram of a project

Description automatically generated

### `Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAppointmentDetailCompleted()* | *String sql = " SELECT appointment.customer\_id, appointment.consultation\_id, customer.fullname AS customer\_fullname, appointment.date, appointment.note, appointment.address, appointment.type, appointment.status,appointment.Request\_trainer\_id,appointment.duration , appointment.DateSubmit,appointment.amount,appointment.fullname,appointment.gmail,appointment.contact\n"*  *+ " FROM tbl\_appointment appointment\n"*  *+ " JOIN tbl\_customer customer ON customer.customer\_id = appointment.customer\_id\n"*  *+ " WHERE appointment.consultation\_id = ? AND appointment.tracking = 'complete' and appointment.history IS NULL";* |
| *02* | *getTrainerByID(in trainerID)* | *String sql = "SELECT T.trainer\_id, T.user\_id, T.fullname, T.achievement, T.img, T.status, T.contact,\n"*  *+ " STRING\_AGG(K.skill\_id, ',') AS skill\_ids,\n"*  *+ " STRING\_AGG(S.name, ',') AS skill\_names\n"*  *+ " FROM tbl\_trainer AS T\n"*  *+ " JOIN tbl\_skillTrainer AS K ON T.trainer\_id = K.trainer\_id\n"*  *+ " JOIN tbl\_skill AS S ON S.skill\_id = K.skill\_id\n"*  *+ " where T.trainer\_id = ?\n"*  *+ " GROUP BY T.trainer\_id, T.user\_id, T.fullname, T.achievement, T.img, T.status, T.contact";* |

## 16. Trainer/ Payment Process

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetListOfBlog()* | *Using to get detail form appointment in database and render in trainer page* |
| *02* | *UpdateTrainerAmount()* | *Using to complelte process training and payment* |
| *03* | *updateTrainerWhenPayment()* | *Using to complete process training and change status of trainer to other tasks* |

### Sequence Diagram(s)

A diagram of a diagram

Description automatically generated

### `Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetListOfBlog()* | *String sql = "SELECT appointment.customer\_id, appointment.consultation\_id, customer.fullname AS customer\_fullname, appointment.date, appointment.note, appointment.address, appointment.type, appointment.status,appointment.Request\_trainer\_id, appointment.duration , appointment.DateSubmit,appointment.amount,appointment.fullname,appointment.gmail,appointment.contact,appointment.tracking\n"*  *+ " FROM tbl\_appointment appointment\n"*  *+ " JOIN tbl\_customer customer ON customer.customer\_id = appointment.customer\_id\n"*  *+ " WHERE appointment.consultation\_id = ? AND appointment.Request\_trainer\_id IS NOT NULL AND appointment.status IS NOT NULL AND trainer\_id IS NOT NULL and appointment.amount = 0 ";* |
| *02* | *UpdateTrainerAmount()* | *String appointmentUpdateQuery = "update tbl\_appointment \n"*  *+ " set Tracking = 'complete'\n"*  *+ " where consultation\_id = ?";*  *String trainerUpdateQuery = "update tbl\_trainer \n"*  *+ " set status = 'available'\n"*  *+ " where trainer\_id =?";* |
| *03* | *updateTrainerWhenPayment()* | *"update tbl\_appointment set amount = ? where consultation\_id = ?";* |

## 17. Trainer/ Detail Form

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetListOfBlog()* | *Using to get detail form appointment in database and render in trainer page* |
|  |  |  |

### Sequence Diagram(s)

A diagram of a diagram

Description automatically generated

### Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetListOfBlog()* | *String sql = "SELECT appointment.customer\_id, appointment.consultation\_id, customer.fullname AS customer\_fullname, appointment.date, appointment.note, appointment.address, appointment.type, appointment.status,appointment.Request\_trainer\_id, appointment.duration , appointment.DateSubmit,appointment.amount,appointment.fullname,appointment.gmail,appointment.contact,appointment.tracking\n"*  *+ " FROM tbl\_appointment appointment\n"*  *+ " JOIN tbl\_customer customer ON customer.customer\_id = appointment.customer\_id\n"*  *+ " WHERE appointment.consultation\_id = ? AND appointment.Request\_trainer\_id IS NOT NULL AND appointment.status IS NOT NULL AND trainer\_id IS NOT NULL and appointment.amount = 0 ";* |
|  |  |  |

## 18. Trainer/ List Form

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetListOfBlog()* | *Using to get List of form appointment in database and render in trainer page* |
|  |  |  |

### Sequence Diagram(s)

A diagram of a project

Description automatically generated

### `Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAppointmentListBYTrainerID* | *String sql = " SELECT appointment.customer\_id, appointment.consultation\_id, customer.fullname AS customer\_fullname, appointment.date, appointment.note, appointment.address, appointment.type, appointment.status,appointment.Request\_trainer\_id, appointment.duration , appointment.DateSubmit,appointment.amount,appointment.fullname,appointment.gmail,appointment.contact\n"*  *+ " FROM tbl\_appointment appointment\n"*  *+ " JOIN tbl\_customer customer ON customer.customer\_id = appointment.customer\_id\n"*  *+ " WHERE appointment.trainer\_id = ? AND appointment.Request\_trainer\_id IS NOT NULL AND appointment.status IS NOT NULL AND trainer\_id IS NOT NULL and appointment.amount = 0 ORDER BY appointment.consultation\_id DESC ";* |
|  |  |  |

## 19. Public / Blog List

### Class Diagram

A screenshot of a diagram

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetListOfBlog()* | *Using to get List of blog in database and render in blog page* |
| *02* | *SearchBlogbyID()* | *Using to searh blog based name.* |

### Sequence Diagram(s)

A diagram of a diagram

Description automatically generated

### `Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetListOfBlog()* | *select \* from tbl\_blog where status = 'available'* |
| *02* | *SearchBlogbyID()* | *select \* from tbl\_blog where status = 'available' and id =?* |

## 20. Public / Blog Detail

### Class Diagram

A diagram of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetDetailOfBlog()* | *Using to get Detail of blog in database and render in blog page* |
|  |  |  |

### Sequence Diagram(s)

A diagram of a diagram

Description automatically generated

### `Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetDetailOfBlog()* | *select\* from tbl\_blog where id =? and status = 'available'* |
|  |  |  |

## 21. Staff/ Manage Blog

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetListOfBlog()* | *Using to get the list of blog form database to to the staff view* |
| *02* | *getStaffID()* | *Using to vaidate that the user with role staff to access blog page to manage* |
| *03* | *updateBlog()* | *Using to update the detail of blog (contain all of information except id, status)* |
| *04* | *createBlog()* | *Using to create the new Blog* |
| *05* | *deleteBlog()* | *Using to delete the blog(set blog to unavailable)* |
| *06* | *GenerateBlogID()* | *Using to generate automatically id of blog* |
| *07* | *getDetailofBlog()* | *Using to get detail of blog* |

### Sequence Diagram(s)

A diagram of a company

Description automatically generated

### `Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **SQL** |
| *01* | *GetListOfBlog()* | *select \* from tbl\_blog where status = 'available'* |
| *02* | *getStaffID()* | *Using to vaidate that the user with role staff to access blog page to manage* |
| *03* | *updateBlog()* | *UPDATE tbl\_blog SET date = ?, thumbnail = ?, title = ?, author = ?, introduction = ?, content1 = ?, content2 = ?, inconclusion = ?, contentIMG = ?, briefinfo = ? WHERE id = ?* |
| *04* | *createBlog()* | *"insert into tbl\_blog(id,staff\_id ,date,thumbnail , title,author , introduction,content1, content2,contentIMG , briefinfo,status)\n"*  *+ "values(?,18,?,?,?,?,?,?,?,?,?,'available')";* |
| *05* | *deleteBlog()* | *update tbl\_blog \n"*  *+ "set status = 'unavailable'\n"*  *+ "where id = ?* |
| *06* | *GenerateBlogID()* | *SELECT \* FROM tbl\_blog WHERE id = (SELECT MAX(TRY\_CAST(id AS INT))FROM tbl\_blog)* |
| *07* | *getDetailofBlog()* | *select\* from tbl\_blog where id =? and status = 'available'* |

## 22. Public / Register

### Class Diagram

A diagram of a server

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GenerateUserID()* | *Using to automatically generate user\_id plus one* |
| *02* | *GenerateCustomerID()* | *Using to automatically generate customer)id plus one* |
| *03* | *CheckAcc()* | *Using to check username and password in the database to check username and password that guest input* |
| *04* | *isUserNameEmailistaken()* | *Using to check Username and Email in the database to check username and email that user input* |
| *05* | *ValidateOtp()* | *Using to check validate otp and send otp to email* |
| *07* | *Registration()* | *Using to register to create a new account in out system* |

### Sequence Diagram(s)

A diagram of a diagram

Description automatically generated

### Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GenerateUserID()* | *SELECT \* FROM tbl\_user WHERE user\_id = (SELECT MAX(TRY\_CAST(user\_id AS INT))FROM tbl\_user)";* |
| *02* | *GenerateCustomerID()* | *" SELECT \* FROM tbl\_customer WHERE customer\_id = (SELECT MAX(TRY\_CAST(customer\_id AS INT))FROM tbl\_customer)"* |
| *03* | *CheckAcc()* | *select \* from tbl\_user where username = ? or gmail =?* |
| *04* | *isUserNameEmailistaken()* | *SELECT COUNT(\*) FROM tbl\_user WHERE username = ?* |
| *05* | *ValidateOtp()* |  |
| *07* | *Registration()* | *String sql\_user = "INSERT INTO tbl\_user (user\_id, username, password, role , gmail) \n"*  *+ " VALUES (?, ?,?,'customer',?); ";*  *//Update in tbl\_customer*  *String sql\_customer = "INSERT INTO tbl\_customer (customer\_id, user\_id, fullName, contact,img) \n"*  *+ " VALUES (?, ?, ?,?,NULL );";* |

## 23. Customer / Change Password

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *CheckOTP* | *Validate OTP that user input and OTP that system send to email* |
| *02* | *ForgotPassword()* | *Using to send otp for email that user input* |
| *03* | *NewPassword()* | *Using to change password* |

### Sequence Diagram(s)

A diagram of a diagram

Description automatically generated

### Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **SQL** |
| *01* | *CheckOTP* |  |
| *02* | *ForgotPassword()* |  |
| *03* | *NewPassword()* | *String sql = ("update tbl\_user\n"*  *+ "set password = ?\n"*  *+ "where gmail = ? ");* |

## 24. Public / Homepage

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAllFeeedBack()* | *Using to get all feeback of customers about my system bird training center* |
| *02* | *getTop3BirdCourse()* | *Using to render all top 3 bird course that popular most to homepage* |
| *03* | *getTop3OnlineTraining()* | *Using to render all top 3 online course that popular most to homepage* |
| *04* | *getTop1CourseWorkshop()* | *Using to render all top 1 workshop that popular most to homepage* |

### Sequence Diagram(s)

A diagram of a diagram

Description automatically generated

### Database Queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAllFeeedBack()* | *String sql = "select MAX(f.comment), c.fullname, c.img \n"*  *+ " from tbl\_feedbacks as f \n"*  *+ " join tbl\_customer as c\n"*  *+ " on f.customer\_id = c.customer\_id\n"*  *+ " GRoup by c.fullname, c.img ";* |
| *02* | *getTop3BirdCourse()* | *String sql = "select TOP 3 tbl\_course.course\_id , tbl\_course.trainer\_id, tbl\_course.staff\_id , tbl\_course.content , tbl\_course.category , tbl\_course.title,tbl\_course.price, tbl\_courseImg.img from tbl\_course JOIN tbl\_courseImg ON tbl\_course.course\_id = tbl\_courseImg.course\_id\n"*  *+ " where tbl\_course.category = 'bird training'";* |
| *03* | *getTop3OnlineTraining()* | *String sql = "select Top 3 tbl\_course.course\_id , tbl\_course.trainer\_id, tbl\_course.staff\_id , tbl\_course.content , tbl\_course.category , tbl\_course.title,tbl\_course.price, tbl\_courseImg.img from tbl\_course JOIN tbl\_courseImg ON tbl\_course.course\_id = tbl\_courseImg.course\_id\n"*  *+ "where tbl\_course.category = 'online'";* |
| *04* | *getTop1CourseWorkshop()* | *String sql = "select Top 1 tbl\_course.course\_id , tbl\_course.trainer\_id, tbl\_course.staff\_id , tbl\_course.content , tbl\_course.category , tbl\_course.title,tbl\_course.price, tbl\_courseImg.img from tbl\_course JOIN tbl\_courseImg ON tbl\_course.course\_id = tbl\_courseImg.course\_id\n"*  *+ "where tbl\_course.category = 'workshop'";* |

## 25. Staff / Set trainer to a booking

### Class Diagram

A diagram of a system

Description automatically generated

### Class specification

1. SetTrainerToBooking

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This class will get the action, if the action is “approve” this class will call setTrainerToBooking(), if the action is “deny” this class will call denyBooking(). |

1. BookingDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | setTrainerToBooking() | This function sets a trainer that is available to the given booking |
| 02 | denyBooking() | This function sets a status “denied” to booking |

### Sequence Diagram(s)

A diagram of a system

Description automatically generated

### Database Queries

* setTrainerToBooking():

Update tbl\_Booking Set trainer\_id = ?, status = 'approved'\n

Where bird\_id = ?

* denyBooking():

Update tbl\_Booking Set status = 'denied'

Where bird\_id = ?

## 26. Staff / View detail of a pending booking

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

1. BookingDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getCourseByBirdID | This method get information about the course have given id |

1. PendingOrderInfo class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | This method get parameter has value of bird\_id and then call BookingDAO.getCourseByID. After getting result set, it will redirect to staff\_formBirdCourse\_detail.jsp |

### Sequence Diagram(s)

A diagram of a flowchart

Description automatically generated

### Database Queries

SELECT c.fullname AS customer\_name, b.name AS bird\_name, b.type AS bird\_type, co.title, co.price, bk.create\_date, bk.preferred\_date

FROM tbl\_Booking bk

INNER JOIN tbl\_customer c ON bk.customer\_id = c.customer\_id

INNER JOIN tbl\_bird b ON bk.bird\_id = b.bird\_id

INNER JOIN tbl\_course co ON bk.course\_id = co.course\_id

WHERE bk.bird\_id = ?;

## 27. Staff / View list of processing bookings

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

* BookingDTO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | BookingDTO | This class contains attributes, constructor, getter setter. This class is mapped to tbl\_Booking in the database |

* BookingDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 02 | getProcessingBooking() | This class will access to database and get all booking that has status approved/in-training/checkin and order it ascendingly by create\_date |

### Sequence Diagram(s)

A diagram of a company

Description automatically generated

### Database Queries

SELECT c.fullname AS customer\_fullname, cr.title AS course\_title, cr.price, t.fullname AS trainer\_fullname, b.bird\_id, b.name AS bird\_name, b.type AS bird\_type, bo.status, bo.start\_date, bo.end\_date, bo.create\_date

FROM tbl\_customer c

INNER JOIN tbl\_Booking bo ON c.customer\_id = bo.customer\_id\n"

INNER JOIN tbl\_course cr ON bo.course\_id = cr.course\_id\n"

INNER JOIN tbl\_trainer t ON bo.trainer\_id = t.trainer\_id\n"

INNER JOIN tbl\_bird b ON bo.bird\_id = b.bird\_id\n"

WHERE bo.status IN ('checkin', 'approved', 'in-training')\n"

ORDER BY bo.create\_date ASC

## 28. Staff / View list of completed bookings

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

* BookingDTO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | BookingDTO | This class contains attributes, constructor, getter setter. This class is mapped to tbl\_Booking in the database |

* BookingDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 02 | getProcessingBooking() | This class will access to database and get all booking that has status approved/in-training/checkin and order it ascendingly by create\_date |

### Sequence Diagram(s)

A diagram of a company

Description automatically generated

### Database Queries

SELECT c.fullname AS customer\_fullname, cr.title, cr.price, t.fullname AS trainer\_fullname, b.name AS bird\_name, b.type AS bird\_type, bk.start\_date, bk.end\_date, bk.checkout\_date, b.bird\_id\n"

FROM tbl\_Booking bk\n"

INNER JOIN tbl\_customer c ON bk.customer\_id = c.customer\_id

INNER JOIN tbl\_trainer t ON bk.trainer\_id = t.trainer\_id

INNER JOIN tbl\_course cr ON bk.course\_id = cr.course\_id

INNER JOIN tbl\_bird b ON bk.bird\_id = b.bird\_id

WHERE bk.status = 'completed'\n"

ORDER BY bk.checkout\_date DESC;

## 29. Staff / Checkout a booking

### Class Diagram

A black line on a white background

Description automatically generated

### Class specification

* CheckOutBirdCourse

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function will get the parameter “action” and then call checkOutBooking() |

* BookingDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | checkOutBooking() | This function will set status “checkout” to the given booking |

### Sequence Diagram(s)

A diagram of a flowchart

Description automatically generated

### Database Queries

Update tbl\_Booking Set status = 'checkout', checkout\_date = ?\n

Where bird\_id = ?

## 30. Trainer / Track a training progress

### Class Diagram

A black line on a white background

Description automatically generated

### Class specification

1. Trainer\_Tracking

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function will call checkOutBooking() and dispatch to “trainer\_trackingBirdCourse.jsp” |

1. BookingDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getCourseByTrainerID() | This function gets the booking based on the trainerID |

### Sequence Diagram(s)

A diagram of a flowchart

Description automatically generated

### Database Queries

SELECT c.course\_id, c.title, ci.img, b.bird\_id, b.name, b.type, bk.start\_date, bk.end\_date, bk.status

FROM tbl\_Booking bk

INNER JOIN tbl\_course c ON bk.course\_id = c.course\_id

INNER JOIN tbl\_courseImg ci ON c.course\_id = ci.course\_id

INNER JOIN tbl\_bird b ON bk.bird\_id = b.bird\_id

WHERE bk.trainer\_id = ? AND bk.status IN ('approved', 'checkin', 'in-training');

## 31. Customer / View tracking progress of the booking

### Class Diagram

A black line on a white background

Description automatically generated

### Class specification

* Customer\_Tracking

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function will call checkOutBooking() and dispatch to “customer\_trackingBirdCourse.jsp” |

* BookingDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getCourseByCustomerID() | This function gets the booking based on the customerID |

### Sequence Diagram(s)

A diagram of a customer tracking system

Description automatically generated

### Database Queries

SELECT c.course\_id, c.title, ci.img, t.trainer\_id, t.fullname, t.img AS trainer\_img, b.bird\_id, b.name, b.type, bk.start\_date, bk.end\_date, bk.status

FROM tbl\_Booking bk

INNER JOIN tbl\_customer cust ON bk.customer\_id = cust.customer\_id

JOIN tbl\_course c ON bk.course\_id = c.course\_id

INNER JOIN tbl\_courseImg ci ON c.course\_id = ci.course\_id

INNER JOIN tbl\_trainer t ON bk.trainer\_id = t.trainer\_id

INNER JOIN tbl\_bird b ON bk.bird\_id = b.bird\_id

WHERE bk.customer\_id = ? AND bk.status IN ('approved', 'checkin', 'in-training', 'completed');

## 32. Staff / View list of all courses

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

* courseDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getCourses() | This function gets all the courses that have status “available” |

### Sequence Diagram(s)

A diagram of a computer

Description automatically generated

### Database Queries

select course\_id, content, category, title, price, duration

from tbl\_course

where status='available'

## 33. Staff / View detail of a course

### Class Diagram

A black line on a white background

Description automatically generated

### Class specification

* CourseDetail

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function call getCourseById() and dispatch to “Staff\_updateCourse.jsp” |

* courseDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getCourseById | This function gets information of a course based on its ID |

### Sequence Diagram(s)

A diagram of a course

Description automatically generated

### Database Queries

select tbl\_course.course\_id , tbl\_course.trainer\_id, tbl\_course.staff\_id , tbl\_course.content , tbl\_course.category , tbl\_course.title, tbl\_course.price, tbl\_course.duration, tbl\_courseImg.img from tbl\_course

JOIN tbl\_courseImg ON tbl\_course.course\_id = tbl\_courseImg.course\_id\n"

where tbl\_course.course\_id =?

## 34. Staff / Create a new course

### Class Diagram

A computer screen shot of a computer

Description automatically generated

### Class specification

* AddCourse

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function call getCourseById() and dispatch to “Staff\_updateCourse.jsp” |

* courseDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | generateCourseID() | This functions generates a new courseID |
| *02* | createBirdCourse() | This function will add a new bird course with these information: course\_id, content, category, title, price, duration |
| *03* | generateCourseImgID() | This function generates a new courseImgID |
| *04* | createBirdCourseImg() | This functions add a new course img with these information: img\_id, course\_id, thumbnailBase64 |

### Sequence Diagram(s)

A diagram of a flowchart

Description automatically generated

### Database Queries

* generateCourseID():

SELECT \* FROM tbl\_course WHERE course\_id = (SELECT MAX(TRY\_CAST(course\_id AS INT))FROM tbl\_course)

* createBirdCourse():

insert into tbl\_course(course\_id, trainer\_id, staff\_id, content, category, title, price, duration, end\_enroll\_date, start\_date, status)

values(?,NULL,NULL,?,?,?,?,?,NULL,NULL,'available')

* createBirdCourseImg():

SELECT \* FROM tbl\_courseImg WHERE id = (SELECT MAX(TRY\_CAST(id AS INT))FROM tbl\_courseImg)

* createBirdCourseImg():

insert into tbl\_courseImg(id, course\_id, img) values(?,?,?)

## 35. Staff / Update a course

### Class Diagram

A black line on a white background

Description automatically generated

### Class specification

* UpdateCourse

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function get parameters and then call courseDAO |

* courseDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | UpdateBirdCourse() | This functions update new information |
| *02* | UpdateBirdCourseImg() | This function update a new img to table courseImg |

### Sequence Diagram(s)

A diagram of a process

Description automatically generated

### Database Queries

* updateBirdCourse();

Update tbl\_course set content=?, category=?, title=?, price=?, duration=?

where course\_id= ?

* updateBirdCourseImg():

update tbl\_courseImg set img=? where course\_id = ?

## 36. Staff / Delete a course

### Class Diagram

A black and grey line

Description automatically generated

### Class specification

1. UpdateCourse

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function get parameters and then call courseDAO |

1. courseDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | deleteCourse() | This functions wil change the course status to “unavailable” based on its ID |

### Sequence Diagram(s)

A diagram of a process

Description automatically generated

### Database Queries

1. update tbl\_course set status='unavailable' where course\_id = ?

## 37. Admin / View list of all account

### Class Diagram

A close-up of a sign

Description automatically generated

### Class specification

* UserDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getAllAccount() | This functions gets all account from the database |

### Sequence Diagram(s)

A diagram of a company

Description automatically generated

### Database Queries

* SELECT u.user\_id, u.username, COALESCE(c.fullname, t.fullname, a.fullname, s.fullname) AS fullname, u.role
* FROM tbl\_user u
* LEFT JOIN tbl\_customer c ON u.user\_id = c.user\_id
* LEFT JOIN tbl\_trainer t ON u.user\_id = t.user\_id
* LEFT JOIN tbl\_admin a ON u.user\_id = a.user\_id
* LEFT JOIN tbl\_staff s ON u.user\_id = s.user\_id

## 38. Admin / View detail of an account

### Class Diagram

A black and white line

Description automatically generated

### Class specification

* AccountDetail

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function call getAccountByID () and dispatch to “Admin\_updateAccount.jsp” |

* courseDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getAccountById | This function gets information of an account based on its ID |

### Sequence Diagram(s)

A diagram of a computer system

Description automatically generated

### Database Queries

SELECT u.user\_id, u.username, u.password, u.role, u.gmail

COALESCE(c.fullname, t.fullname, s.fullname, a.fullname) AS fullname,

COALESCE(c.contact, t.contact, s.contact, a.contact) AS contact

FROM tbl\_user u

LEFT JOIN tbl\_customer c ON c.user\_id = u.user\_id

LEFT JOIN tbl\_trainer t ON t.user\_id = u.user\_id

LEFT JOIN tbl\_staff s ON s.user\_id = u.user\_id

LEFT JOIN tbl\_admin a ON a.user\_id = u.user\_id

WHERE u.user\_id = ?

## 39. Admin / Create a new account

### Class Diagram

A diagram of a server

Description automatically generated

### Class specification

* AddAccountServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function get request parameters and call userDAO to add a new account |

* userDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | generateStaffID () | This functions generates a new staffID |
| *02* | addNewStaff() | This function will add a new staff |
| *03* | generateAdminID() | This function generates a new adminID |
| *04* | addNewAdmin() | This functions add a new admin |

* customerDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | generateCustomerID() | This functions generates a new customerID |
| *02* | addNewCustomer() | This function will add a new staff |

* trainerDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | generateTrainerID() | This functions generates a new trainerID |
| *02* | addNewAdmin() | This function will add a new staff |

### Sequence Diagram(s)

A diagram of a computer flow

Description automatically generated

### Database Queries

* generateCustomerID():

SELECT \* FROM tbl\_customer WHERE customer\_id = (SELECT MAX(TRY\_CAST(customer\_id AS INT))FROM tbl\_customer)

* addNewCustomer():

insert into tbl\_customer(customer\_id, user\_id , fullname, contact, img)

values(?,?,?,?,NULL)

* generateTrainerID()

SELECT \* FROM tbl\_trainer WHERE trainer\_id = (SELECT MAX(TRY\_CAST(trainer\_id AS INT))FROM tbl\_trainer)

* addNewTrainer()

insert into tbl\_trainer(trainer\_id, user\_id, fullname, achievement, img, status, contact)

values(?,?,?,NULL,?,'available',?)

* generateStaffID()

SELECT \* FROM tbl\_staff WHERE staff\_id = (SELECT MAX(TRY\_CAST(staff\_id AS INT))FROM tbl\_staff)

* addNewStaff()

insert into tbl\_staff(staff\_id, user\_id, fullname, contact) values(?,?,?,?)

* generateAdminID()

SELECT \* FROM tbl\_admin WHERE admin\_id = (SELECT MAX(TRY\_CAST(admin\_id AS INT))FROM tbl\_admin)

* addNewAdmin()

insert into tbl\_admin(admin\_id, user\_id, fullname, contact) values(?,?,?,?)

## 40. Admin / Update an account Class Diagram

A diagram of a math problem

Description automatically generated

### Class specification

* UpdateAccount

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function get parameters and then call userDAO |

* userDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | update() | This function updates new information based on the role |
| *02* | updateUser() | This function updates new information to table tbl\_user |

### Sequence Diagram(s)

A diagram of a system

Description automatically generated

### Database Queries

* update():

update " + table + " set fullname=?, contact=? where user\_id=?

* updateUser():

update tbl\_user set password=?, role=?, gmail=? "

+ "where user\_id=?

## 41. Admin / Delete an account

### Class Diagram

A black arrow pointing to the right

Description automatically generated

### Class specification

* UpdateAccount

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function get parameters and then call userDAO |

* userDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | delete() | This function delete the user based on its id and role |

### Sequence Diagram(s)

A diagram of a system

Description automatically generated

### Database Queries

DELETE FROM " + table + " WHERE user\_id = ?

## 42. Staff / Dashboard to statistic revenue

### Class Diagram

A screenshot of a computer

Description automatically generated

### Class specification

* Dashboard

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | processRequest() | This function call DashboardDAO and dispatch to “Staff\_Dashboard.jsp” |

* DashboardDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getRevenueOfBirdCourse() | This function gets the revenue of bird course |
| *02* | getRevenueOfWorkshop() | This function gets the revenue of workshop |
| *03* | getRevenueOfOnlineCourse() | This function gets the revenue of online course |
| *04* | getRevenueOfConsultation() | This function gets the revenue of consultation |
| *01* | getTotalRevenue() | This function gets the revenue of 4 services and total |

### Sequence Diagram(s)

A diagram of a system

Description automatically generated

### Database Queries

1. getRevenueOfBirdCourse():

SELECT SUM(amount) AS total\_amount

FROM tbl\_Booking

WHERE MONTH(checkout\_date) = ? AND status='checkout'

1. getRevenueOfWorkshop()

SELECT SUM(amount) AS total\_amount

FROM tbl\_attendance\n

WHERE MONTH(dateCheck) = ?;

1. getRevenueOfOnlineCourse()

SELECT SUM(amount) AS total\_amount\n

FROM tbl\_courseBuy\n

WHERE MONTH(date) = ?;

1. getRevenueOfConsultation()

SELECT SUM(amount) AS total\_amount\n

FROM tbl\_appointment\n

WHERE MONTH(date) = ?;