

**Dentist Booking**

**Software Design Document**

– HCM, July 2022 –

**Table of Contents**

[I. Overview 3](#_Toc108598413)

[1. Code Packages/Namespaces 3](#_Toc108598414)

[2. Database Schema 3](#_Toc108598415)

[II. Code Designs 4](#_Toc108598416)

[1. Booking appointment 4](#_Toc108598417)

[a. Class Diagram 4](#_Toc108598418)

[b. Class Specifications 4](#_Toc108598419)

[c. Sequence Diagram(s) 7](#_Toc108598420)

[d. Database queries 7](#_Toc108598421)

[2. Create Dentist 8](#_Toc108598422)

[a. Class Diagram 8](#_Toc108598423)

[b. Class Specifications 8](#_Toc108598424)

[c. Sequence Diagram(s) 9](#_Toc108598425)

[d. Database queries 9](#_Toc108598426)

[3. Create Invoice 10](#_Toc108598427)

[a. Class Diagram 10](#_Toc108598428)

[b.Sequence Diagram(s) 10](#_Toc108598429)

[c. Database queries 11](#_Toc108598430)

[4. Create Feedback 12](#_Toc108598431)

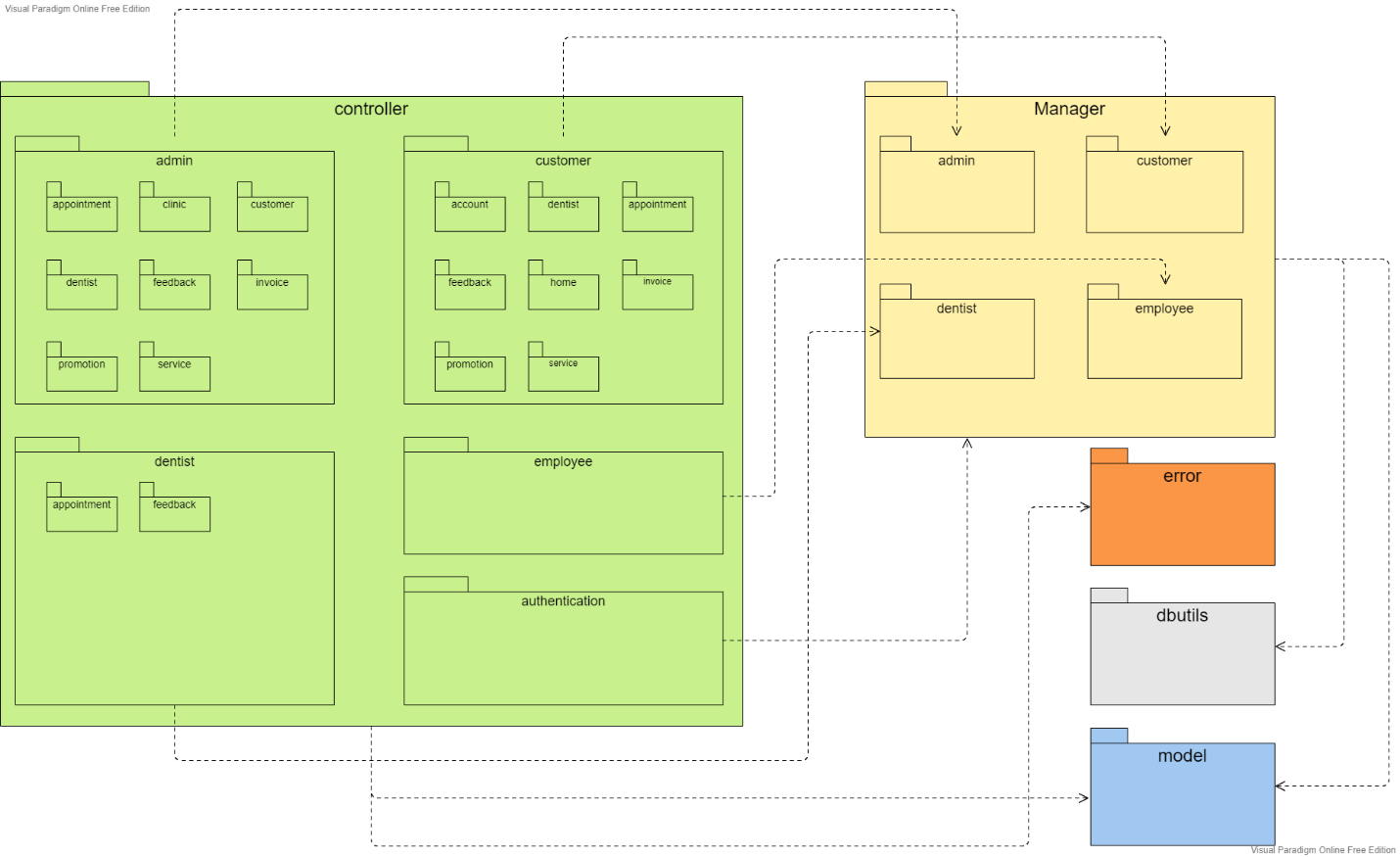
[a. Class Diagram 12](#_Toc108598432)

[b. Sequence Diagram(s) 12](#_Toc108598433)

[c. Database queries 12](#_Toc108598434)

# I. Overview

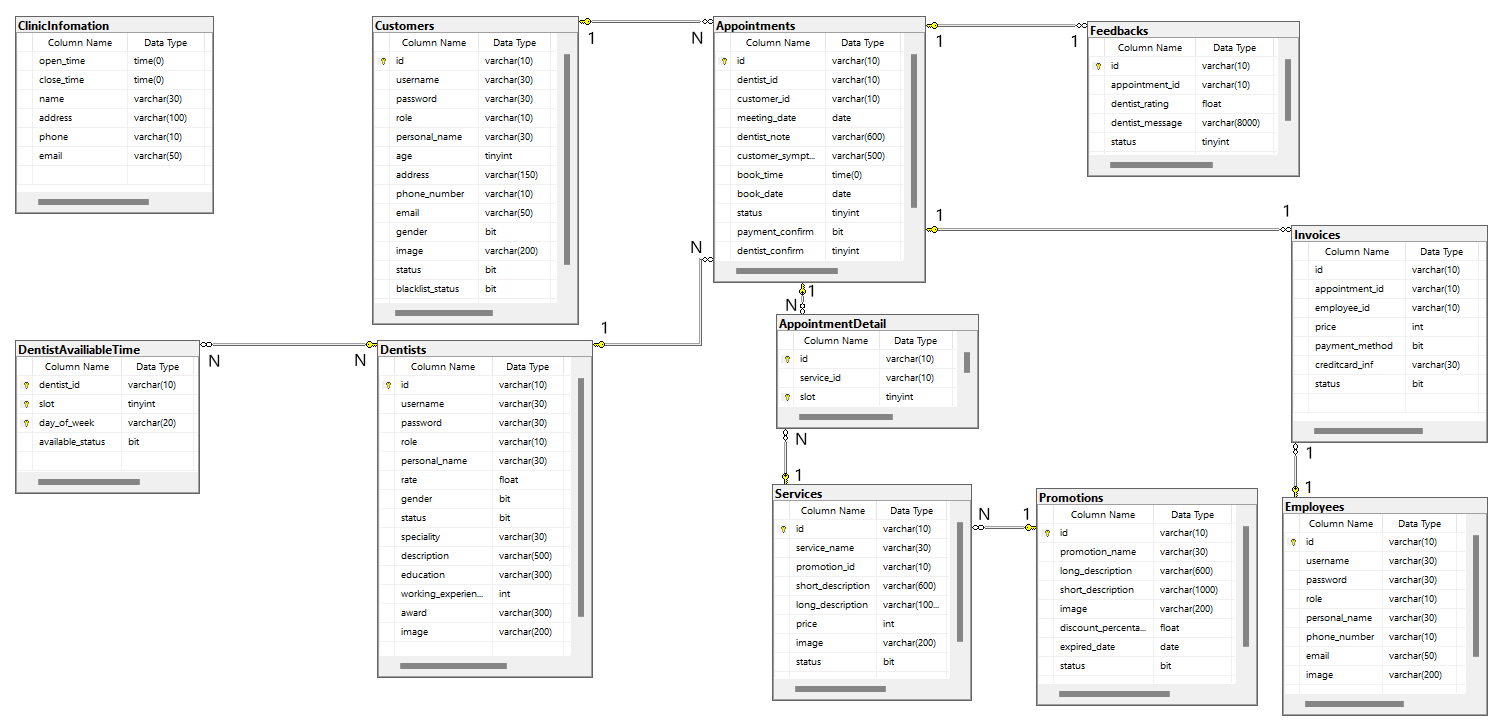
## 1. Code Packages/Namespaces



***Package descriptions & package class naming conventions***

|  |  |  |  |
| --- | --- | --- | --- |
| **Controller** | | | |
| 1 | **admin** | appointment | - Contain: **+** AdminDeleteAppointmentController for deleting a appointment where appointment\_id = ? **+** AdminSearchAppointmentController for showing list, searching appointments **-** Uppercase the first letter of any words, no space, "role + action + object + controller" |
| 2 | clinic | - Contain:  + AdminLoadClinicInformationController for showing clinic's bacsic information + AdminStatisticController for showing statistic  + AdminUpdateClinicInformationController for updating clinic's basic information - Uppercase the first letter of any words, no space, "role + action + object + controller" |
| 3 | customer | - Contain:  + AdminDeleteCustomerController for delete a customer where customer\_id = ? + AdminRestoreController for restoring a customer after deleting + AdminSearchCustomerController for showing list, searching customers + AdminSetBlaclistController for setting blacklist status of a customer to true  + AdminUnSetBlaclistController for setting blacklist status of a customer to false - Uppercase the first letter of any words, no space, "role + action + object + controller" |
| 4 | dentist | - Contain:  + AdminCreateDentistController for creating a new dentist to in database + AdminDeleteDentistController for delete a dentist where dentist\_id = ? + AdminRestoreDentistController for restoring a dentist after deleting + AdminSearchDentistController for showing list, searching dentists + AdminUpdateDentistController for updating changes on a dentist's information - Uppercase the first letter of any words, no space, "role + action + object + controller" |
| 5 | feedback | - Contain:  + AdminDeleteFeedbackController for delete a feedback where feedback\_id = ? + AdminRestoreController for restoring a feedback after deleting + AdminSearchDentistController for showing list, searching feedbacks - Uppercase the first letter of any words, no space, "role + action + object + controller" |
| 6 | invoice |  |
| 7 | promotion | - Contain:  + AdminCreatePromotionController for creating a new promotion to in database + AdminDeletePromotionController for delete a promotion where Promotion\_id = ? + AdminRestorePromotionController for restoring a promotion after deleting + AdminSearchPromotionController for showing list, searching promotions + AdminUpdatePromotionController for updating changes on a promotion's information - Uppercase the first letter of any words, no space, "role + action + object + controller" |
| 8 | service | - Contain:  + AdminCreateServiceController for creating a new service to in database + AdminDeleteServiceController for delete a service where service\_id = ? + AdminRemovePromotionController for removing an out of date promotion away from a service + AdminRestoreServiceController for restoring a serviceafter deleting + AdminSearchServiceController for showing list, searching services + AdminUpdateServiceController for updating changes on a service's information - Uppercase the first letter of any words, no space, "role + action + object + controller" |
| 9 | **customer** | account | - Contain:  + ChangePasswordController for updating a new password to the customer's account + ChangeProfileController for updating new information to the customer's profile - Uppercase the first letter of any words, no space, "action + object + controller" |
| 10 | appointment | - Contain:  + CheckoutAppointmentController for checking out the completed appointment + BookingAppointmentController for booking a new appointment + ViewAppointmentController for showing list, viewing booked appointments + CheckPaymentController for viewing, confirming payment - Uppercase the first letter of any words, no space, "action + object + controller" |
| 11 | feedback | - Contain:  + CreateFeedbackController for creating a feedback of a completed appointment - Uppercase the first letter of any words, no space, "action + object + controller" |
| 12 | dentist | - Contain:  + ShowDentistController for showing a list of dentist, introduction information + availble time of each dentist, searching dentists - Uppercase the first letter of any words, no space, "action + object + controller" |
| 13 | home | - Contain:  + LoadHomeController for loading information of the customer's index page - Uppercase the first letter of any words, no space, "action + object + controller" |
| 14 | invoice | - Contain:  + ViewInvoiceController for showing list of invoices + ViewDetailInvoiceController for showing detail information of each invoice  - Uppercase the first letter of any words, no space, "action + object + controller" |
| 15 | promotion | - Contain:  + ShowPromotionController for showing a list of promotions, information of each promotion, searching and sorting promotions - Uppercase the first letter of any words, no space, "action + object + controller" |
| 16 | service | - Contain:  + ShowServiceController for showing a list of services, information of each service, searching and sorting services - Uppercase the first letter of any words, no space, "action + object + controller" |
| 17 | **employee** | employee | - Contain:  + EmployeeAppointmentController for showing list of appointment + EmployeeCreateInvoiceController for creating invoice after customer checkout service + EmployeeDeleteAppointmentController for deleting appointment + EmployeeFeedbackStatusController for updating the status of the feedback after customer send feedback +EmployeeInvoiceDetailController for showing detail of appointment in appointment detail +EmployeeListFeedbackController for showing list of feedback customer send +EmployeeSetFinishStatusController updating the status of appointment to finish when they checkout +EmployeeShowAppointmentDashboardController for ... +EmployeeUpdateAppointmentStatusController for... - Uppercase the first letter of any words, no space, "role+ action + object + controller" |
| 18 | **dentist** | dentist | - Contain:  + ChangePasswordController for updating a new password + ScheduleController for showing list, create and delete of schedule + UpdateProfileController for updating new information to the dentist profile - Uppercase the first letter of any words, no space, "action + object + controller" |
| 19 | appointment | - Contain: + ConfirmAppointmentController for showing list appointment go with customer + ConfirmDentistAppointmentController for confirming appointment after customer book + ShowListDashboardController showing list of appointment from customer + CreateDentistNoteController for create a note for customer after they finish appointment - Uppercase the first letter of any words, no space, "action + object + controller" |
| 20 | feedback | - Contain: + FeedbackController for showing feedback of customer after they finish appointment - Uppercase the first letter of any words, no space, "object + controller" |
| **Model** | | | |
| 21 | **model** | | - Contain DTO model of each entity: + Appointment + AppointmentDetail + ClinicInformation + Customer + Dentist + DentistAvailableTime + Employee + Feedback + Invoice + Promotion + Serivce + UserGoogleDTO + Mail - Uppercase the first letter of any words, entity's name |
| **DBUtils** | | | |
| 22 | **dbutils** | | - Contain DBUtils for database connection |
| **Error** | | | |
| 23 | **error** | | - Contain error DTO model for each entity:  + AppointmentError + AppointmentDetailError + ClinicInformationError + CustomerError + DentistAvailableTimeError + DentistError + EmployeeError + FeedbackError + InvoiceError + PromotionError + SeriveError - Uppercase the first letter of any words, "entity's name + error" |

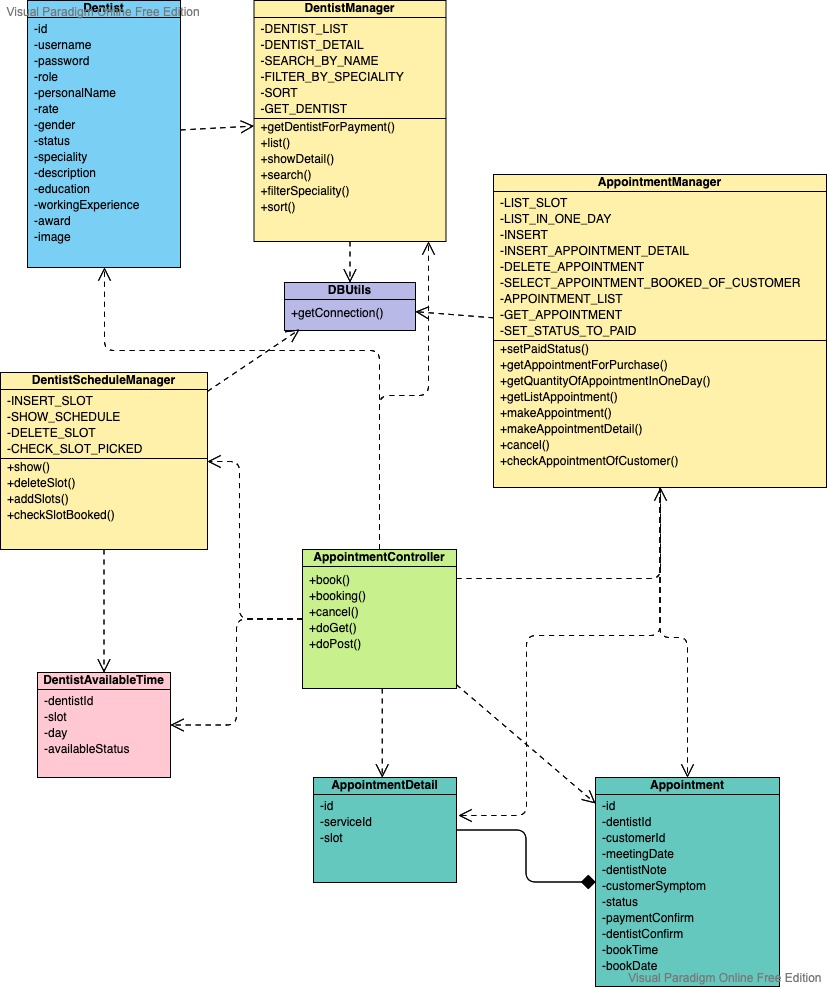
## 2. Database Schema



# II. Code Designs

## 1. Booking appointment

### a. Class Diagram



### b. Class Specifications

#### AppointmentController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *booking()* | * Inputs: HttpServletRequest, HttpServletResponse * Internal processing:  1. check if customer have already had an appointment that hasn’t been finished yet then load all information of services, dentists, available time of dentists from database through class DentistManager, AppointmentManager, DentistScheduleManager, ServiceManager 2. forwarding to book-appointment.jsp page |
| *02* | *book()* | * Inputs: HttpServletRequest, HttpServletResponse * Internal processing:  1. process the query string 2. take current time, date for saving booked time and date 3. initialize appointment id in format of APddMMYYYYQUANTITY 4. insert appointment’s information to database 5. send mail to customer about appointment's information |
| *03* | *cancel()* | * Inputs: HttpServletRequest, HttpServletResponse * Internal processing:  1. process the query string 2. *take current date and appointment's bookDate* 3. *take current time and appointment's bookTime* 4. *check if time of appointment is over 2 hours after bookTime or not* 5. *Redirect to customer-dashboard.jsp page*  * send mail to customer about appointment's information |
| *04* | *processRequest ()* | * Inputs: HttpServletRequest, HttpServletResponse * Internal processing:  1. check if there’s any session or not 2. get path info of the URL (servletURL/….? queryString) 3. process the request and response to method booking() or book() or cancel() |
| *05* | *doGet()* | * Inputs: HttpServletRequest, HttpServletResponse * Internal processing: call methods processRequest() when request call method GET |
| *06* | *doPost()* | * Inputs: HttpServletRequest, HttpServletResponse * Internal processing: call methods processRequest() when request call method POST |

#### AppointmentManager

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *checkAppointmentOfCustomer()* | * Inputs: customerId * Internal processing:  1. *Call method getConnection () of DBUtils class for connecting to database* 2. *Execute SELECT sql query string using method executeQuery () of class PrepareStatement* 3. *Get the result*  * Output: true if there’s an appointment of customer that hasn’t been finished yet, else return false |
| *02* | *makeAppointment ()* | * Inputs: Appointment, AppointmentDetail[] * Internal processing:  1. *Call method getConnection () of DBUtils class for connecting to database* 2. *Execute INSERT sql query string using method executeUpdate () of class PrepareStatement* 3. *Execute method makeAppointmentDetail () for multiple slots added*  * Output: true if insert appointment’s information to database successfully, else return false |
| *03* | *makeAppointmentDetail ()* | * Inputs: AppointmentDetail * Internal processing:  1. *Call method getConnection () of DBUtils class for connecting to database* 2. *Execute INSERT sql query string using method executeUpdate () of class PrepareStatement*  * Output: true if insert appointment’s detail information to database successfully, else return false |
| *04* | *getQuantityOfAppointmentInOneDay ()* | * Internal processing:  1. *Call method getConnection () of DBUtils class for connecting to database* 2. *Execute SELECT sql query string using method executeQuery () of class PrepareStatement*  * Output: (integer) quantity of appointment in one day |
| *05* | *cancel ()* | * Inputs: appointmentId * Internal processing:  1. *Call method getConnection () of DBUtils class for connecting to database* 2. *Execute DELETE sql query string using method executeUpdate () of class PrepareStatement*  * Output: true if delete appointment successfully, else return false |

#### DentistManager

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *list ()* | * Internal processing:  1. *Call method getConnection () of DBUtils class for connecting to database* 2. *Execute SELECT sql query string using method executeQuery() of class PrepareStatement* 3. *Get the result*  * Output: list of dentists |

***DentistScheduleManager***

***Class Methods***

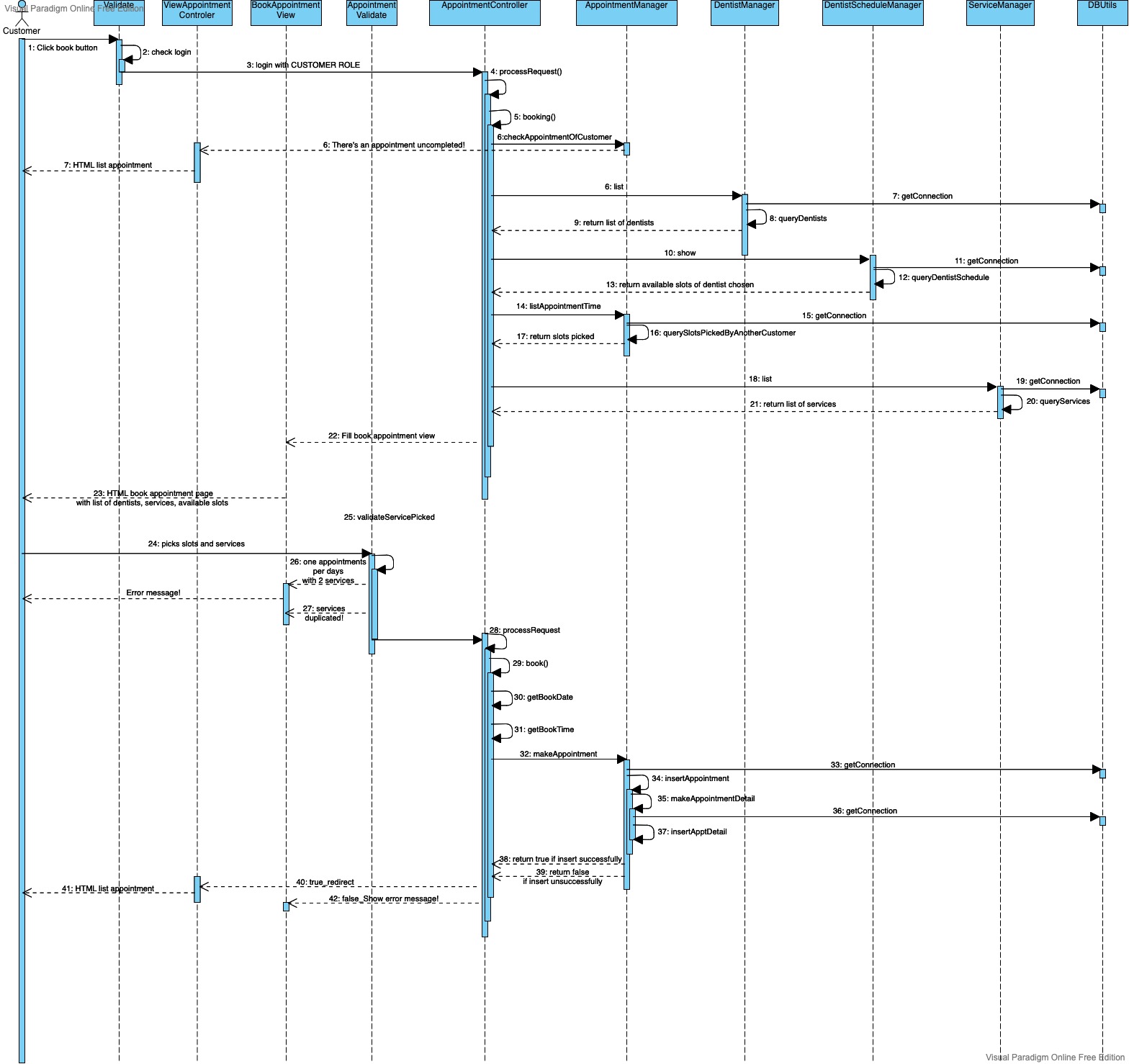
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *show ()* | * *Inputs:* *dentistId, day(day of week)* * Internal processing:  1. *Call method getConnection () of DBUtils class for connecting to database* 2. *Execute SELECT sql query string using method executeQuery() of class PrepareStatement* 3. *Get the result*  * Output: list all dentist’s available slots on a day |

***DBUtils***

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection ()* | * *Make connection to database* |

### c. Sequence Diagram(s)



### d. Database queries

* INSERT\_APPOINTMENT = "INSERT INTO Appointments ( [id], [dentist\_id], [customer\_id], [meeting\_date], [dentist\_note], [customer\_symptom], [book\_time], [book\_date], [status], [payment\_confirm], [dentist\_confirm] VALUES (?,?,?,?,?,?,?,?,?,?,?);”
* INSERT\_APPOINTMENT\_DETAIL = "INSERT INTO AppointmentDetail VALUES (?,?,?);"
* SELECT\_APPOINTMENT\_BOOKED\_OF\_CUSTOMER = "SELECT \* FROM Appointments,(SELECT CAST( GETDATE() AS Date ) as now) as CurrentDate

WHERE Appointments.meeting\_date >= CurrentDate.[now]

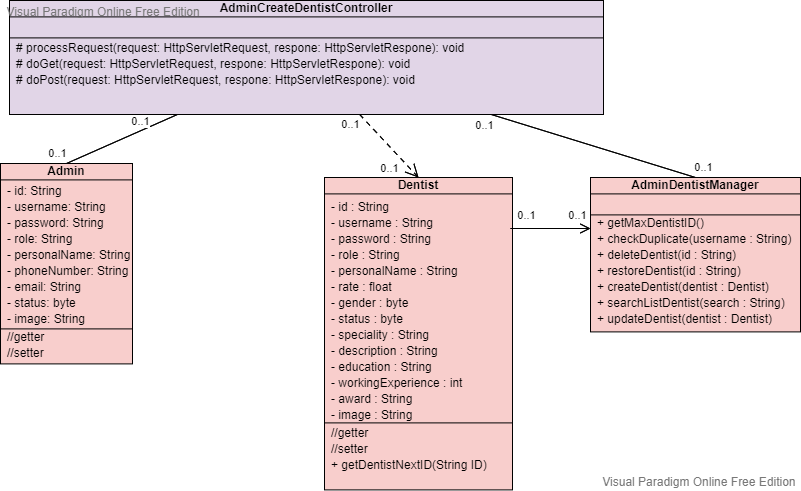
AND customer\_id = ?

AND [status] = 1 AND payment\_confirm = 0 AND dentist\_confirm = 0 ;"

* DELETE\_APPOINTMENT = "DELETE FROM Appointments WHERE Appointments.id= ?;"
* LIST\_IN\_ONE\_DAY = " SELECT \* FROM Appointments WHERE meeting\_date = ? ;"
* DENTIST\_LIST = "SELECT \* FROM Dentists WHERE status = 1;"
* SHOW\_SCHEDULE = "SELECT \* FROM DentistAvailiableTime WHERE dentist\_id = ? AND day\_of\_week = ? "

## 2. Create Dentist

### a. Class Diagram

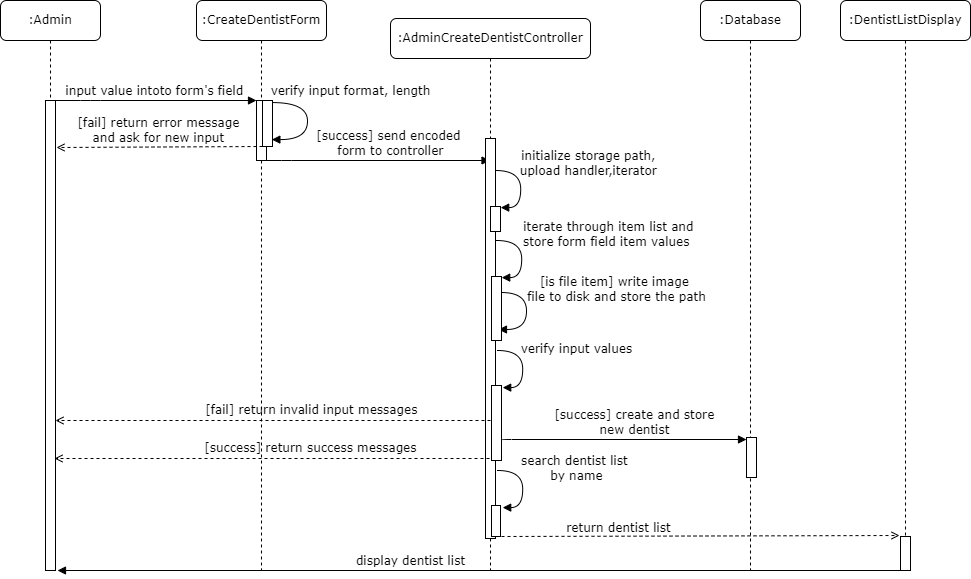


### b. Class Specifications

***AdminCreateDentistController***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest ()* | * Inputs: HttpServletRequest, HttpServletResponse * Internal processing:  1. Retreive encoded request form 2. Initialize storage path, upload file handler, iterator 3. Iterate through item list, store item value, and write image file to the disk if item is not form field 4. Verify input value. If valid send success message else send error message include reason. |
| *05* | *doGet()* | * Inputs: HttpServletRequest, HttpServletResponse * Internal processing: call methods processRequest() when request call method GET |
| *06* | *doPost()* | * Inputs: HttpServletRequest, HttpServletResponse * Internal processing: call methods processRequest() when request call method POST |

### c. Sequence Diagram(s)

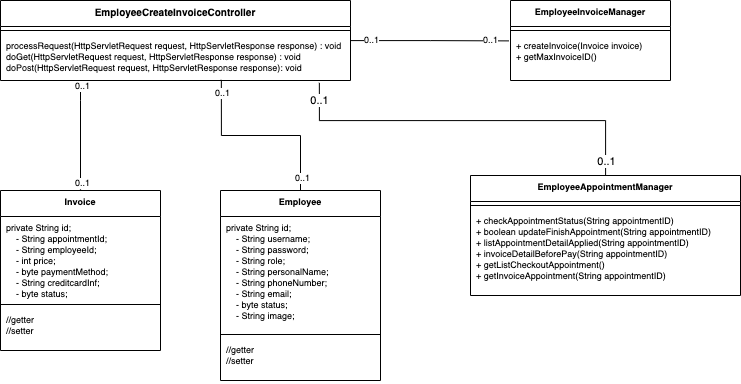


### d. Database queries

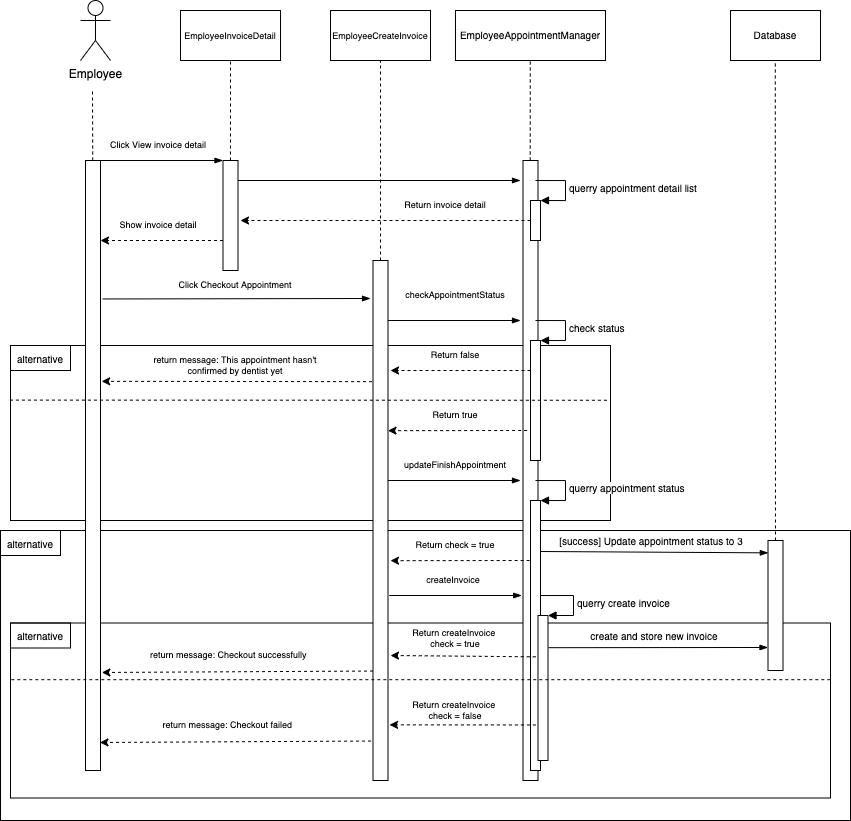
* CREATE = "INSERT INTO Dentists (id, username, password, role, personal\_name, rate, gender, status, speciality, description, education, working\_experience, award, image) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?) ";
* SELECT\_MAX\_DENTIST\_ID= "SELECT MAX(id) AS maxDentistID FROM Dentists WHERE LEN(id) = (SELECT MAX(LEN(id)) FROM Dentists) ";
* SEARCH = "SELECT \* FROM Dentists WHERE personal\_name LIKE ? ";

## 3. Create Invoice

### a. Class Diagram



### b.Sequence Diagram(s)



### c. Database queries

## 4. Create Feedback

### a. Class Diagram

### b. Sequence Diagram(s)

### c. Database queries

LIST\_FEEDBACK = "SELECT \* FROM Feedbacks";

CREATE = "INSERT INTO Feedbacks (id, appointment\_id, dentist\_rating, dentist\_message, status) VALUES (?, ?, ?, ?, ?)";

SELECT\_MAX\_FEEDBACK\_ID = "SELECT MAX(id) AS maxFeedbackId FROM Feedbacks WHERE LEN(id) = (SELECT MAX(LEN(id)) FROM Feedbacks)";

UPDATE\_AVG\_RATE = "UPDATE Dentists SET rate = ? WHERE id = ?";

AVG\_RATE = "SELECT AVG(dentist\_rating) AS averageRate FROM Feedbacks\n"

+ "INNER JOIN Appointments ON Appointments.id = appointment\_id\n"

+ "WHERE Appointments.dentist\_id = ?";

GET\_DENTIST\_ID = "SELECT dentist\_id FROM Feedbacks\n"

+ "INNER JOIN Appointments ON Appointments.id = appointment\_id\n"

+ "WHERE Feedbacks.appointment\_id = ?";