

**Dentist Booking**

**Software Design Document**

– HCM, July 2022 –

**Table of Contents**

[I. Overview 3](#_Toc69802609)

[1. Code Packages 3](#_Toc69802610)

[2. Database Schema 3](#_Toc69802611)

[II. Code Designs 4](#_Toc69802612)

[1. Booking Appointment 4](#_Toc69802613)

[a. Class Diagram 4](#_Toc69802614)

[b. Class Specifications 4](#_Toc69802615)

[c. Sequence Diagram(s) 4](#_Toc69802616)

[d. Database queries 5](#_Toc69802617)

[2. <Feature/Function Name2> 5](#_Toc69802618)

[III. Database Tables 5](#_Toc69802619)

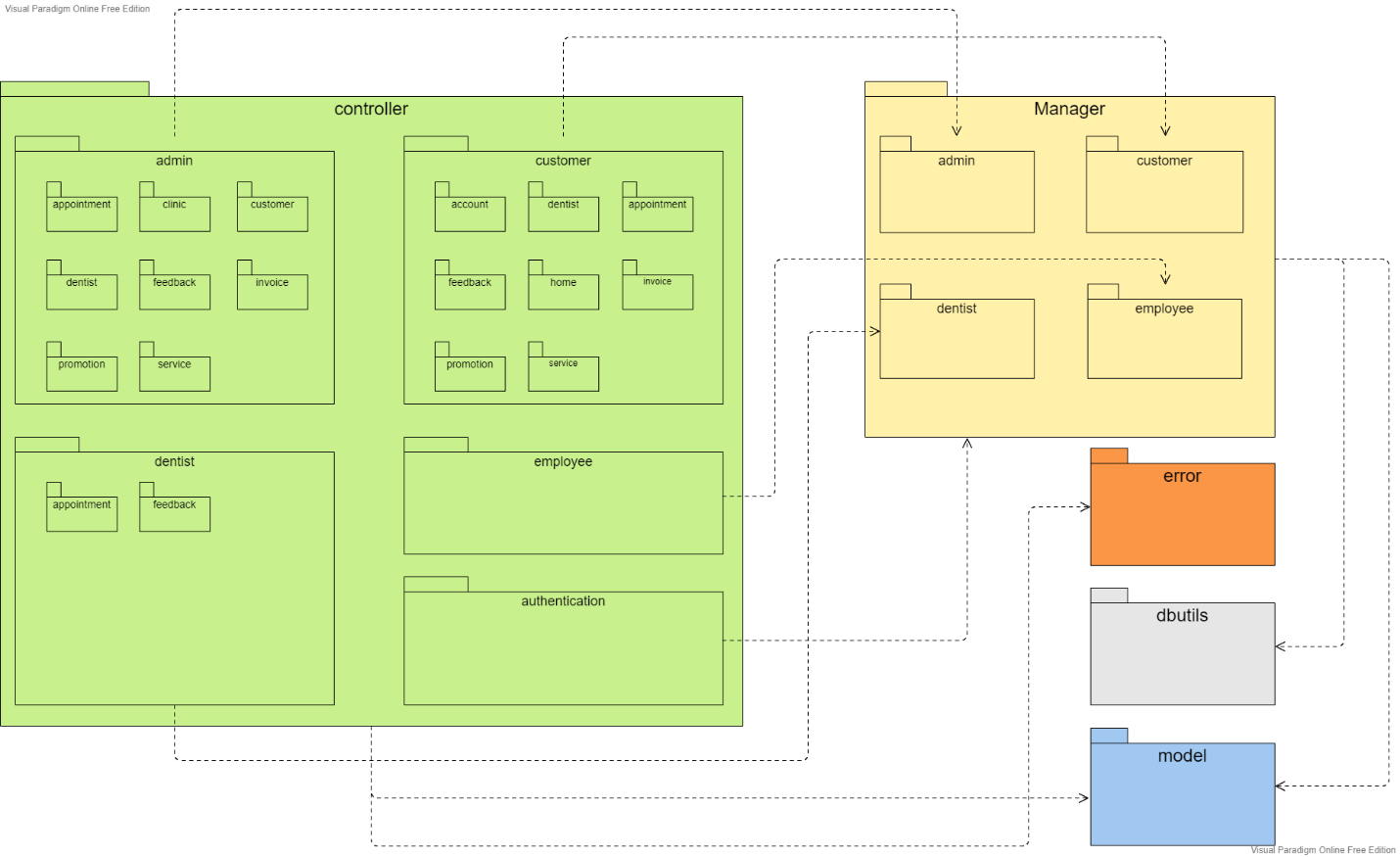
[1. <Table name 1> 5](#_Toc69802620)

[2. <Table name 2…> 5](#_Toc69802621)

# I. Overview Hiếu & Vinh

## 1. Code Packages/Namespaces

*[Provide the package diagram for each sub-system. The content of this section including the overall package diagram, the explanation, package and class naming conventions in each package. Please see the sample and description table format below – following Java project naming convention]*

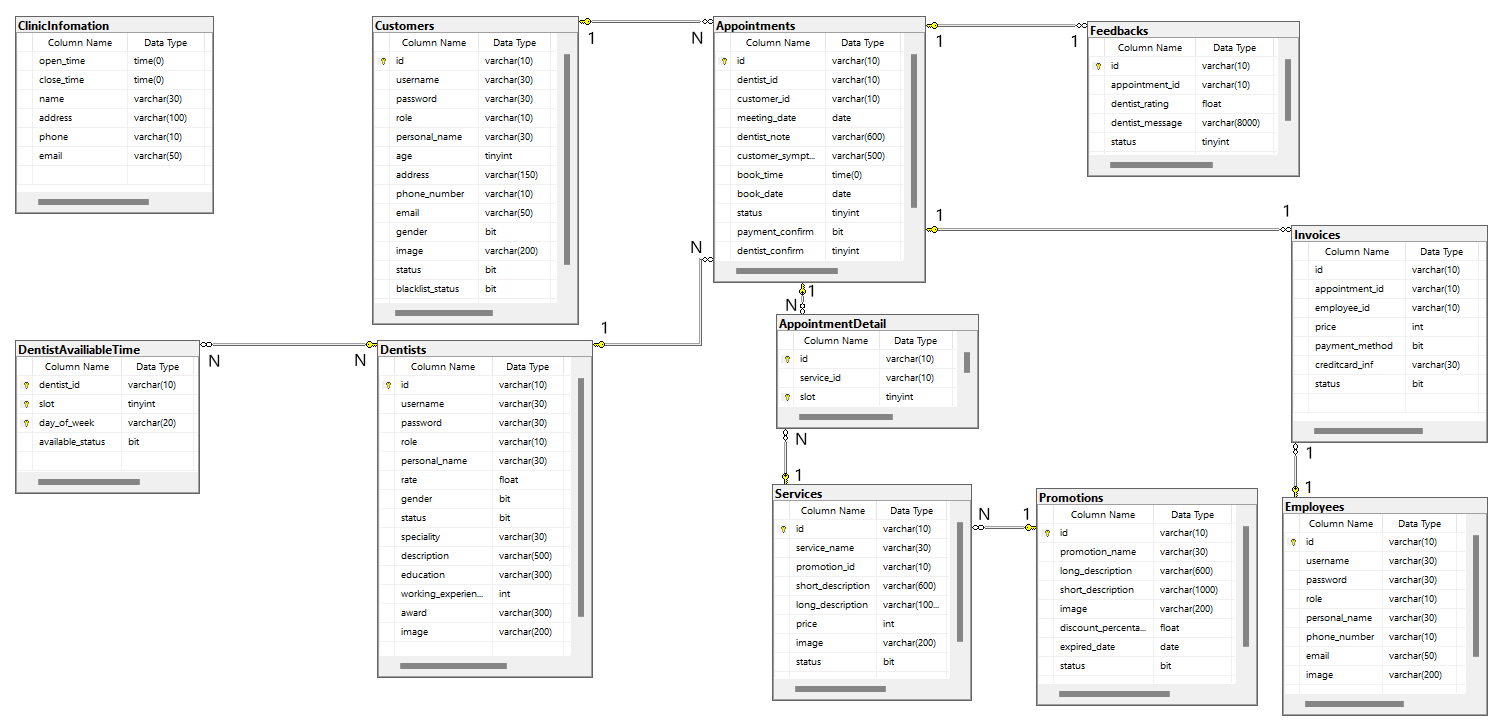


***Package descriptions & package class naming conventions An&Đăng***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| *01* | *controller* | *<Description of the package>*  *<Class naming convention>* |
| *02* |  |  |
|  |  |  |

## 2. Database Schema

*[Provide the tables relationship like example below – following MySQL database naming convention]*



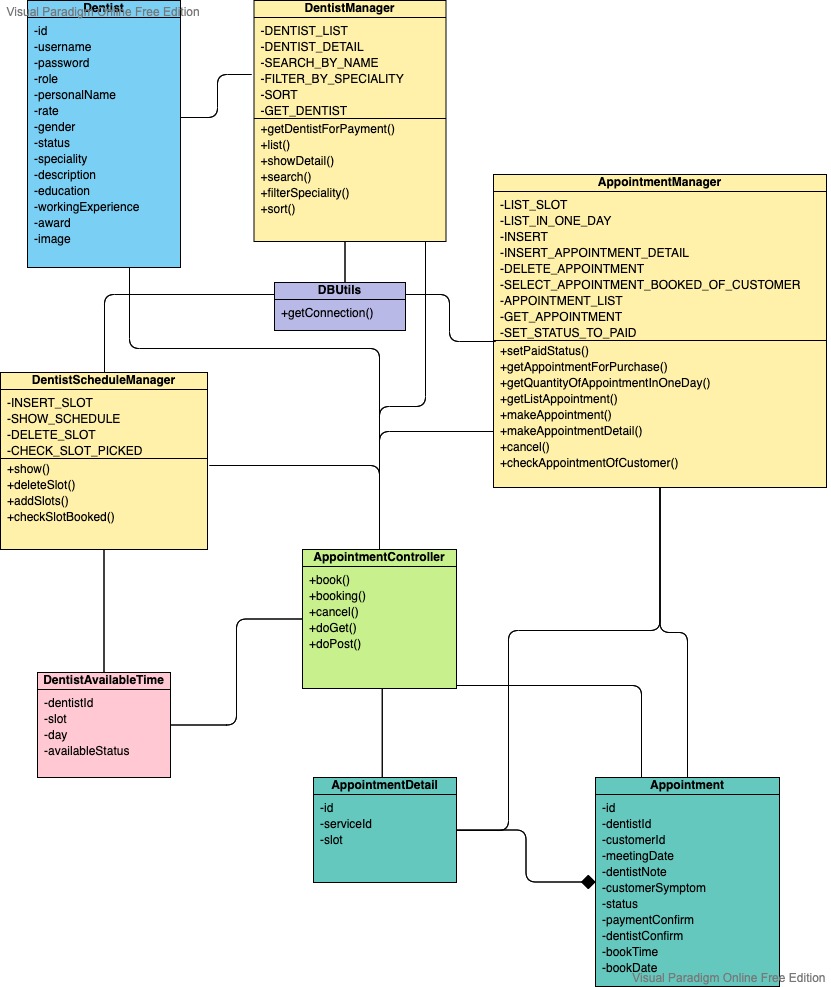
***Table descriptions & package class naming conventions are as below***

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| *01* | *<Table name>* | *<Description of the table>*  *- Primary keys: <<list of primary key fields>>*  *- Foreign keys: <<list of foreign key fields>>* |
| *02* |  |  |

# 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 |

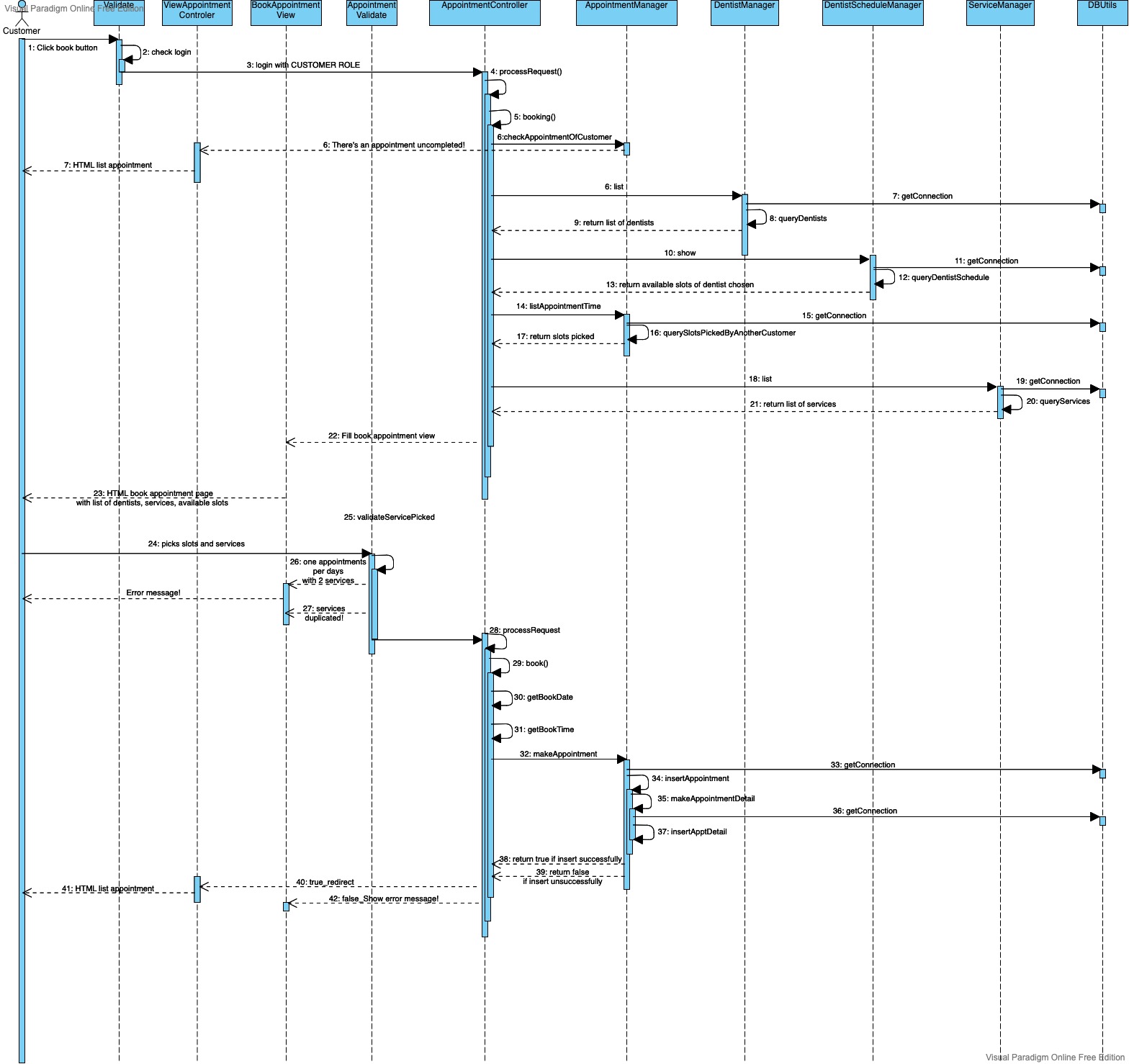
***DBUtils***

***Class Methods***

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

### c. Sequence Diagram(s)

*[Provide the sequence diagram(s) for the feature, see the sample below]*



### 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]

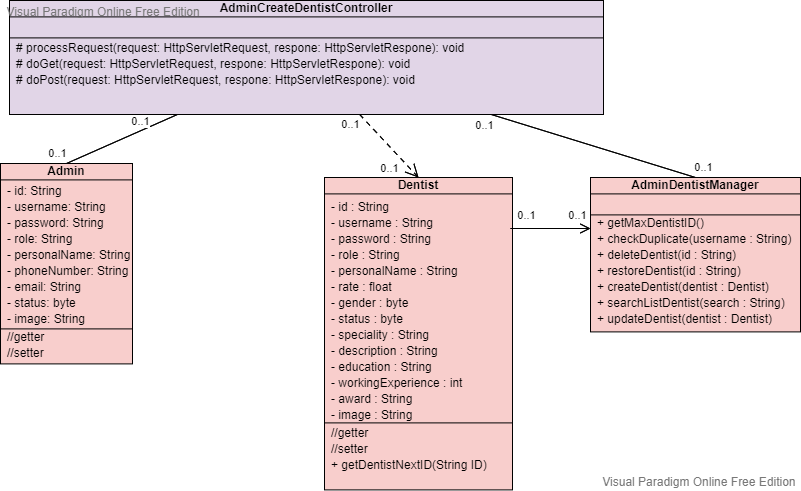
AND customer\_id = ?

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

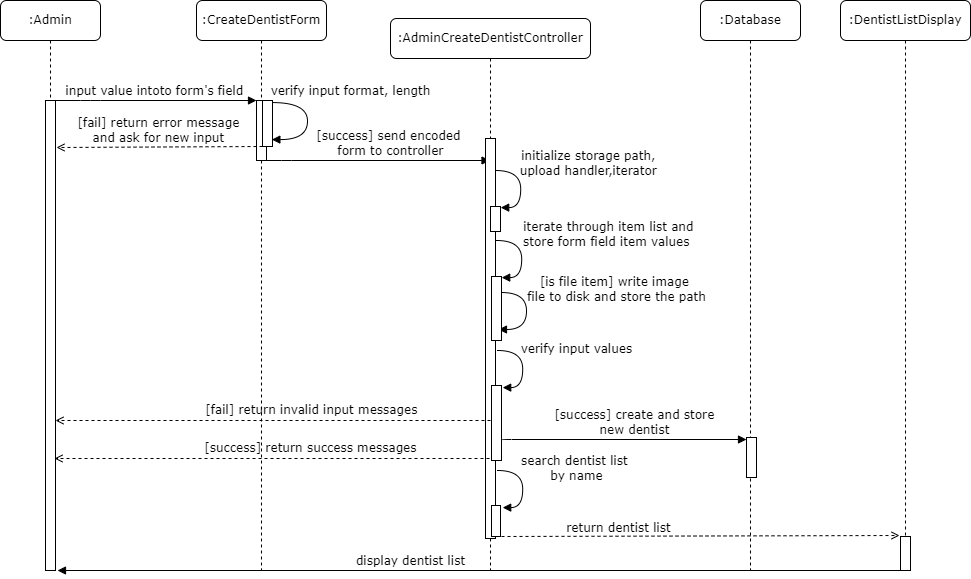


### b. Class Specifications

***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)

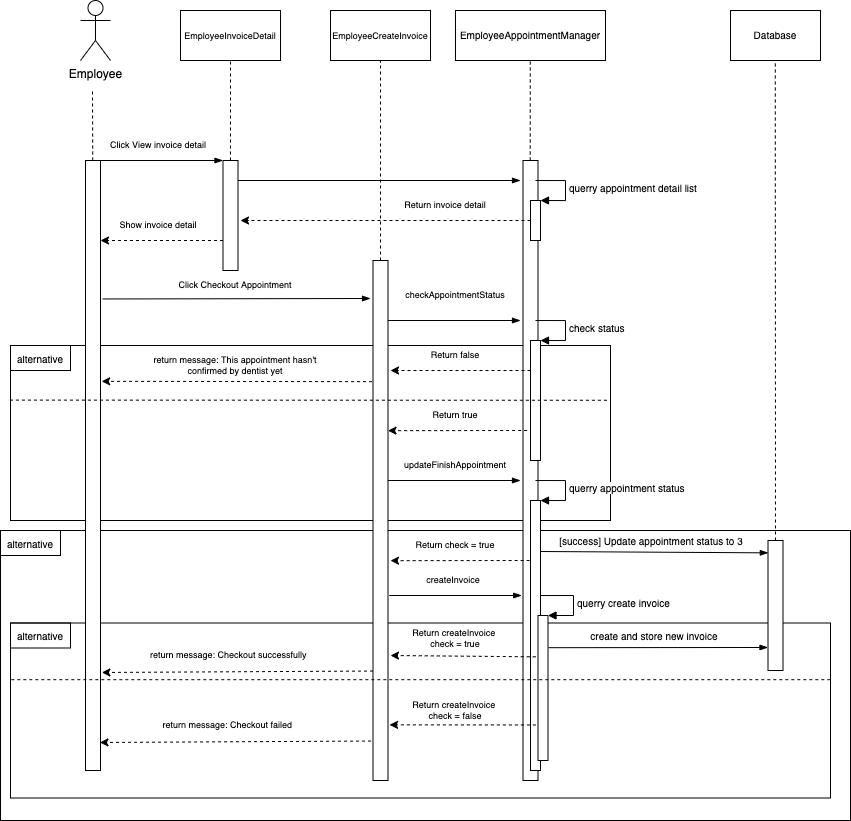


### d. Database queries

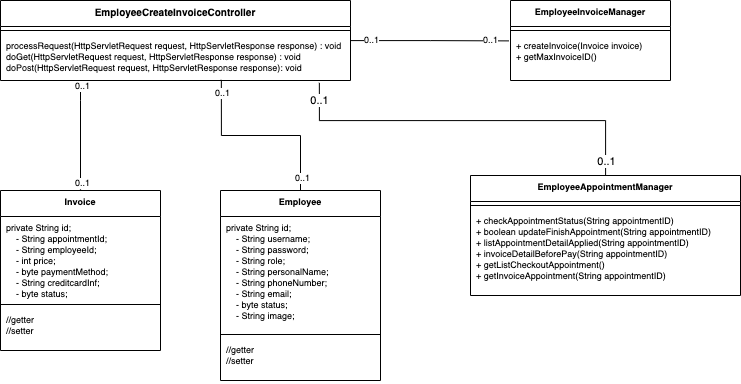
* CREATE = "INSERT INTO Dentists (id, username, password, role, personal\_name, rate, gender, status, speciality, description, education, working\_experience, award, image) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
* SELECT\_MAX\_DENTIST\_ID= "SELECT MAX(id) AS maxDentistID FROM Dentists WHERE LEN(id) =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

## 2. Create Invoice

### Sequence Diagram(s)



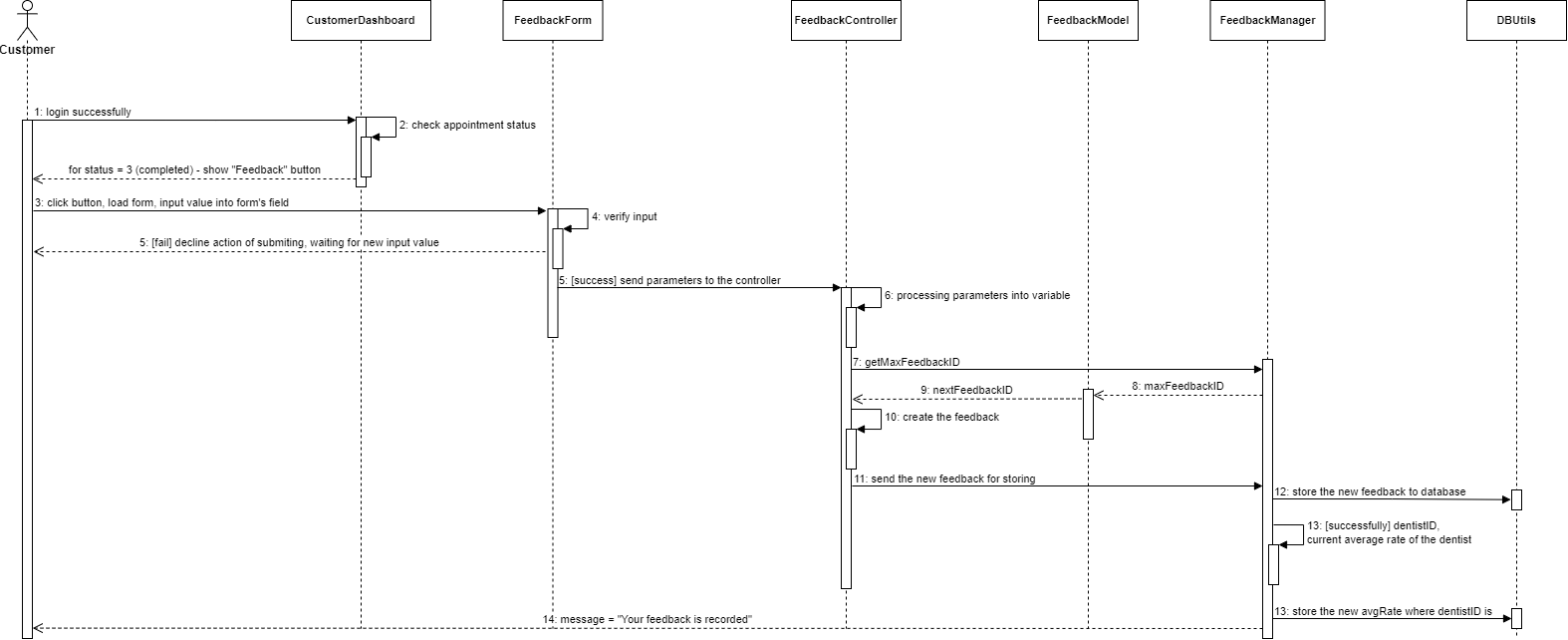
### b. Class Specifications



***4. Create Feedback***

a. Class diagram

b. Class Specification

c. Sequence Diagram