

**ONLINE TUTOR**

**Software Design Document**

– Ho Chi Minh, November 2021 –

**Table of Contents**

[I. Overview](#_heading=h.gjdgxs) **1**0

[1. Code Packages/Namespaces](#_heading=h.30j0zll) 10

[a.](#_heading=h.8y8n18vmrh5l) Package diagram 10

[b.](#_heading=h.zazuy6ue4zj0) Package descriptions 10

[2. Database Schema](#_heading=h.1fob9te) 12

[a. Entities relationship diagram](#_heading=h.8y8n18vmrh5l) 12

[b. Table Description](#_heading=h.zazuy6ue4zj0) 12

[II. Code Designs](#_heading=h.3znysh7) **13**

[1. Authenticate account](#_heading=h.2et92p0) 13

[1.1. Register new account](#_heading=h.mx6kytsdf1w6) 13

[a. Class Diagram](#_heading=h.v3rhvy1jxhki) 13

[b. Class Specifications](#_heading=h.tyjcwt) 13

[c. Sequence Diagram(s)](#_heading=h.qjjky8ai4nih) 14

[d. Database queries](#_heading=h.1t3h5sf) 14

[1.2. Login](#_heading=h.ecv3u4asdg82) 14

[a. Class Diagram](#_heading=h.1xxkxakeq4e6) 15

[b. Class Specifications](#_heading=h.2hkiu7jg5dfy) 15

[c. Sequence Diagram(s)](#_heading=h.uadl8ql9e71j) 16

[d. Database queries](#_heading=h.kj9yd06c5lbq) 17

[1.3. Log out](#_heading=h.daoms3romfge) 17

[a. Class Diagram](#_heading=h.w0cjjtsnnf5e) 17

[b. Class Specifications](#_heading=h.tsed4l4l66bs) 17

[c. Sequence Diagram(s)](#_heading=h.h8c7cosalnd6) 18

[d. Database queries](#_heading=h.l073eobx7tpu) 18

[2.](#_heading=h.4d34og8) User Management 19

[2.1. Change user’s role](#_heading=h.1ynkfkj6bpty) 19

[a. Class Diagram](#_heading=h.ebcl37nbnp7g) 19

[b. Class Specifications](#_heading=h.ge48t5rbj94z) 19

[c. Sequence Diagram(s)](#_heading=h.zf8lozux1pr1) 20

[d. Database queries](#_heading=h.r18n2b94gwuc) 20

[2.2. Delete user](#_heading=h.85qm8o2umtmh) 21

[a. Class Diagram](#_heading=h.t59tr0bfbac8) 21

[b. Class Specifications](#_heading=h.repr39xeuh2j) 21

[c. Sequence Diagram(s)](#_heading=h.tfi530i2hp4w) 22

[d. Database queries](#_heading=h.6rbqyuut4k4g) 22

[2.3. View user](#_heading=h.vd3k958ubpvj) 23

[a. Class Diagram](#_heading=h.oouxxoryabt8) 23

[b. Class Specifications](#_heading=h.t210ndu2fiyz) 23

[c. Sequence Diagram(s)](#_heading=h.sofcjp8abzc2) 23

[d. Database queries](#_heading=h.lnyudlenslad) 24

[2.4. Search tutor](#_heading=h.if3vvutq5yru) 25

[a. Class Diagram](#_heading=h.ed11ju21l0s0) 25

[b. Class Specifications](#_heading=h.851zr35mj94l) 25

[c. Sequence Diagram(s)](#_heading=h.g0ine2pyyggz) 26

[d. Database queries](#_heading=h.p6p13woqc92m) 26

[3.](#_heading=h.2lwamvv) Course Management 27

[3.1. View course](#_heading=h.111kx3o) 27

[3.1.1 View all course (admin](#_heading=h.93efnq5iai81)) 27

[a. Class Diagram](#_heading=h.7mbftdxm2f3q) 27

[b. Class Specifications](#_heading=h.bs3uac5xlg2v) 27

[c. Sequence Diagram(s)](#_heading=h.x0imq5px4tm5) 28

[d. Database queries](#_heading=h.whwym1mijvez) 28

[3.1.2 View details one course (admin](#_heading=h.c71bncf7hdti)) 29

[a. Class Diagram](#_heading=h.asdn5mk3rv0v) 29

[b. Class Specifications](#_heading=h.u4bj3apupr5) 29

[c. Sequence Diagram(s)](#_heading=h.kmu14csta1uz) 30

[d. Database queries](#_heading=h.w6dph43nxkyq) 30

[3.1.3 View all course (public](#_heading=h.rsumcm1a2e23)) 31

[a. Class Diagram](#_heading=h.tk7xcgtbkjfe) 31

[b. Class Specifications](#_heading=h.u4zxp7p6uvai) 31

[c. Sequence Diagram(s)](#_heading=h.tn0fl2wsxp5p) 32

[d. Database queries](#_heading=h.qpaxz1baqcd3) 32

[3.1.2 View details one course (public](#_heading=h.noxb7wgf5ybk)) 33

[a. Class Diagram](#_heading=h.5rosc1ypndhj) 33

[b. Class Specifications](#_heading=h.xv80pv7l5rq2) 33

[c. Sequence Diagram(s)](#_heading=h.20z6i2iahpq7) 34

[d. Database queries](#_heading=h.bawazcufn27) 34

[3.2. Create course](#_heading=h.bx99mcrqif7k) 35

[a. Class Diagram](#_heading=h.2dtibg2y1hi3) 35

[b. Class Specifications](#_heading=h.rjf8zam9vmag) 35

[c. Sequence Diagram(s)](#_heading=h.cw2bjwplj16p) 36

[d. Database Queries](#_heading=h.3a61f1fa2j5i) 36

[3.3. Update course](#_heading=h.206ipza) 37

[a. Class Diagram](#_heading=h.m4x6gq8wesxc) 37

[b. Class Specifications](#_heading=h.rtxinac29vb9) 37

[c. Sequence Diagram(s)](#_heading=h.evushiz7jbib) 37

[d. Database queries](#_heading=h.7wlmpp9i7at4) 38

[3.4. Delete course](#_heading=h.4k668n3) 39

[a. Class Diagram](#_heading=h.g3g7xpub3kn8) 39

[b. Class Specifications](#_heading=h.ps8of8bugnc) 39

[c. Sequence Diagram(s)](#_heading=h.7azd6kvc8sij) 39

[d. Database queries](#_heading=h.sfzo7w46voke) 40

[3.6 Toggle course](#_heading=h.dnkgfxv7213h) 41

[a. Class Diagram](#_heading=h.xe6xetpk9gkm) 41

[b. Class Specifications](#_heading=h.snkxxn3ju840) 41

[c. Sequence Diagram(s)](#_heading=h.803mqems57gn) 41

[d. Database queries](#_heading=h.7v3yo511mq5h) 42

[4.](#_heading=h.2zbgiuw) Course Material Management 43

[4.1. Create course material](#_heading=h.1egqt2p) 43

[a. Class Diagram](#_heading=h.pb3i62ebc0qc) 43

[b. Class Specifications](#_heading=h.vgg7ga6w66a0) 43

[c. Sequence Diagram(s)](#_heading=h.6nj2o74hmvpu) 44

[d. Database queries](#_heading=h.ti7o8vkm41oc) 44

[4.2. View course material](#_heading=h.1aexf8ascwaz) 45

[a. Class Diagram](#_heading=h.p0300nus41c4) 45

[b. Class Specifications](#_heading=h.unyjwv7ia70h) 45

[c. Sequence Diagram(s)](#_heading=h.ajic27yca4v) 46

[d. Database queries](#_heading=h.dcpu4bo5ne7q) 46

[4.3. Update course material](#_heading=h.q5f02ly18bc3) 47

[a. Class Diagram](#_heading=h.9yvtuk2n76ay) 47

[b. Class Specifications](#_heading=h.wcdeshcc656n) 47

[c. Sequence Diagram(s)](#_heading=h.bli7lvwc4zva) 48

[d. Database queries](#_heading=h.gwby6re2f73f) 48

[4.4. Delete course material](#_heading=h.sqyw64) 49

[a. Class Diagram](#_heading=h.8178voviq4pg) 49

[b. Class Specifications](#_heading=h.chvmh08at7zk) 50

[c. Sequence Diagram(s)](#_heading=h.gr7n8vrtq0m1) 50

[d. Database Queries](#_heading=h.xb6ldlujowqf) 51

[5.](#_heading=h.3cqmetx) Course Timetable Management 52

[5.1. Create timetable](#_heading=h.cyj7gb393mj7) 52

[a. Class Diagram](#_heading=h.2an2vbifc9x8) 52

[b. Class Specifications](#_heading=h.xous9sgfh9f) 53

[c. Sequence Diagram(s)](#_heading=h.6toonnwa98y8) 53

[d. Database queries](#_heading=h.hak7a9163a98) 53

[5.2. View course timetable](#_heading=h.cupxnxtuk1vu) 54

[a. Class Diagram](#_heading=h.n7if5c3811o) 54

[b. Class Specifications](#_heading=h.y98cnp3xnc0o) 54

[c. Sequence Diagram(s)](#_heading=h.q5bmm2uifvbz) 55

[d. Database queries](#_heading=h.fpvrtbm08w7v) 55

[5.3. Update course timetable](#_heading=h.o9l333j8u2eg) 56

[a. Class Diagram](#_heading=h.ens1eh2fzxo) 56

[b. Class Specifications](#_heading=h.7ld4s1dh41va) 57

[c. Sequence Diagram(s)](#_heading=h.fj890no9hrzu) 57

[d. Database Queries](#_heading=h.6k1k71avv33p) 58

[5.4. Delete course timetable](#_heading=h.9pks0xdvs9lo) 59

[a. Class Diagram](#_heading=h.fk79o9bf2odl) 59

[b. Class Specifications](#_heading=h.6igee395h20g) 60

[c. Sequence Diagram(s)](#_heading=h.yg9hwpruaa4l) 60

[d. Database Queries](#_heading=h.d0vjaxsq46kx) 61

[6. Course Registration](#_heading=h.a07gfwaf4v8c) 62

[6.1.  Register course](#_heading=h.di0cr1vdyzm1) 62

[a. Class Diagram](#_heading=h.2h8j28s3och) 62

[b. Class Specifications](#_heading=h.v6wi9tfexjue) 62

[c. Sequence Diagram(s)](#_heading=h.syre0usbek7w) 63

[d. Database queries](#_heading=h.x59vlph4cwud) 63

[6.2.  Approve course registration](#_heading=h.avb9kxpfjnee) 64

[a. Class Diagram](#_heading=h.erk0tvnwdb7) 64

[b. Class Specifications](#_heading=h.bqkuimwkt89i) 64

[c. Sequence Diagram(s)](#_heading=h.59cxzrwqhgmc) 65

[d. Database queries](#_heading=h.u69rv7qho3sq) 65

[7. Rate](#_heading=h.1664s55) Management 65

[7.1.  Add tutor rating](#_heading=h.3q5sasy) 66

[a. Class Diagram](#_heading=h.ncb6i1wxnt3k) 66

[b. Class Specifications](#_heading=h.7lrlkn391sdn) 66

[c. Sequence Diagram(s)](#_heading=h.fbzxxrp0dtfj) 67

[d. Database queries](#_heading=h.gj6vjdgteeml) 67

[7.2.  Update tutor rating](#_heading=h.60kn9vi4iund) 68

[a. Class Diagram](#_heading=h.7l4vlb4zk55g) 68

[b. Class Specifications](#_heading=h.8e8tkdvl2u5i) 68

[c. Sequence Diagram(s)](#_heading=h.6ym92gp8cnjw) 69

[d. Database queries](#_heading=h.yab1f8eszjk1) 69

[7.3.  Delete tutor rating](#_heading=h.r9eidhb1hnn3) 70

[a. Class Diagram](#_heading=h.2nppnjgddyqt) 70

[b. Class Specifications](#_heading=h.6a2otygo5mys) 70

[c. Sequence Diagram(s)](#_heading=h.4wuf4tin3n3v) 70

[d. Database queries](#_heading=h.ccrbfovvyun4) 71

[7.4.  View tutor rating](#_heading=h.udnbls4zw1v) 72

[a. Class Diagram](#_heading=h.oj9bfi38btqq) 72

[b. Class Specifications](#_heading=h.6cvo15o22qux) 72

[c. Sequence Diagram(s)](#_heading=h.d81rw0xhssic) 73

[d. Database queries](#_heading=h.2zbit0ppdph6) 73

[8.  Profile management](#_heading=h.25b2l0r) 74

[8.1.  View profile](#_heading=h.kgcv8k) 74

[a. Class Diagram](#_heading=h.9ivo9srsylrs) 74

[b. Class Specifications](#_heading=h.rg1budwbh61t) 74

[c. Sequence Diagram(s)](#_heading=h.7po4rx3fuqth) 74

[d. Database queries](#_heading=h.jnjt41306i6w) 75

[8.2.  Edit profile](#_heading=h.34g0dwd) 76

[a. Class Diagram](#_heading=h.fudhlmgyuc1h) 76

[b. Class Specifications](#_heading=h.aqgqp74jnm7i) 76

[c. Sequence Diagram(s)](#_heading=h.izgkt58zrsnx) 76

[d. Database queries](#_heading=h.6tpaoegnkwlz) 77

[8.3.  Update password](#_heading=h.yavv2ej406bq) 78

[a. Class Diagram](#_heading=h.cqkxapqpxqx5) 78

[b. Class Specifications](#_heading=h.x0nu0lqt41xw) 78

[c. Sequence Diagram(s)](#_heading=h.gio1qlswphtn) 78

[d. Database queries](#_heading=h.tdltjf8g0g0j) 79

[9. Community](#_heading=h.1jlao46) 80

[9.1. Question management](#_heading=h.kxakx0ccuyrd) 80

[9.1.1.  Add question](#_heading=h.43ky6rz) 80

[a. Class Diagram](#_heading=h.2marp0blq8up) 80

[b. Class Specifications](#_heading=h.vv7msxylv36d) 80

[c. Sequence Diagram(s)](#_heading=h.u6mw3zsfsabl) 81

[d. Database queries](#_heading=h.mz8ztsf6235i) 81

[9.1.2.  View question](#_heading=h.2iq8gzs) 82

[a. Class Diagram](#_heading=h.456o6gebskoa) 82

[b. Class Specifications](#_heading=h.fvs58fzbi3la) 83

[c. Sequence Diagram(s)](#_heading=h.4jc7dj639255) 83

[d. Database queries](#_heading=h.hy9yeqcr91g8) 84

[9.1.3. Delete question](#_heading=h.xomwdb8e5al0) 85

[a. Class Diagram](#_heading=h.w38j00qkdm1e) 85

[b. Class Specifications](#_heading=h.hud7ilvpk5a) 85

[c. Sequence Diagram(s)](#_heading=h.y1mxrxuic1el) 86

[d. Database queries](#_heading=h.d2iqbh1v73ba) 86

[9.2. Answer management](#_heading=h.9rp8hlikuug8) 87

[9.2.1. View answer](#_heading=h.rp6eb3m8ar25) 87

[a. Class Diagram](#_heading=h.qj0ukcxpptde) 87

[b. Class Specifications](#_heading=h.muylbgu6i8w5) 87

[c. Sequence Diagram(s)](#_heading=h.bezuai6c5s3r) 88

[d. Database queries](#_heading=h.gfkvkayxbw8) 88

[9.2.2. Add answer](#_heading=h.3hv69ve) 89

[a. Class Diagram](#_heading=h.3aezt1sp2ism) 89

[b. Class Specifications](#_heading=h.ngn184cqic67) 89

[c. Sequence Diagram(s)](#_heading=h.lq5b0cc21hzs) 90

[d. Database queries](#_heading=h.34uyjisholzn) 90

[9.2.3. Edit answer](#_heading=h.s2cqkv5lt1p8) 91

[a. Class Diagram](#_heading=h.mqo435wfjup0) 91

[b. Class Specifications](#_heading=h.tnso2j9jh0oh) 91

[d. Database queries](#_heading=h.he1289ltzbvs) 92

[9.2.4. Delete answer](#_heading=h.75nm5021gl93) 93

[a. Class Diagram](#_heading=h.m37o5gol8k50) 93

[b. Class Specifications](#_heading=h.eu9nnkrvkspl) 93

[c. Sequence Diagram(s)](#_heading=h.a3vx822y11c7) 94

[d. Database queries](#_heading=h.ic4qsdwwn6en) 94

[III. Database Tables](#_heading=h.2s8eyo1) **95**

[1. users\_account](#_heading=h.17dp8vu) 95

[2. user\_roles](#_heading=h.3rdcrjn) 95

[3. roles](#_heading=h.rmou7r5x2qhq) 95

[4. rate](#_heading=h.q4w6omeg8bp) 96

[5. subject](#_heading=h.mvh5b94vlh7a) 96

[6. question](#_heading=h.f91em7ytqysq) 96

[7. answer](#_heading=h.2m4khpeuc6r2) 96

[8. course](#_heading=h.qc93nzak7m7e) 97

[9. course\_student](#_heading=h.i2opxhmnkh20) 97

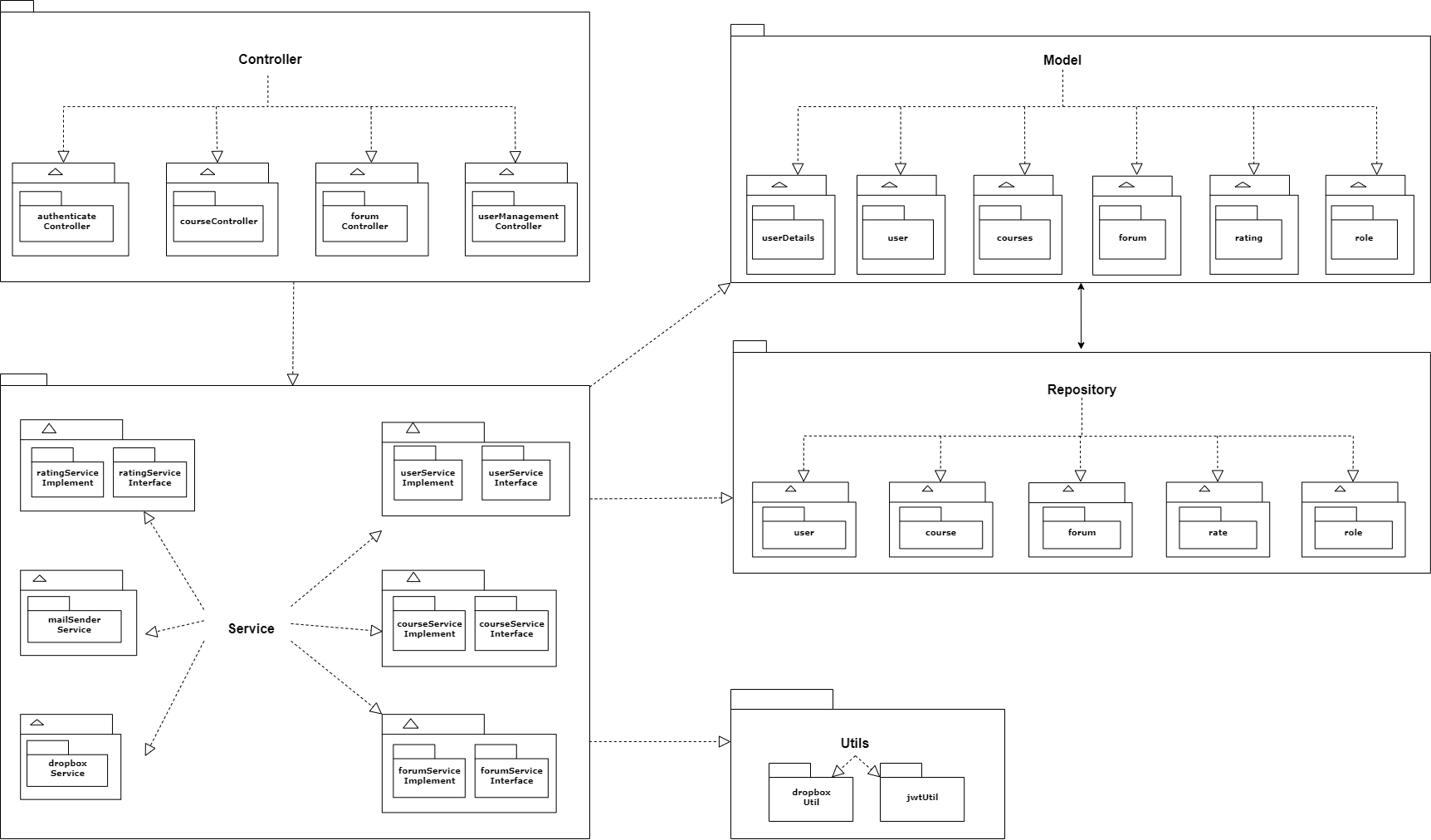
[10. course\_material](#_heading=h.rn1emxea2ewx) 97

[11. course\_timetable](#_heading=h.q14ia8ckjg7j) 98

# I. Overview

## 1. Code Packages/Namespaces

#### Package diagram

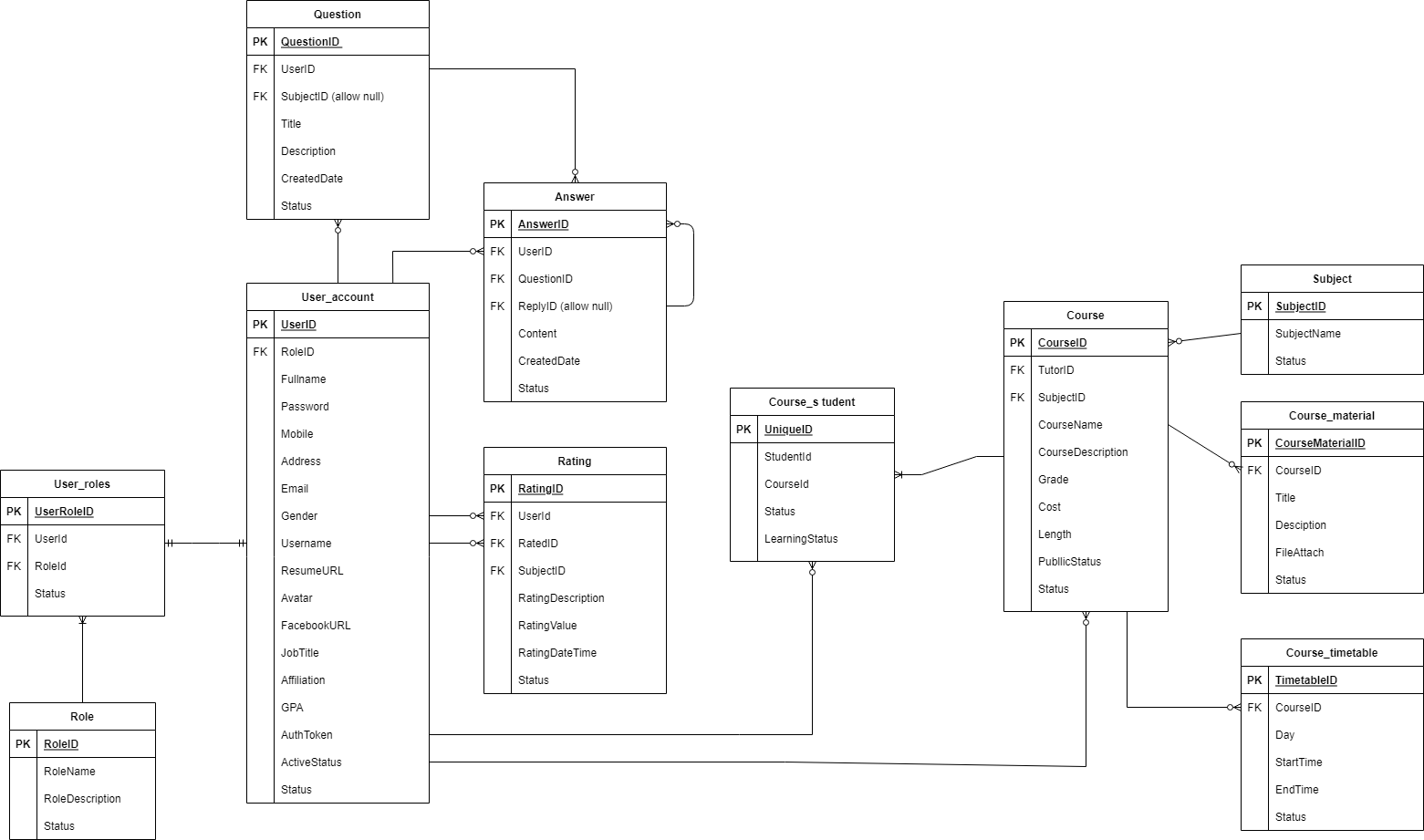


#### Package descriptions & package class naming conventions

| **No** | **Package** | **Description** |
| --- | --- | --- |
| *01* | *controller* | *Containing subpackages for to control each feature specify* |
| *02* | *service* | *Containing subpackages for to service each feature specify* |
| *03* | *model* | *Containing subpackage of each model* |
| *04* | *utils* | *Containing subpackage utils for service use* |
| *05* | *repository* | *Containing subpackage repository for model interact with database* |
| *06* | *authenticateController* | *Containing class for authenticate user such as register account, login, logout* |
| *07* | *courseController* | *Containing classes to control interact between user with the course* |
| *08* | *forumController* | *Containing classes to control interact between user with the forum* |
| *09* | *userController* | *Containing classes to control interact between user with other user or themself* |
| *10* | *userServiceInterface* | *Containing interface classes used to authenticate user and handle request from client* |
| *11* | *userServiceImplement* | *Containing classes implement functions in userServiceInterface* |
| *12* | *courseServiceInterface* | *Containing interface classes used to handle request from client* |
| *13* | *courseServiceImplement* | *Containing classes implement functions in courseServiceInterface* |
| *14* | *forumServiceInterface* | *Containing interface classes used to handle request from client* |
| *15* | *forumServiceImplement* | *Containing classes implement functions in forumServiceInterface* |
| *16* | *ratingServiceInterface* | *Containing interface classes used to handle request from client* |
| *17* | *mailSenderService* | *Containing classes used to handle request from service layer to send email* |
| *18* | *dropboxService* | *Containing classes used to handle request from service layer to interact with dropbox* |
| *19* | *user::model* | *Containing classes define attributes of user* |
| *20* | *course::model* | *Containing classes define attributes of course* |
| *21* | *role::model* | *Containing classes define role of each user* |
| *22* | *rate::model* | *Containing classes define attributes of rate* |
| *23* | *forum::model* | *Containing classes define attributes of forum* |
| *24* | *userDetails::model* | *Containing classes define attributes of user to authenticate* |
| *25* | *user::repository* | *Containing interfaces used to handle request between user and database* |
| *26* | *course::repository* | *Containing interfaces used to handle request between course and database* |
| *27* | *rate::repository* | *Containing interfaces used to handle request between rate and database* |
| *28* | *role::repository* | *Containing interfaces used to handle request between user role and database* |
| *29* | *dropboxUtil* | *Containing classes used to provide method for dropboxService to use when handle request from client* |
| *30* | *jwtUtil* | *Containing classes used to provide jwt information when user log in to the application* |

## 2. Database Schema

#### a. Entities relationship diagram



#### b. Table Description

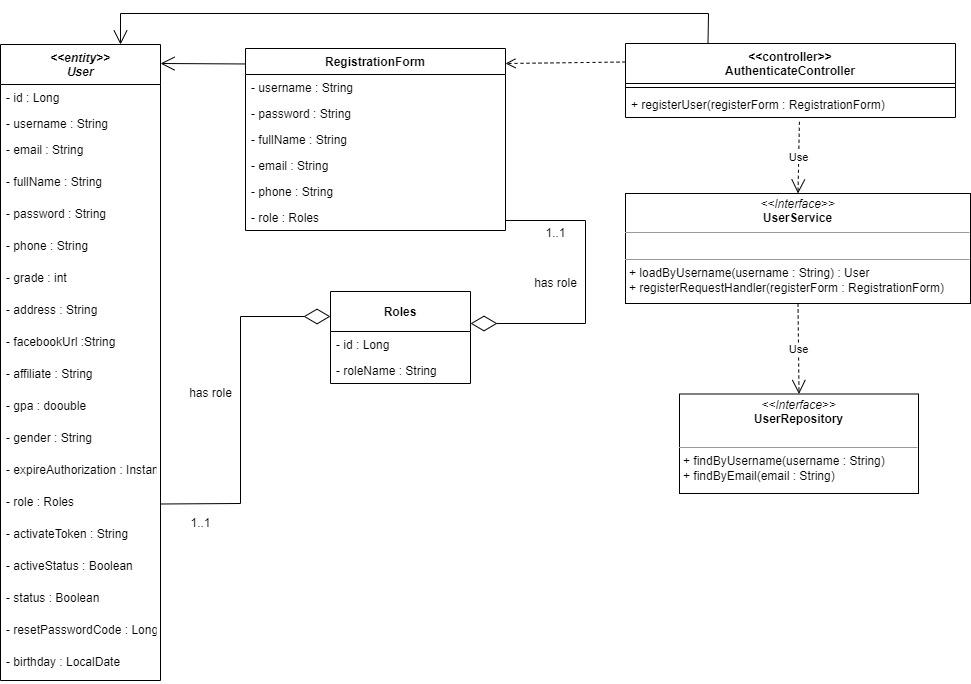
| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | Roles | Contains the roles that users in the system are able to have  - Primary keys: RoleID |
| 02 | User\_account | Contains basic information of users  - Primary keys: UserID |
| 03 | User\_roles | Map role is the roles table to each user in the User\_account table  - Primary keys: RoleID  - Foreign keys: UserID, RoleID |
| 04 | Question | Contains basic ìnormation of a question posted by student in forum feature  - Primary keys: QuestionID  - Foreign keys: UserID, SubjectID |
| 05 | Answer | Contains detail of a answer to a question  - Primary keys: AnswerID  - Foreign keys: QuestionID, UserID, ReplyID |
| 06 | Rating | Rating detail of a user to another user  - Primary keys: RateID  - Foreign keys: StudentID, TutorID, SubjectID |
| 07 | Course | Contains basic information of a course  - Primary keys: CourseID  - Foreign keys: SubjectID, TutorID |
| 08 | Subject | Subject that each courses or questions can be related  - Primary keys: SubjectID |
| 09 | Course\_student | Listing registered students of a course  - Primary keys: CourseStudentID  - Foreign keys: StudentId, CourseID |
| 10 | Course\_material | Contains materials of a course  - Primary keys: CourseMaterialD  - Foreign keys: CourseID |
| 11 | Course\_timetable | Contains timetable of a course  - Primary keys: CourseTimetableID  - Foreign keys: CourseID |

# II. Code Designs

## 1. Authentication

### 1.1. Register new account

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of User.

##### Roles Class

* Model of roles.

##### RegistrationForm Class

* Controller will get a Request with this format.

##### UserRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findByUsername* | *- This method finds and gets a User by existing name.* |
| *02* | *findByEmail* | *- This method finds and gets a User by existing email.* |

##### AuthenticateController Class

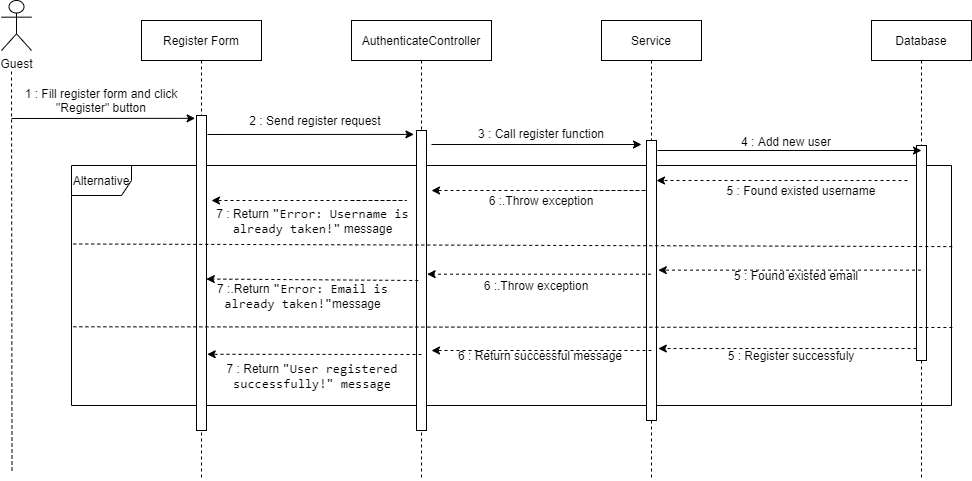
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *RegisterUser* | *- This method will call API Register by using a service from UserService.* |

##### 

##### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *LoadByUserName* | *- This method will find and get all users By userName.* |
| *02* | *registerRequestHandler* | *- This method will perform Registerfunction.* |

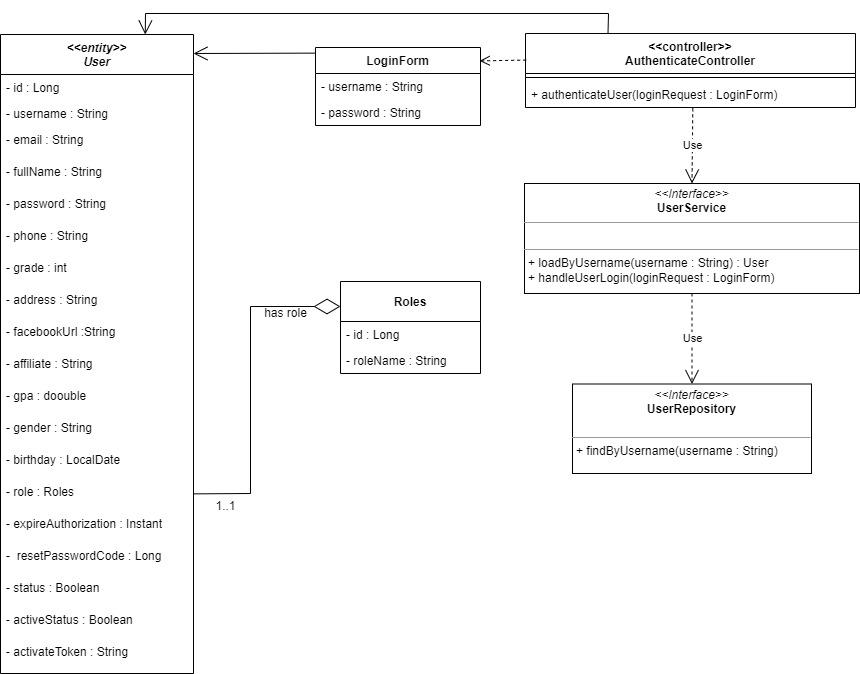
#### c. Sequence Diagram(s)



#### d. Database queries

### 1.2. Login

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of User.

##### Roles Class

* Model of roles.

##### LoginForm Class

* Controller will get a Request with this format.

##### UserRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findByUsername* | *- This method find and get User by existing name.* |

##### 

##### 

##### 

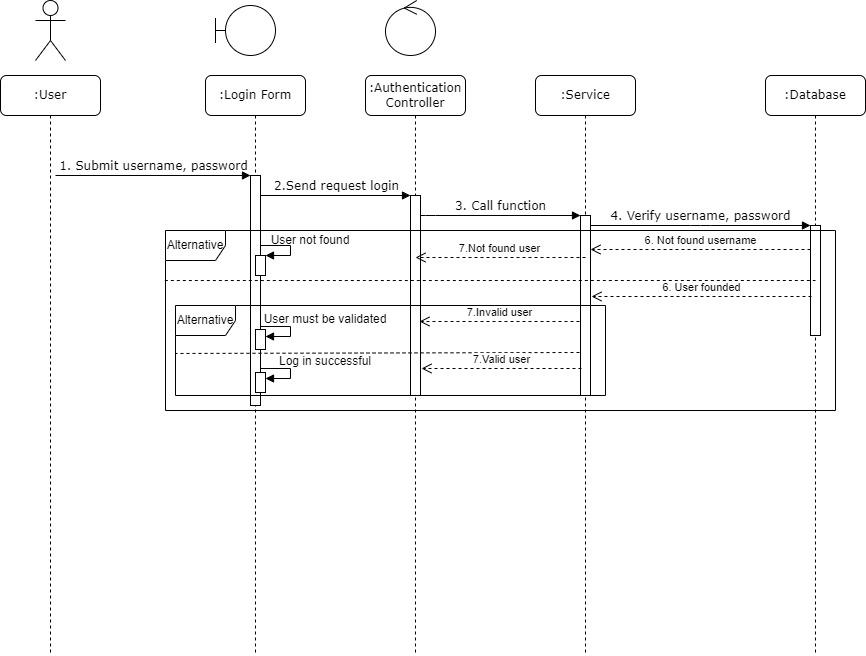
##### AuthenticateController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *authenticate* | *- This method will call API Login by using service from UserService.* |

##### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *LoadByUserName* | *- This method will find and get all user By UserName.* |
| *02* | *handleUserLogin* | *- This method will perform Login function.* |

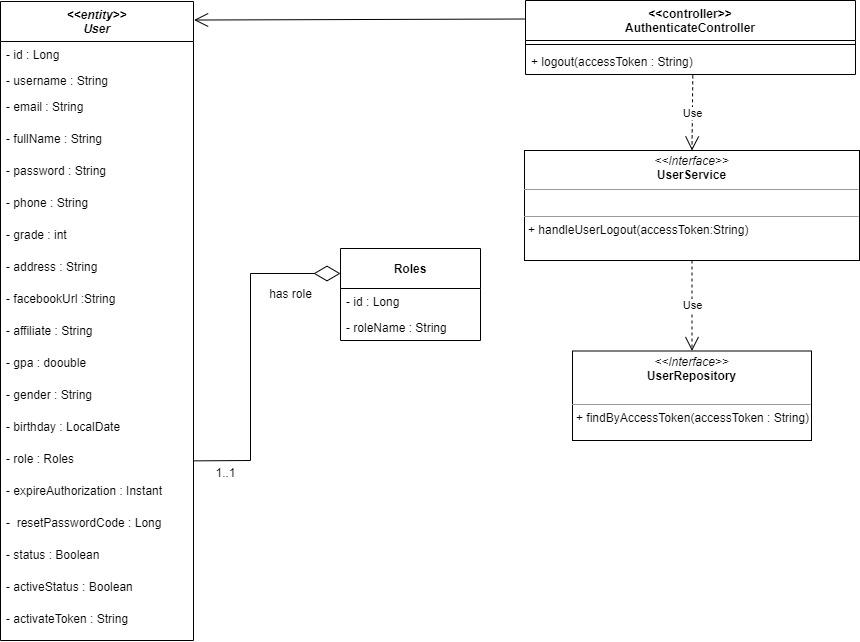
#### c. Sequence Diagram(s)



#### d. Database queries

### 1.3. Log out

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of User.

##### Roles Class

* Model of roles.

##### UserRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findByAccessToken* | *- This method finds and gets User by their Access Token.* |

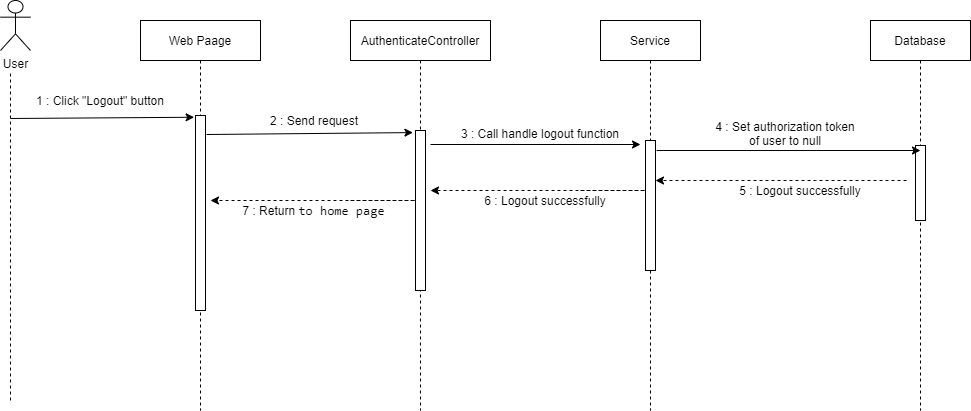
##### AuthenticateController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *logout* | *- This method will call API Logout by using a service from UserService.* |

##### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *handleUserLogout* | *- This method will perform Logout function.* |

#### c. Sequence Diagram(s)

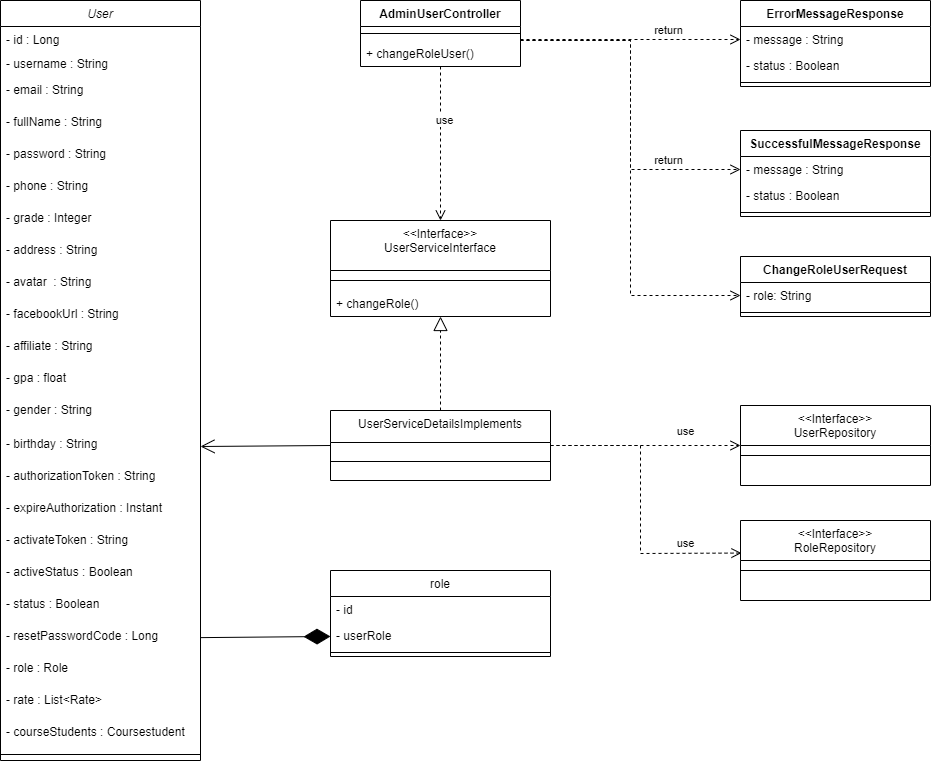


#### d. Database queries

## 2. User Management

### 2.1. Change user’s role

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of user class

##### Role Class

* Model of role class

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### SuccessfulMessageResponse Class

* Controller API will return this format if action successful

##### ChangeRoleUserRequest Class

* Controller API will receive this request to perform action

##### AdminUserController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *changeRoleUser* | *This method will call service to handle change role request* |

##### UserServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | changeRole | *This method are using to perform change role of user* |

##### UserServiceDetailsImplements Class

* This class will implement the UserServiceInterface Interface

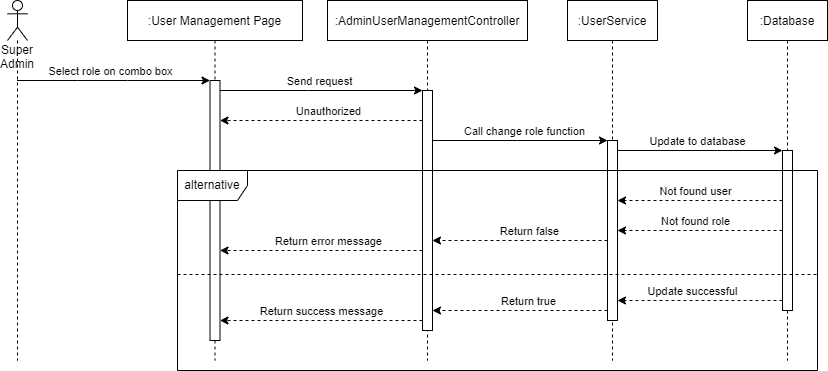
##### UserRepository Interface

* This interface are used to query data on User Table

##### RoleRepository Interface

* This interface are used to query data on Role Table

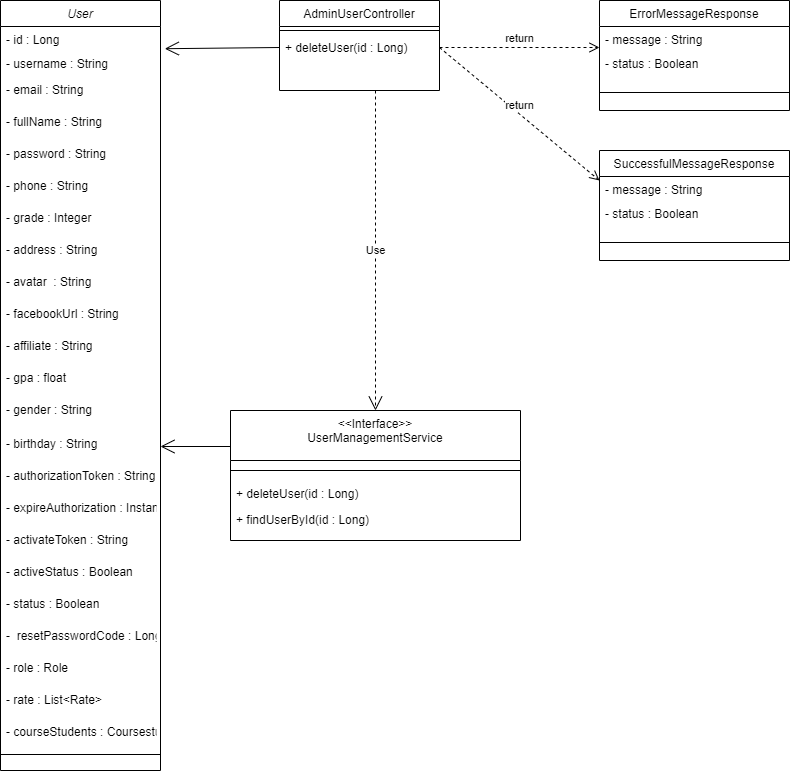
#### c. Sequence Diagram(s)



#### d. Database queries

### 2.2. Delete user

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* model of User.

##### ErrorMessageResponse Class

* Controller API will return with this format response.

##### SuccessfulMessageResponse Class

* Controller API will return with this format response.

##### AdminUserController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteUser | - This method will call the API of DeleteUser. |

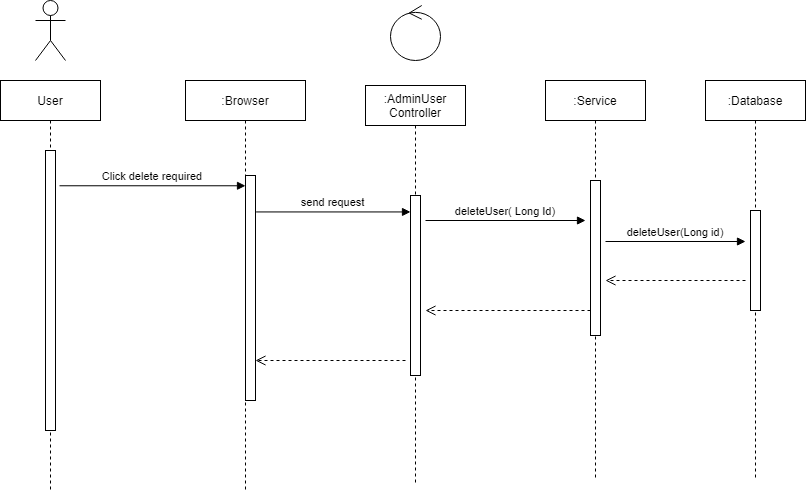
### 

##### UserManagement Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteUser | - This method allows a user with the SuperAdmin role to delete a user by finding the User ID and setting the user’s status to False ( Deleted ).  - SuperAdmin can not delete other SuperAdmin. |
| 02 | findUserById | - This method allows you to find a user by user ID. |

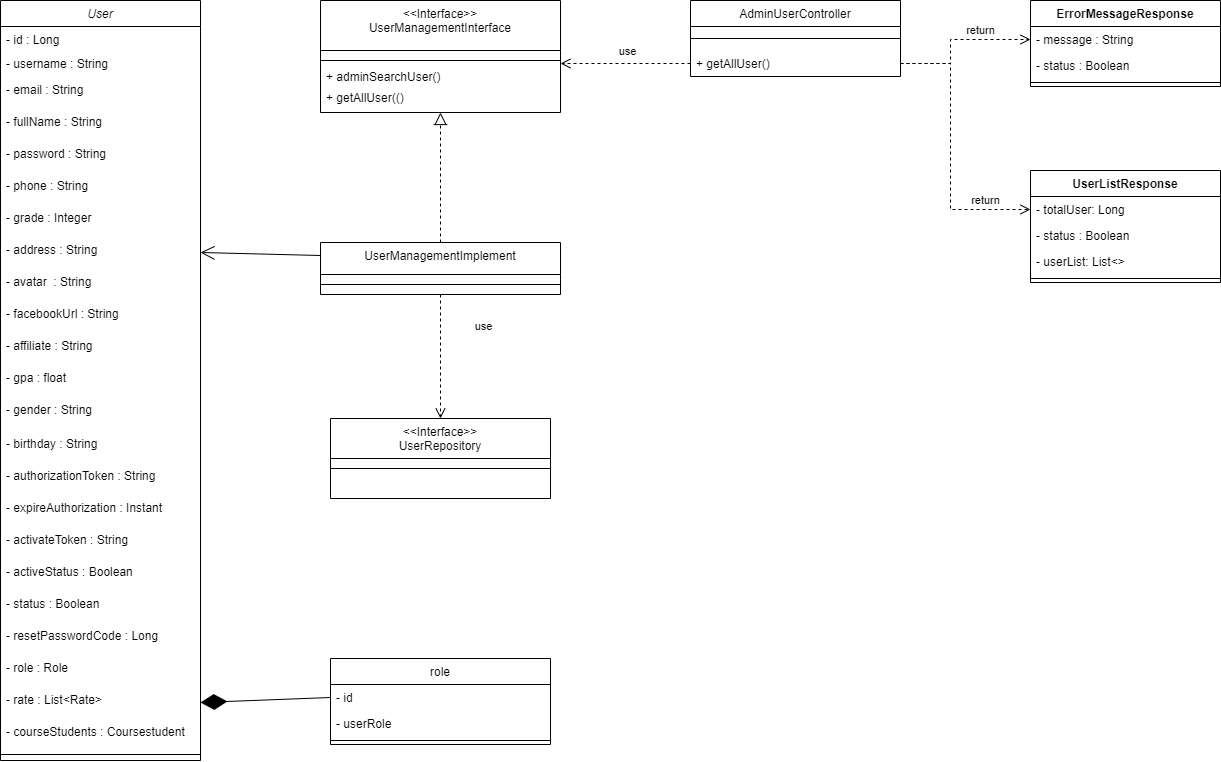
#### c. Sequence Diagram(s)

#### d. Database queries



### 2.3. View user

#### a. Class Diagram



#### b. Class Specifications

##### Role Class

* Model of role

##### User Class

* Model of user

##### ErrorMessageResponse

* Controller API will return this format if error occur

##### UserListResponse

* Controller API will use return User Information list with this format

##### AdminUserController Class

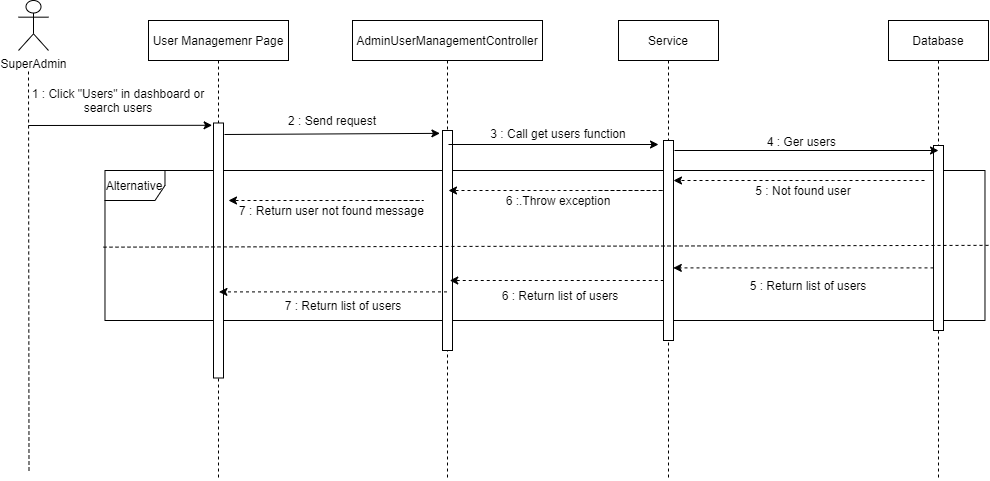
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllUser* | *This method call service to handle get user list* |

##### 

***UserManagementInterface Interface***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllUser* | *This method returns all users with pagination* |
| *02* | *adminSearchUser* | *This method returns searched users with pagination.* |

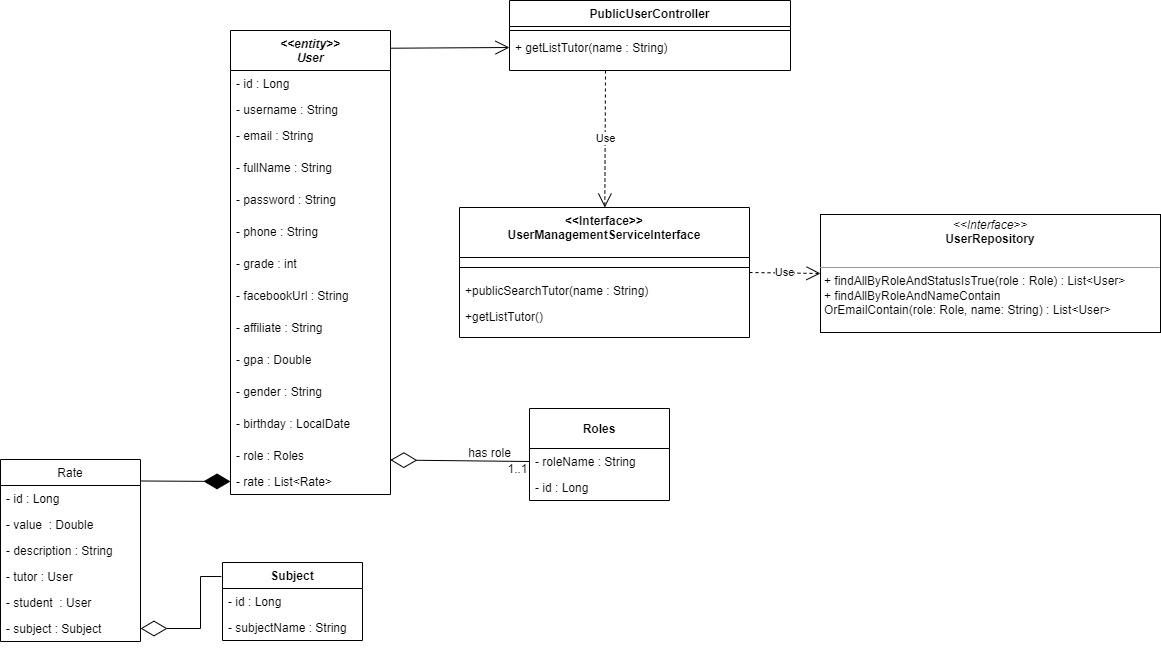
#### c. Sequence Diagram(s)



#### d. Database queries

### 2.4. Search tutor

#### a. Class Diagram

**

#### b. Class Specifications

##### User Class

* Model of user.

##### Rate Class

* Model of rate.

##### Roles Class

* Model of roles.

##### Subject Class

* Model of subject.

##### PublicUserManagementController Class

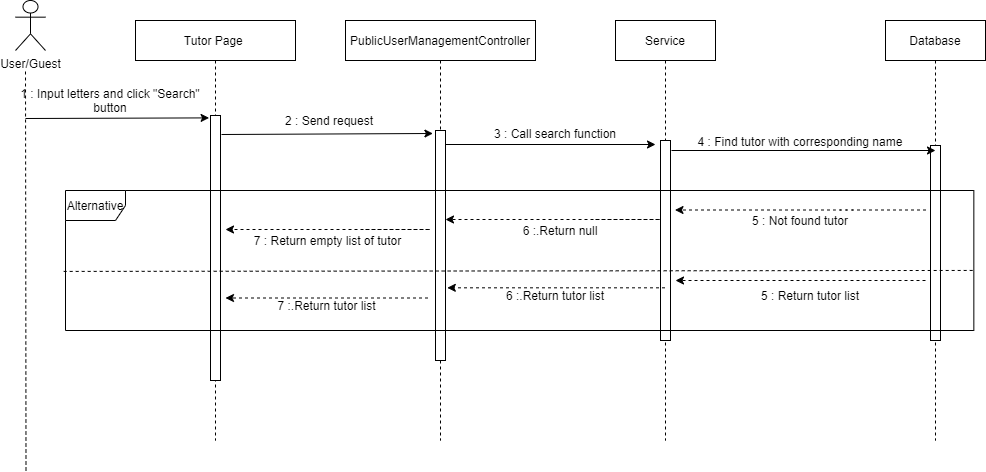
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getListTutor* | *Call the API of publicSearchTutor* |

#### 

##### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *publicSearchTutor* | *This method returns a list of tutors with their names.* |
| *02* | *findByUserRole* | *Find the user by role.* |
| *03* | *findAllByStatusIsTrueAndRolesAndEmailContainsOrRolesAndFullNameContainingOrderByIdDesc* | *Finds the tutors with corresponding letters in their names.* |

#### c. Sequence Diagram(s)

**

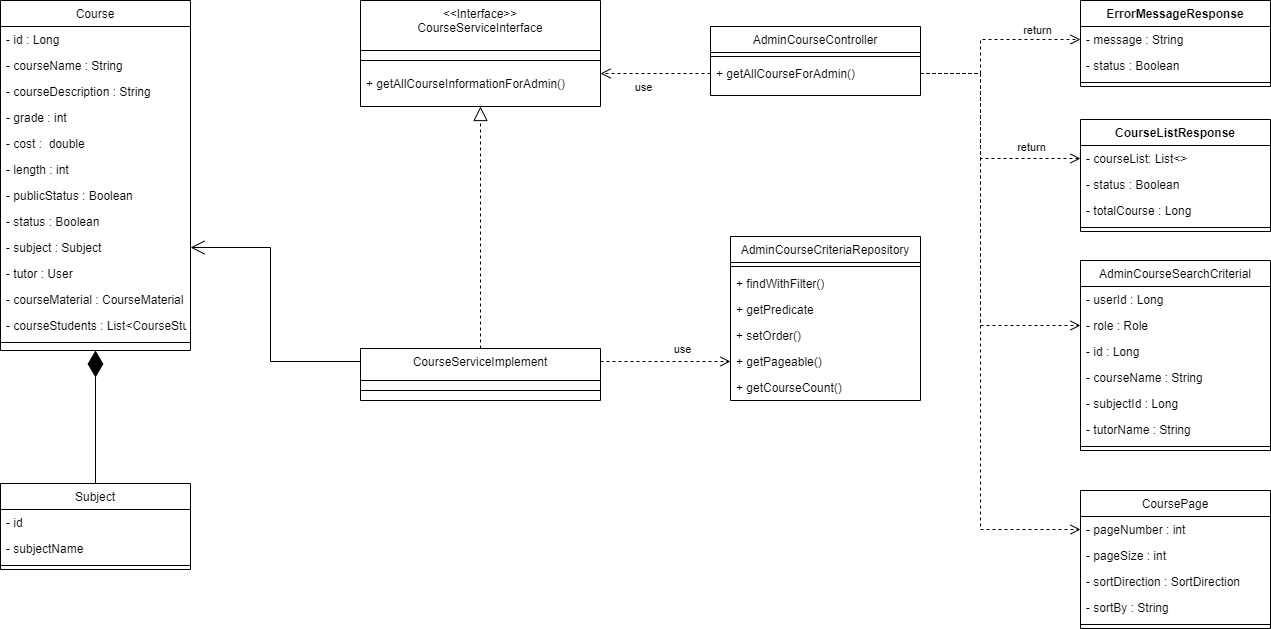
#### d. Database queries

## 3. Course Management

### 3.1. View course

### 3.1.1 View all course (admin)

#### a. Class Diagram



#### b. Class Specifications

##### Course Class

* Model of course

##### Subject Class

* Model of subject

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### CourseListResponse Class

* Controller API will return this Course Information list format if action perform successful

##### CoursePage class

* Controller API will use this class to perform pagination

##### AdminCourseSearchCriteria

* This class contains search/filter option for Controller API to perform view course

##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCourseForAdmin* | *This method will call service to handle view course request* |

##### CourseServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getAllCourseInformationForAdmin | *This method are using to perform view course* |

##### CourseServiceDetailsImplements Class

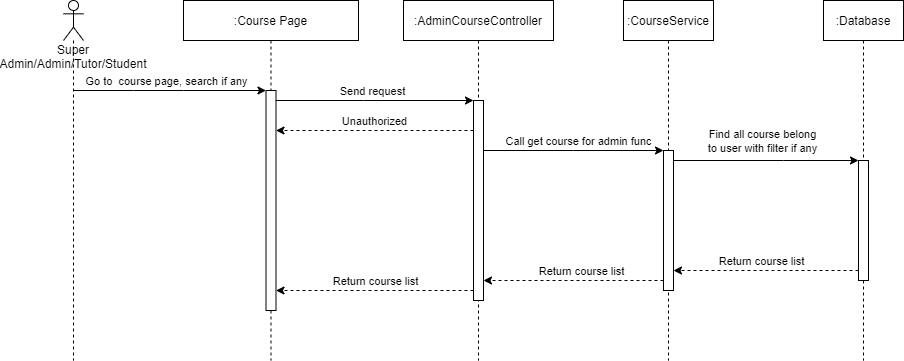
* This class will implement the CourseServiceInterface Interface

##### AdminCourseCriteriaRepository Interface

##### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | findWithFilter() | *This method are using to perform get course list with filter* |
| *02* | getPredicate() | *This method are using to add predicate for each exist search option* |
| *03* | setOrder() | *This method are using to set course list order* |
| *04* | getPageable() | *This method are using for pagination* |
| *05* | getCourseCount() | *This method are using for counting total course* |

#### c. Sequence Diagram(s)

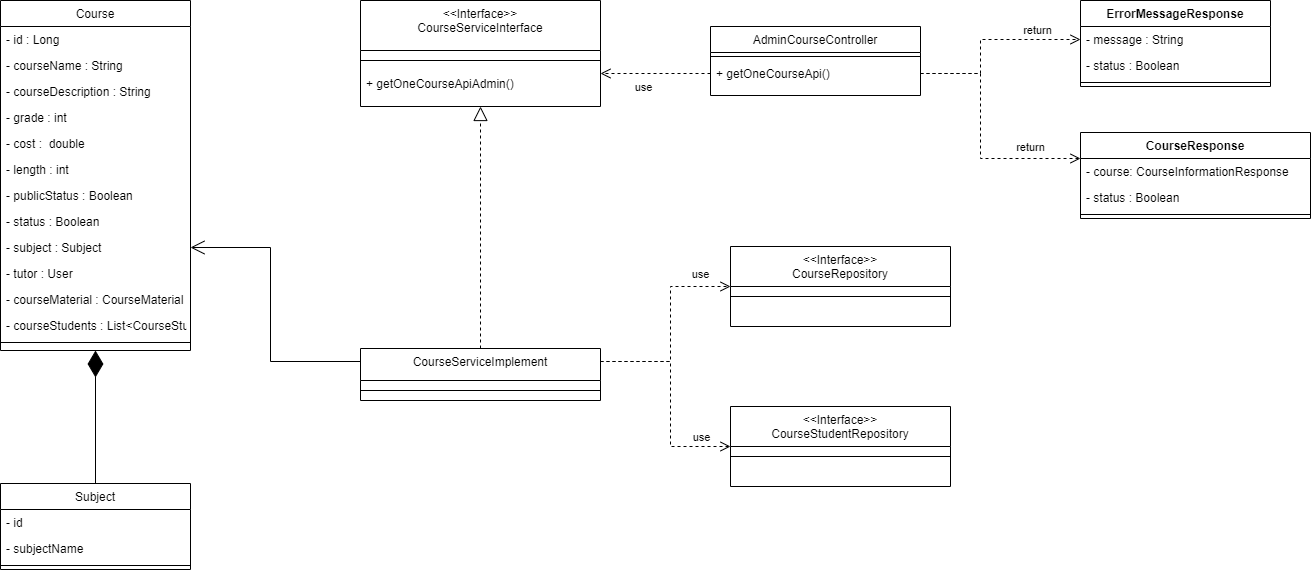


#### d. Database queries

#### 

### 3.1.2 View details one course (admin)

#### a. Class Diagram



#### b. Class Specifications

##### Course Class

* Model of course

##### Subject Class

* Model of subject

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### CourseResponse Class

* Controller API will return this Course Information format if action perform successful

##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getOneCourseApi* | *This method will call service to handle view one course request* |

##### CourseServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getOneCourseApi* | *This method are using to perform view one course* |

##### CourseServiceDetailsImplements Class

* This class will implement the CourseServiceInterface Interface

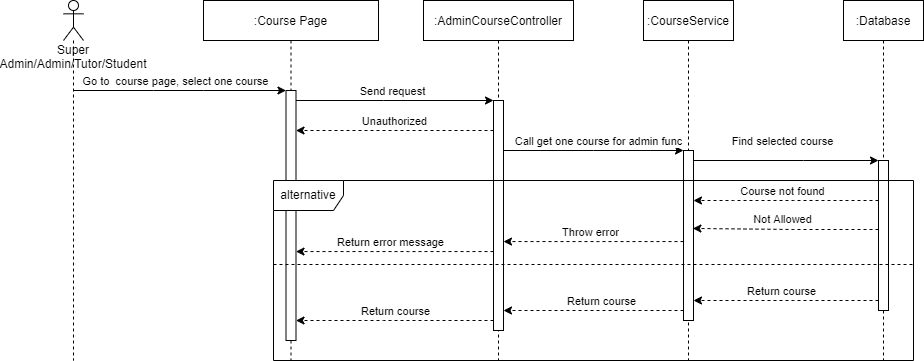
##### CourseRepository Interface

* This interface are used to query data on Course Table

##### CourseStudentRepository Interface

* This interface are used to query data on CourseStudent Table

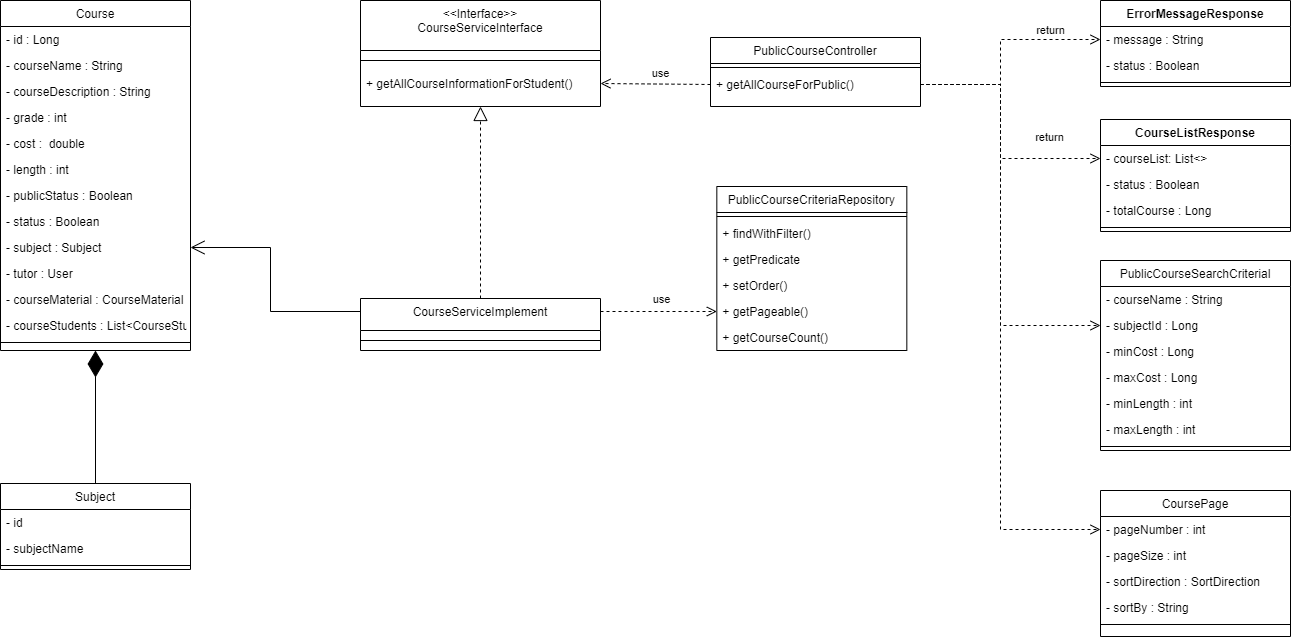
#### c. Sequence Diagram(s)



#### d. Database queries

### 3.1.3 View all course (public)

#### a. Class Diagram



#### b. Class Specifications

##### Course Class

* Model of course

##### Subject Class

* Model of subject

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### CourseListResponse Class

* Controller API will return this Course Information list format if action perform successful

##### CoursePage class

* Controller API will use this class to perform pagination

##### PublicCourseSearchCriteria

* This class contains search/filter option for Controller API to perform view course

##### PublicCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCourseForPublic* | *This method will call service to handle view course request* |

##### CourseServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getAllCourseInformationForPublic | *This method are using to perform view course* |

##### CourseServiceDetailsImplements Class

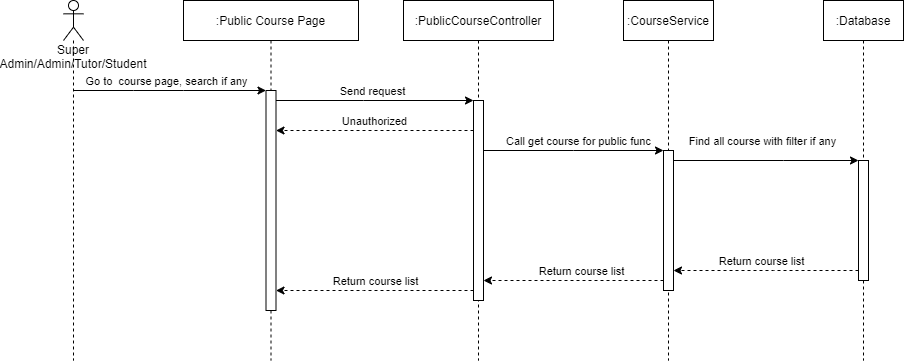
* This class will implement the CourseServiceInterface Interface

##### PublicCourseCriteriaRepository Class

##### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | findWithFilter() | *This method are using to perform get course list with filter* |
| *02* | getPredicate() | *This method are using to add predicate for each exist search option* |
| *03* | setOrder() | *This method are using to set course list order* |
| *04* | getPageable() | *This method are using for pagination* |
| *05* | getCourseCount() | *This method are using for counting total course* |

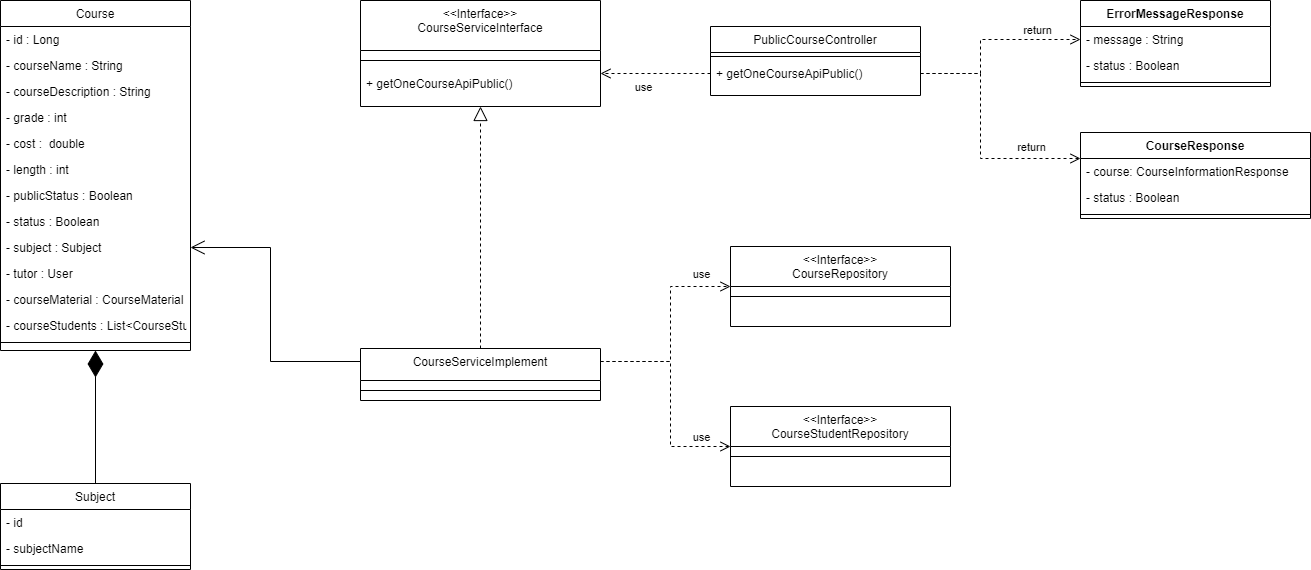
#### c. Sequence Diagram(s)



#### d. Database queries

### 3.1.2 View details one course (public)

#### a. Class Diagram



#### b. Class Specifications

##### Course Class

* Model of course

##### Subject Class

* Model of subject

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### CourseResponse Class

* Controller API will return this Course Information format if action perform successful

##### PublicCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getOneCourseApiPublic* | *This method will call service to handle view one course request* |

##### CourseServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getOneCourseApiPublic* | *This method are using to perform view one course* |

##### CourseServiceDetailsImplements Class

* This class will implement the CourseServiceInterface Interface

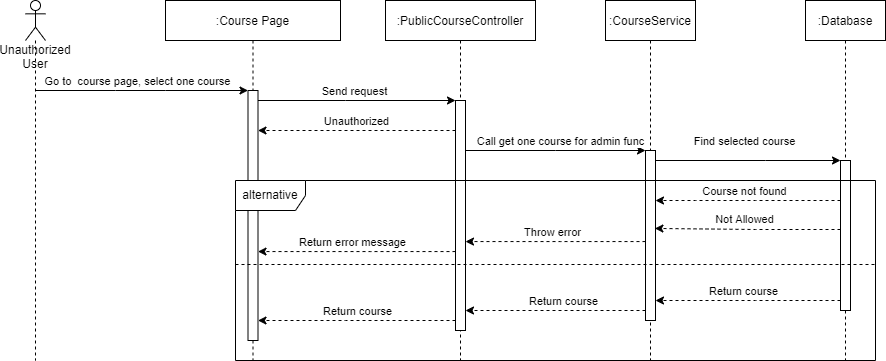
##### CourseRepository Interface

* This interface are used to query data on Course Table

##### CourseStudentRepository Interface

* This interface are used to query data on CourseStudent Table

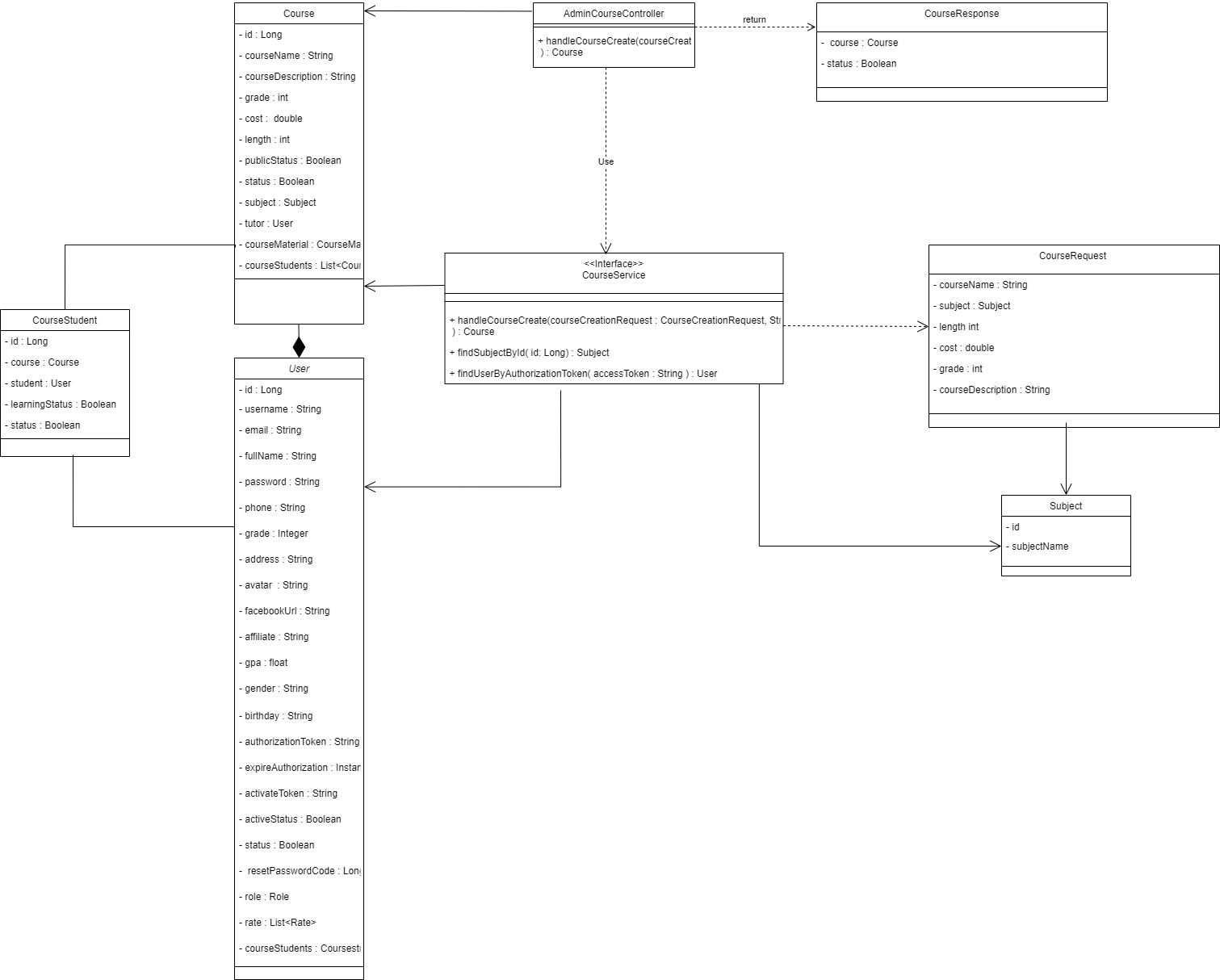
#### c. Sequence Diagram(s)



#### d. Database queries

### 3.2. Create course

#### a. Class Diagram



#### b. Class Specifications

##### CourseStudent Class

* Model of specific student of one course.

##### User Class

* Model of User.

##### Course Class

* Model of Course .

##### CourseResponse Class

* Controller will return with this format response.

##### CourseCreateRequest Class

* Controller will receive a Request with this format.

##### Subject Class

* Model of Subject.

##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | handleCourseCreate | - This method will call the API of handleCourseCreate . |

#### 

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | handleCourseCreate | - This method allows users with roles Tutor, Admin, SuperAdmin to create courses through CourseCreateRequest and return objects. |
| 02 | findSubjectById | - Finding a Subject by Id. |
| 03 | findUserByAuthorizationToken | - Find a User by Authorization Token. |

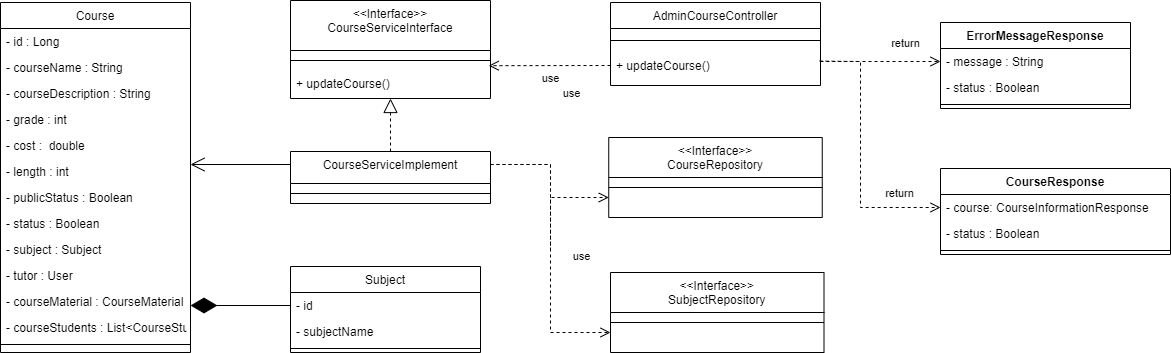
#### c. Sequence Diagram(s)

#### 

#### d. Database Queries

### 3.3. Update course

#### a. Class Diagram



#### b. Class Specifications

##### Course Class

* Model of course

##### Subject Class

* Model of subject

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### CourseResponse Class

* Controller API will return this Course Information format if update successful

##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateCourse* | *This method will call service to handle update course request* |

##### CourseServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | changeRole | *This method are using to perform update course* |

##### CourseServiceDetailsImplements Class

* This class will implement the CourseServiceInterface Interface

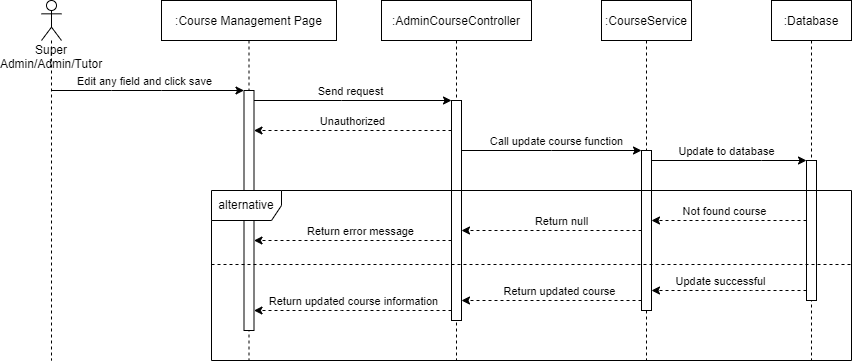
##### SubjectRepository Interface

* This interface are used to query data on Subject Table

##### CourseRepository Interface

* This interface are used to query data on Course Table

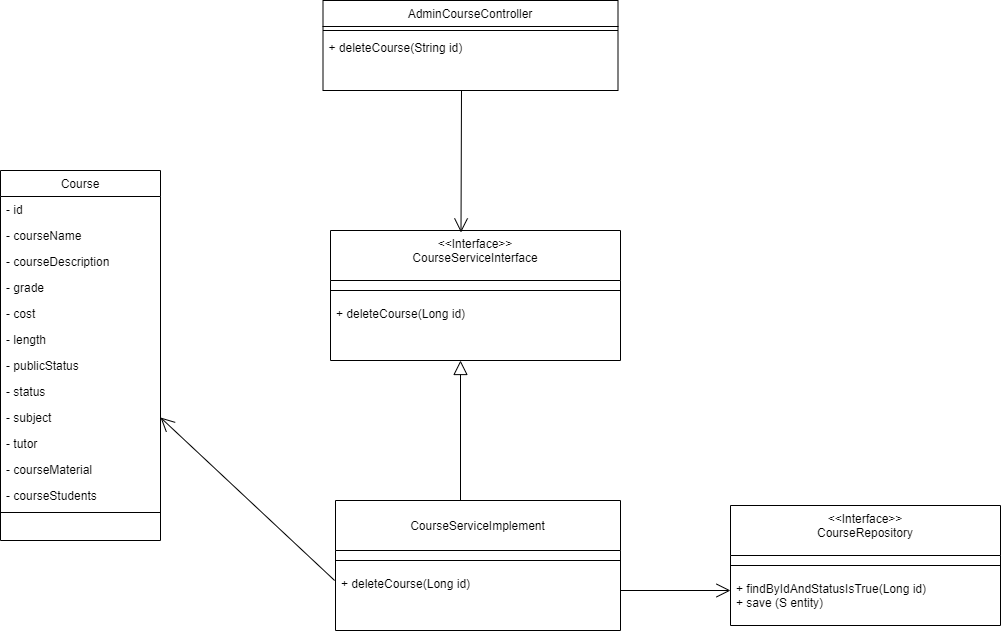
#### c. Sequence Diagram(s)



#### d. Database queries

### 3.4. Delete course

#### a. Class Diagram



#### b. Class Specifications

#### 

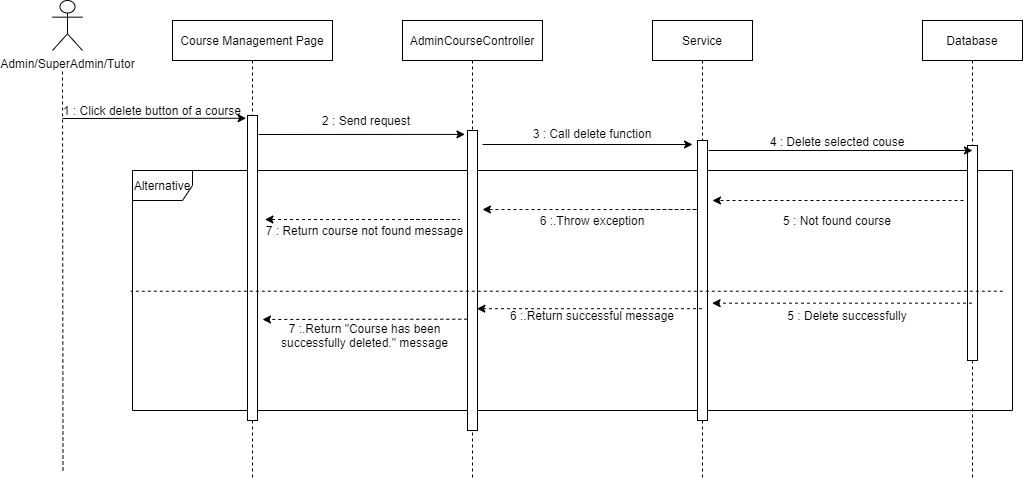
##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deleteCourse* | *Call the API of deleteCourse* |

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findByIdAndStatusIsTrue* | *Find a course with id and status is true.* |
| *02* | *SetStatus* | *Change the status of a course to false (soft delete).* |
| *03* | *setPublicStatus* | *Change the public status of a course to false (soft delete).* |

#### c. Sequence Diagram(s)

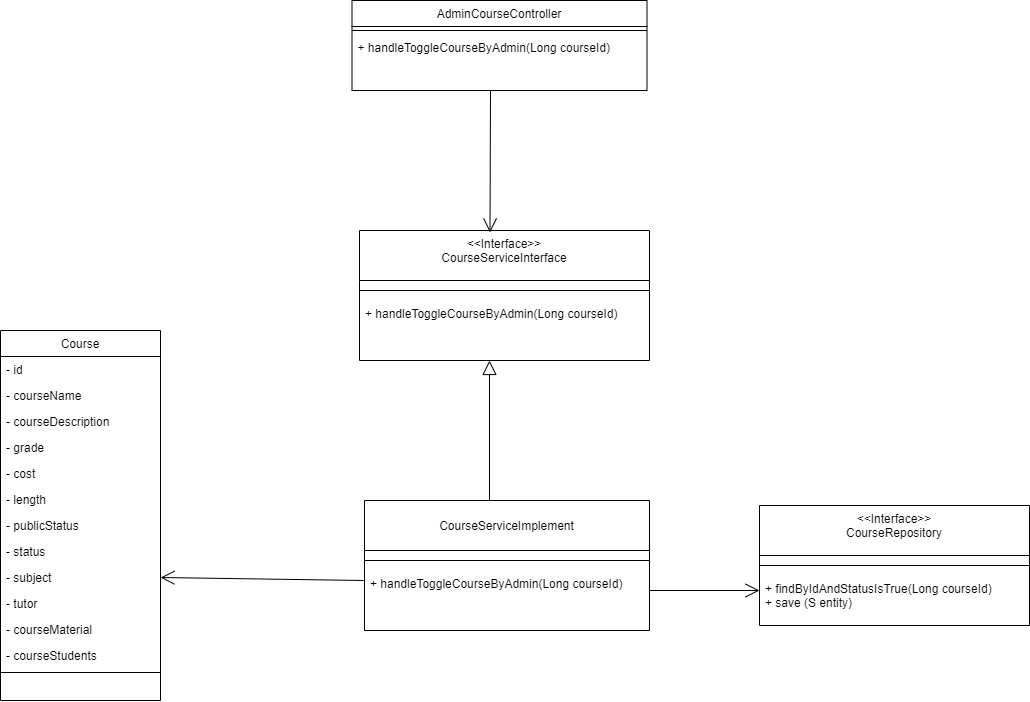


#### d. Database queries

#### 

### 3.6 Toggle course

#### a. Class Diagram

**

#### b. Class Specifications

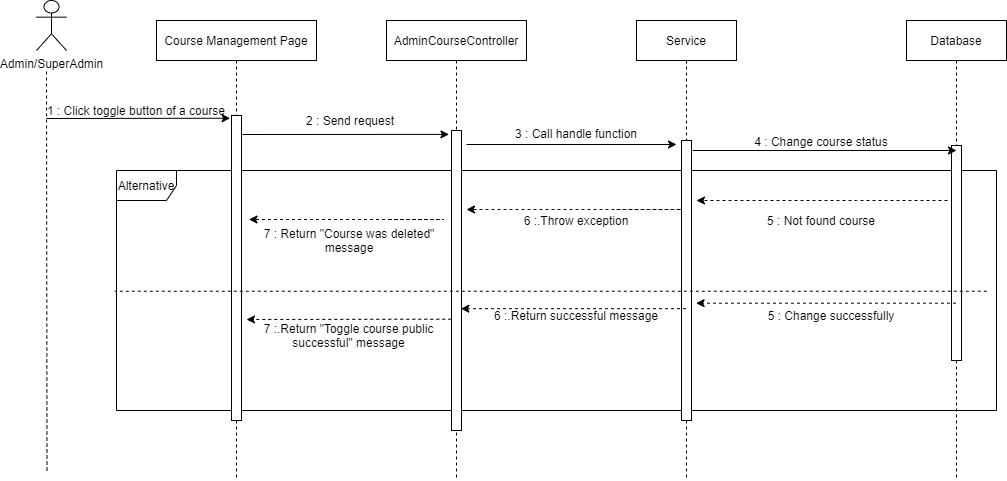
##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *handleToggleCourseByAdmin* | *Call the API of handleToggleCourse* |

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *handleToggleCourseByAdmin* | *This method changes the public status of a course, making it visible to everyone.* |
| *02* | *findByIdAndStatusIsTrue* | *Find a course with id and status is true.* |
| *03* | *setPublicStatus* | *Change the public status of a course to True.* |

#### c. Sequence Diagram(s)

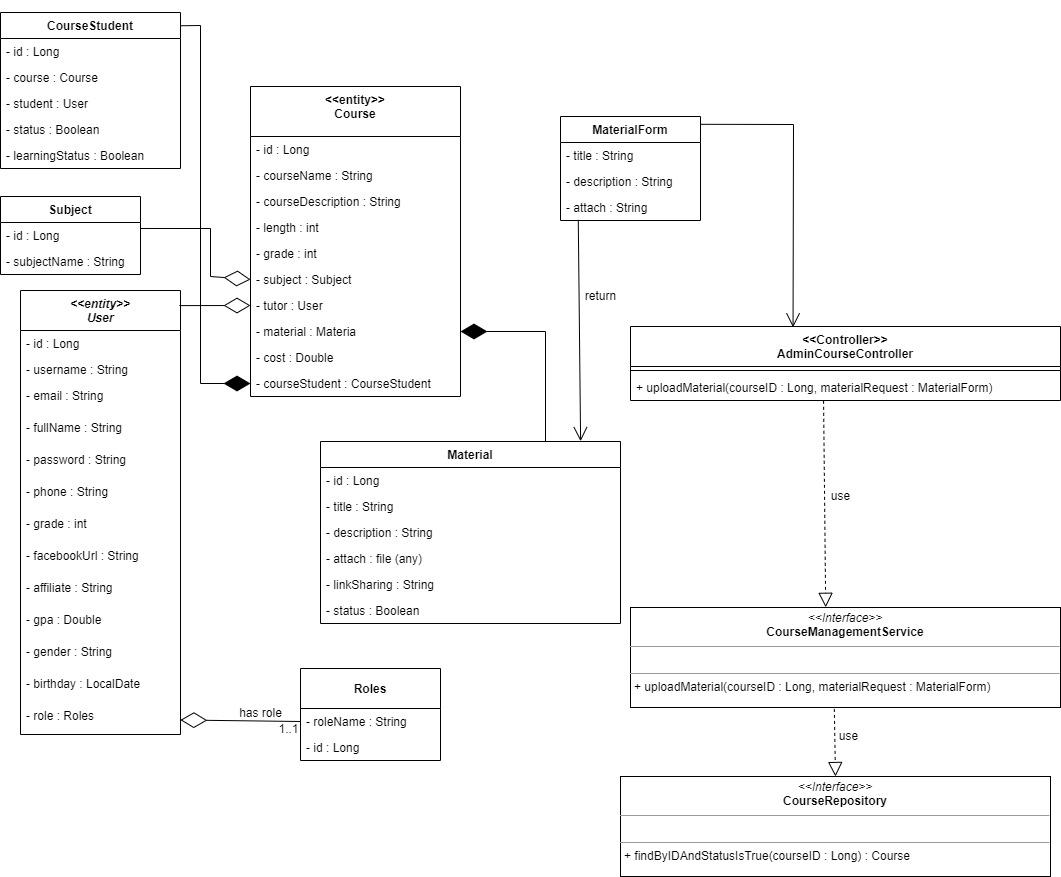
**

#### d. Database queries

## 4. Course Material Management

### 4.1. Create course material

#### a. Class Diagram



#### b. Class Specifications

##### CourseMaterial Class

* Model of specific material of one course.

##### User Class

* Model of User.

##### Course Class

* Model of Course .

***AdminCourseController Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | uploadMaterial | - This method used to receive request to create material from client then call service layer to implement request |

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | uploadMaterial | - This method allows users with roles Admin, SuperAdmin or owner this course to upload the new material |

##### CourseRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByIdAndStatusIsTrue | - This is query command using Jpa framework to find out specify course match with ID from parameter and have status is true |

##### 

#### c. Sequence Diagram(s)

#### d. Database queries

### 4.2. View course material

#### a. Class Diagram

#### b. Class Specifications

##### Course Class

* Model of user

##### CourseMaterial Class

* Model of CourseMaterial

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### MaterialListResponse Class

* Controller API will return this Course Material Information list format if action perform successful

##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllMaterial()* | *This method will call service to handle view course material request* |

##### CourseServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getCourseMaterial() | *This method are using to perform view course material* |

##### CourseServiceDetailsImplements Class

* This class will implement the CourseServiceInterface Interface

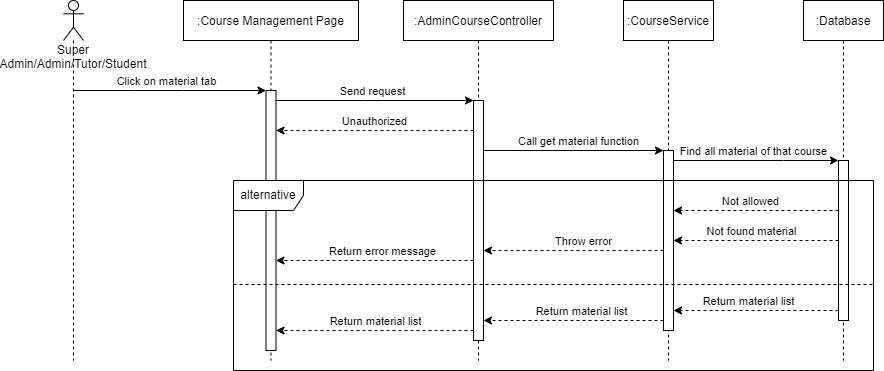
##### CourseMaterialRepository Interface

* This interface are used to query data on Course Material Table

##### CourseRepository Interface

* This interface are used to query data on Course Table

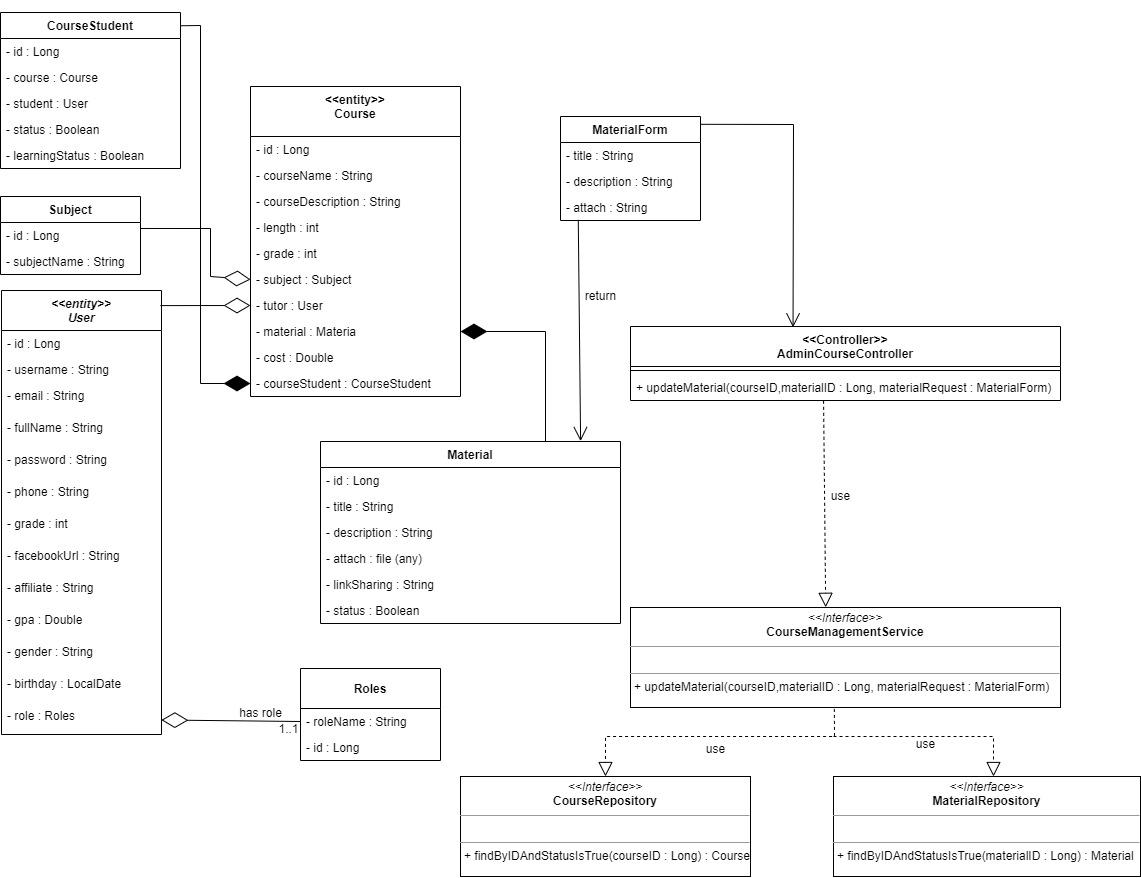
#### c. Sequence Diagram(s)



#### d. Database queries

### 4.3. Update course material

#### a. Class Diagram



#### b. Class Specifications

##### CourseMaterial Class

* model of specific material of one course.

##### User Class

* model of User.

##### Course Class

* model of Course .

***AdminCourseController Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateMaterial | - This method used to receive request to create material from client then call service layer to implement request |

#### 

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateMaterial | - This method allows users with roles Admin, SuperAdmin or owner this course to update the material existed |

##### CourseRepository Class

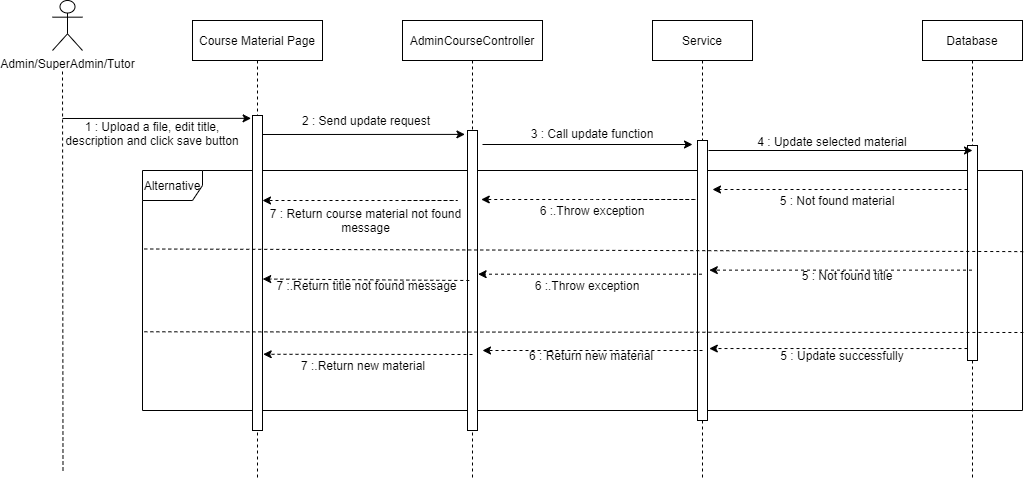
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByIdAndStatusIsTrue | - This is query command using Jpa framework to find out specify course match with ID from parameter and have status is true |

##### 

##### MaterialRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByIdAndStatusIsTrue | - This is query command using Jpa framework to find out specify material match with ID from parameter and have status is true |

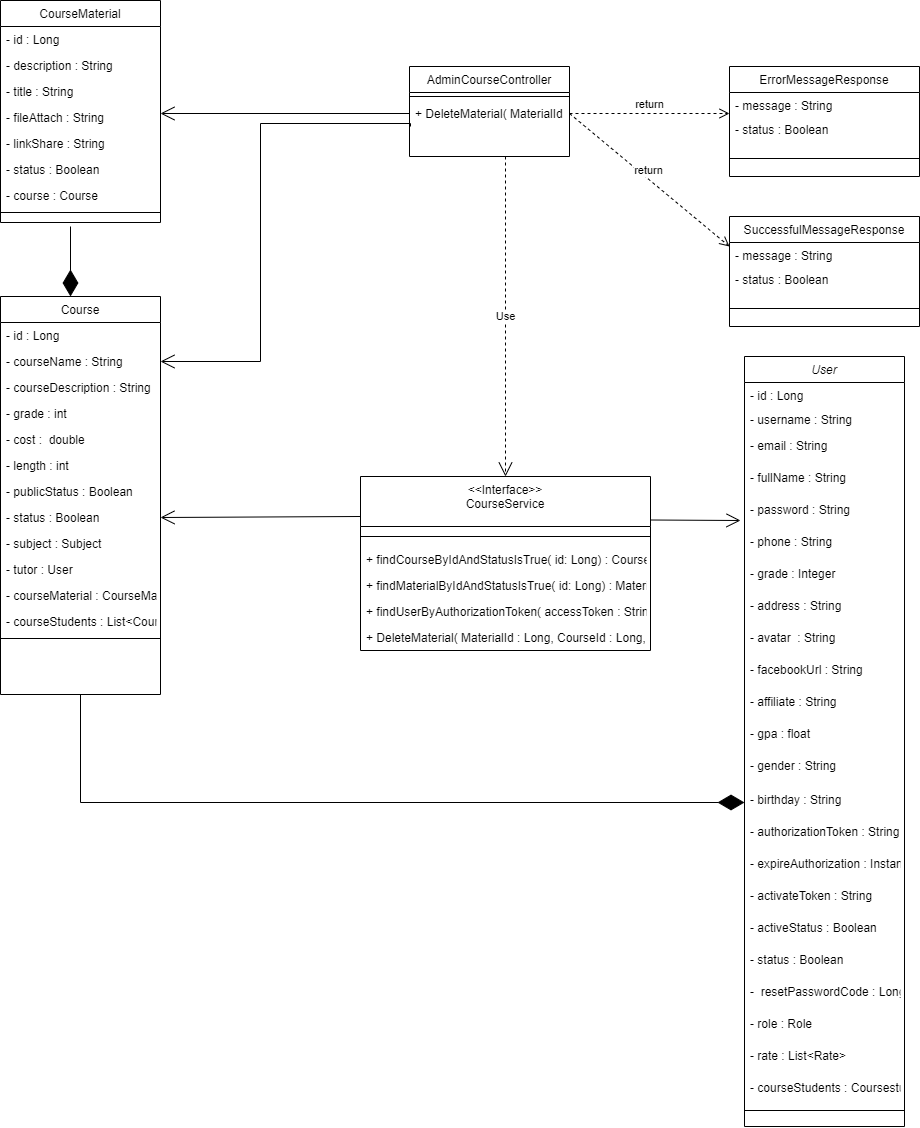
#### c. Sequence Diagram(s)

**

#### d. Database queries

### 4.4. Delete course material

#### a. Class Diagram



#### b. Class Specifications

##### CourseMaterial Class

* Model of specific material of one course.

##### User Class

* Model of User.

##### Course Class

* Model of Course .

##### ErrorMessageResponse Class

* Controller will return with this format response.

##### SuccessfulMessageResponse Class

* Controller will return with this format response.

##### Subject Class

* Model of Subject.

##### AdminUserController Class

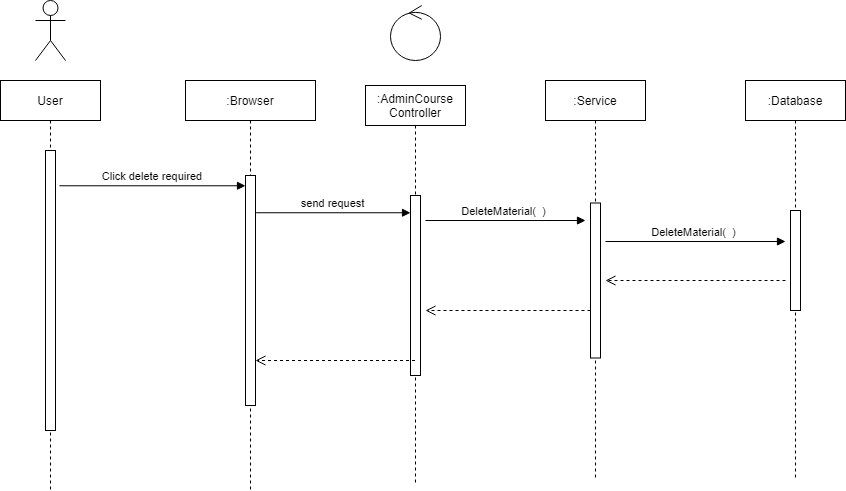
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DeleteMaterial | - This method will call API of deleteMaterial. |

#### 

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DeleteMaterial | - This method allow user with role Tutor, Admin, SuperAdmin to create course through CourseCreateRequest and return object. |
| 02 | findCourseByIdAndStatusIsTrue | - This method will find and return all course by id and status is true. |
| 03 | findMaterialByIdAndStatusIsTrue | - This method will find and return all Material of specific course by id and status is true. |
| 04 | findUserByAuthorizationToken | - Find a User by Authorization Token. |

#### c. Sequence Diagram(s)

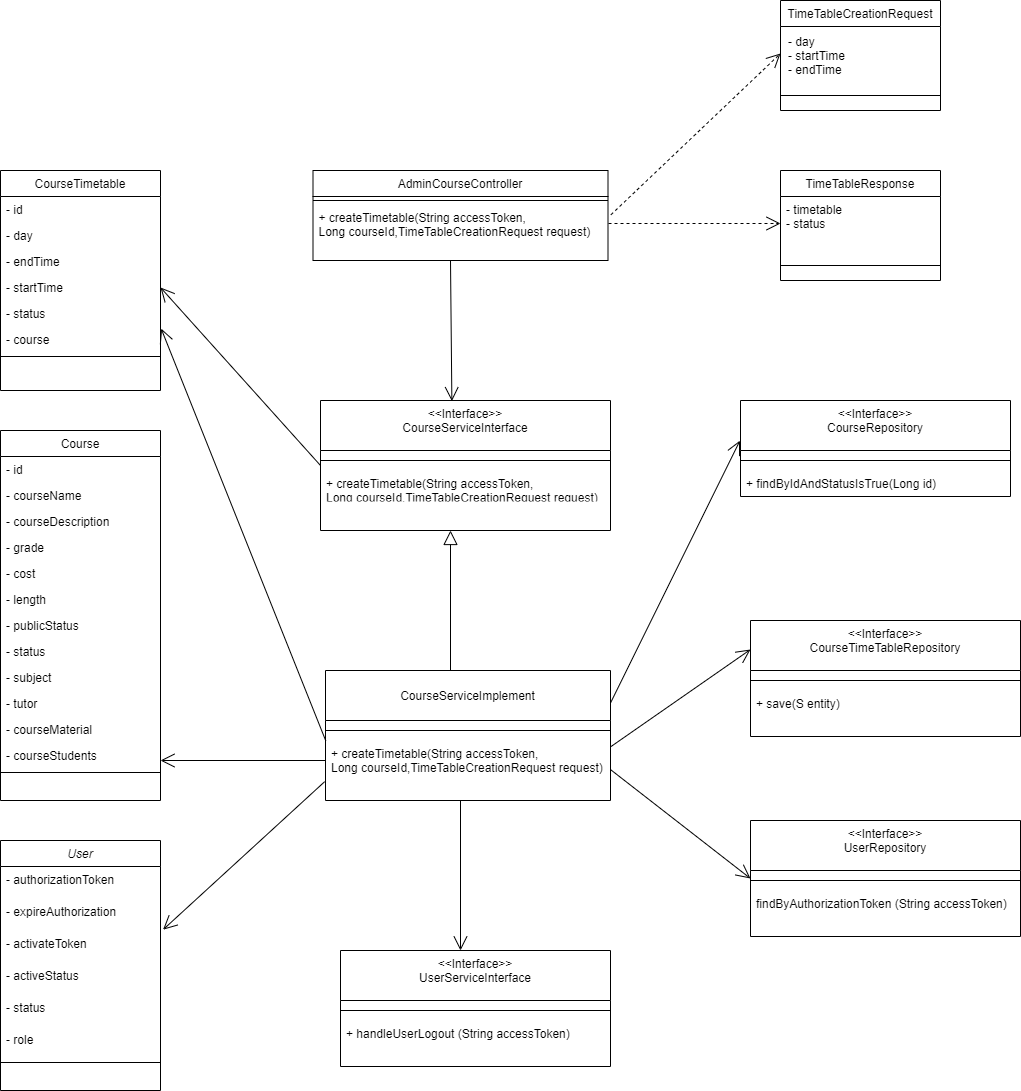


#### d. Database Queries

## 5. Course Timetable Management

### 5.1. Create timetable

#### a. Class Diagram



#### 

#### 

#### b. Class Specifications

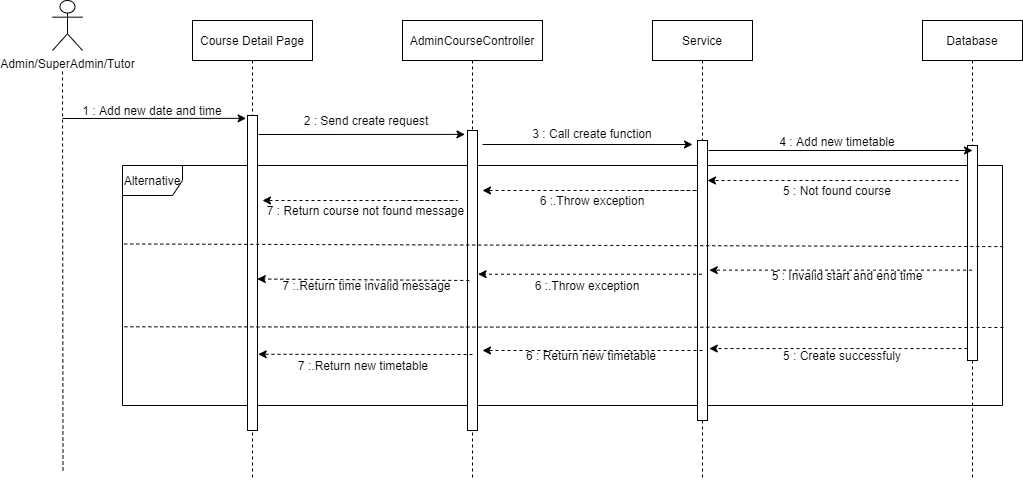
##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *createTimetable* | *Call the API of createTimetable* |

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findByIdAndStatusIsTrue* | *Find a course with id and status is True.* |
| *02* | *setStartTime* | *Set the start time of a course.* |
| *03* | *setEndTime* | *Set the end time of a course.* |
| *04* | *setDay* | *Set the day of a course.* |
| *05* | *isAfter* | *Check if the end time is before the start time or not.* |

#### c. Sequence Diagram(s)

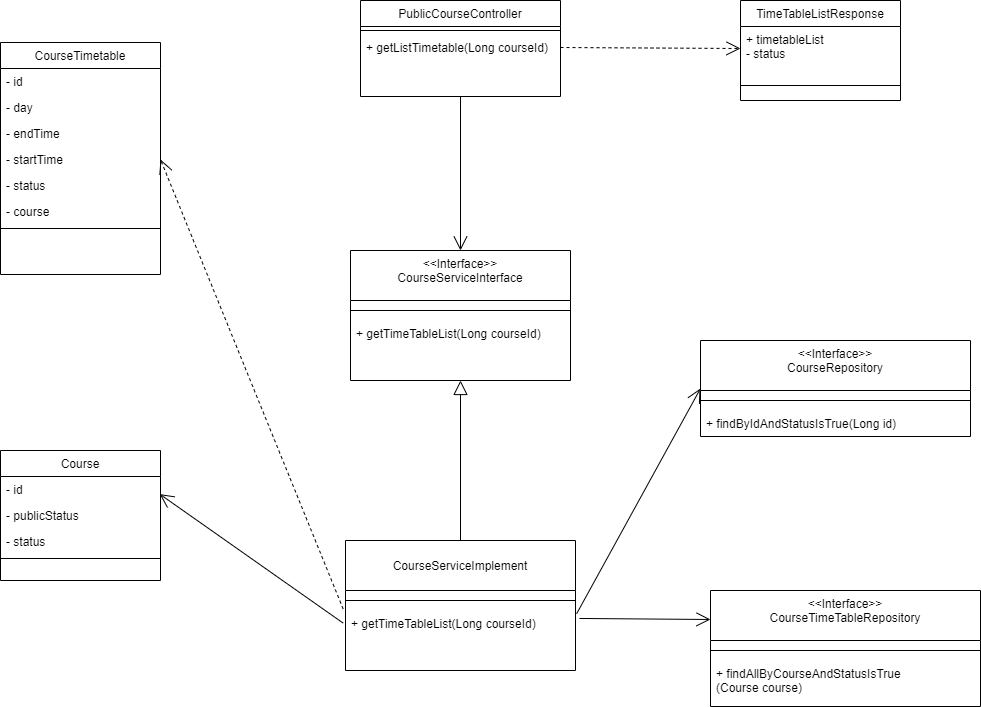


#### d. Database queries

### 

### 5.2. View course timetable

#### a. Class Diagram



#### b. Class Specifications

##### PublicCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getListTimetable* | *Call the API of getTimeTableList* |

***CourseService Class***

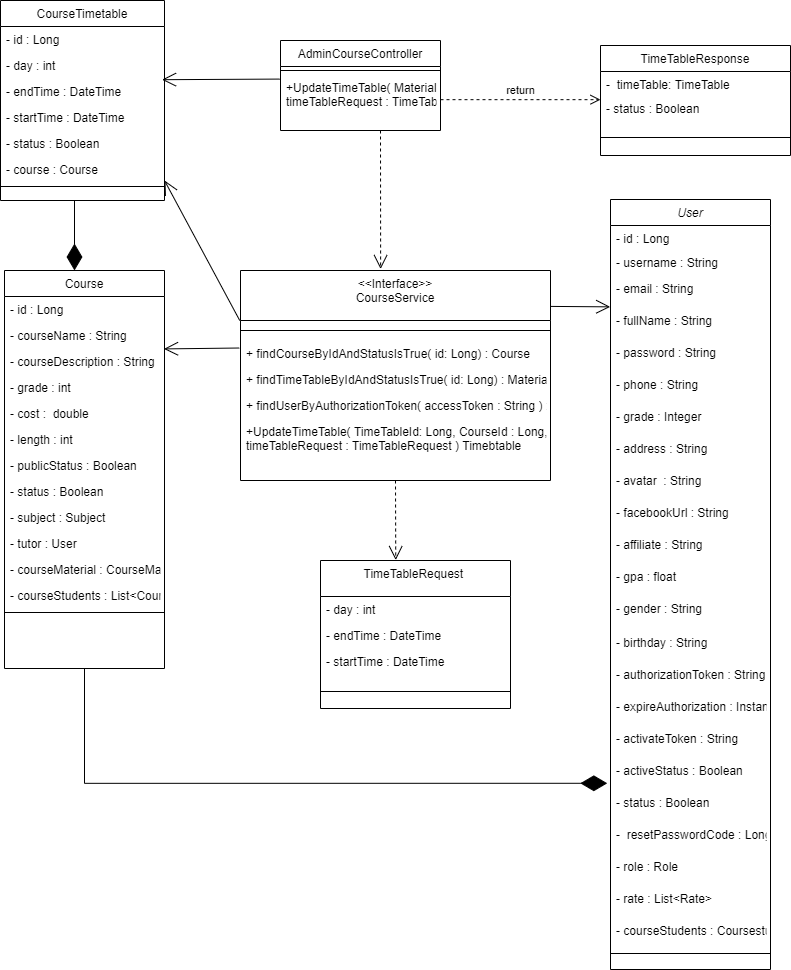
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findAllByCourseAndStatusIsTrue* | *Find all the timetables of a course.* |
| *02* | *findByIdAndStatusIsTrue* | *Find a course with id and status is true.* |
| *03* | *getTimeTableList* | *Return timetable list of a course.* |

#### c. Sequence Diagram(s)

#### d. Database queries

### 5.3. Update course timetable

#### a. Class Diagram



#### b. Class Specifications

##### CourseTimetable Class

* Model of specific timetable of one course.

##### User Class

* Model of User.

##### Course Class

* Model of Course .

##### TimeTableResponse Class

* Controller will return with this format response.

##### TimeTableRequest Class

* Controller will receive a Request with this format.

##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | UpdateTimetable | - This method will call the API of UpdateMaterial. |

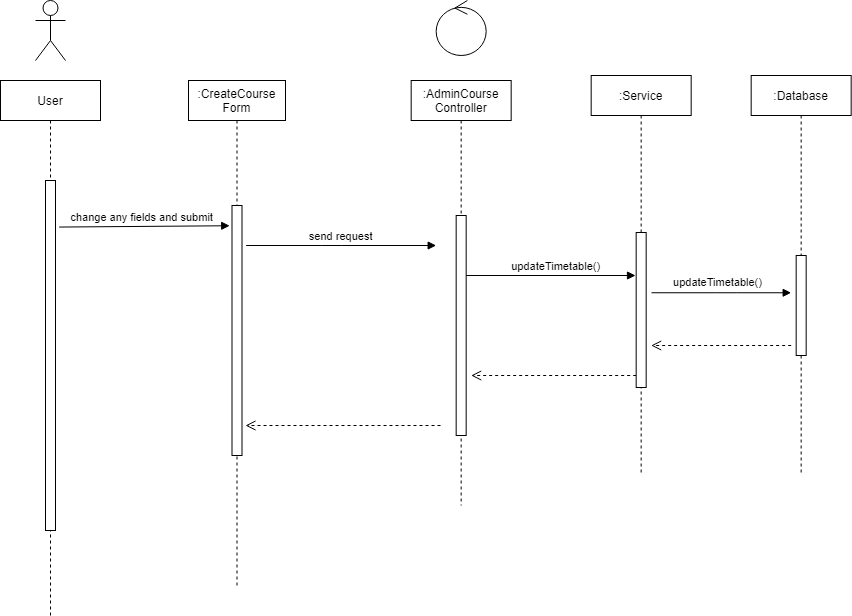
#### 

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | UpdateTimetable | - This method allows the Tutor to update the Timetable of a specific course through TimetableRequest and return object. |
| 02 | findCourseByIdAndStatusIsTrue | - This method will find and return all courses by id and status is true. |
| 03 | findTimetableByIdAndStatusIsTrue | - This method will find and return all timetables of a specific course by id and status. |
| 03 | findUserByAuthorizationToken | - Find a User by Authorization Token. |

### 

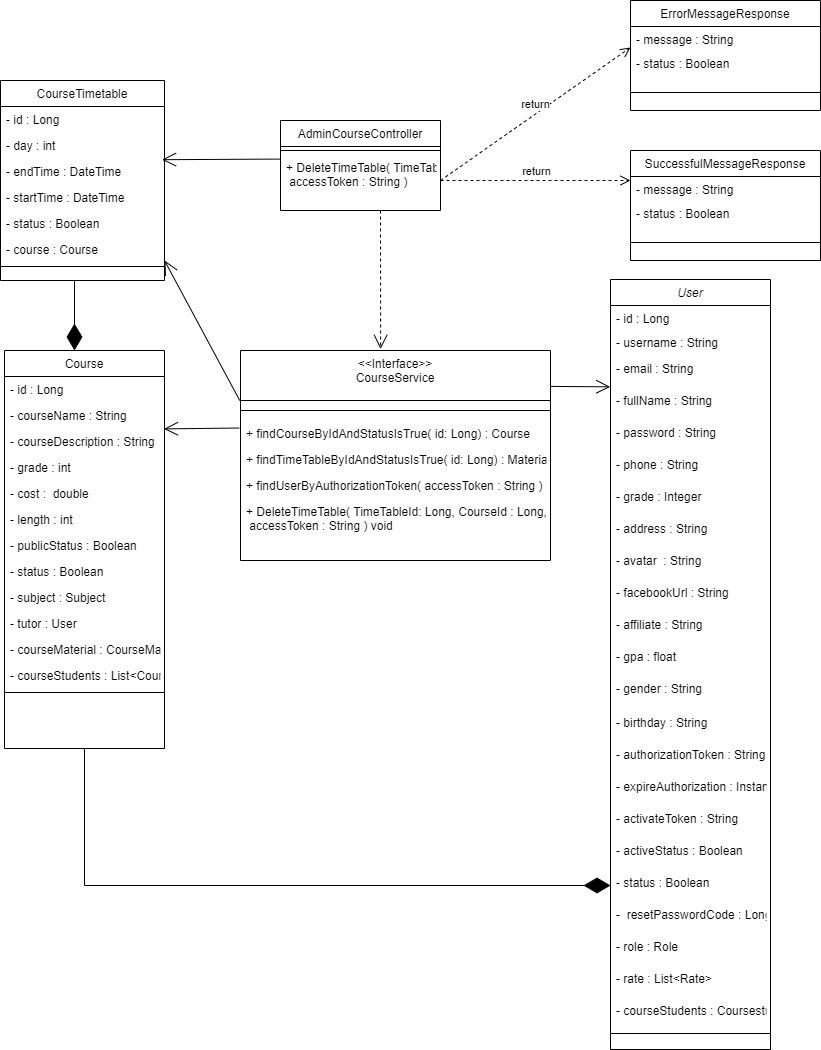
#### c. Sequence Diagram(s)



#### d. Database Queries

### 5.4. Delete course timetable

#### a. Class Diagram



#### b. Class Specifications

##### CourseTimetable Class

* Model of specific timetable of one course.

##### User Class

* Model of User.

##### Course Class

* Model of Course .

##### SuccessfulMessageResponse Class

* Controller will return with this format response if action success.

##### ErrorMessageResponse Class

* Controller will return with this format response if action error.

##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DeleteTimetable | - This method will call API of UpdateMaterial. |

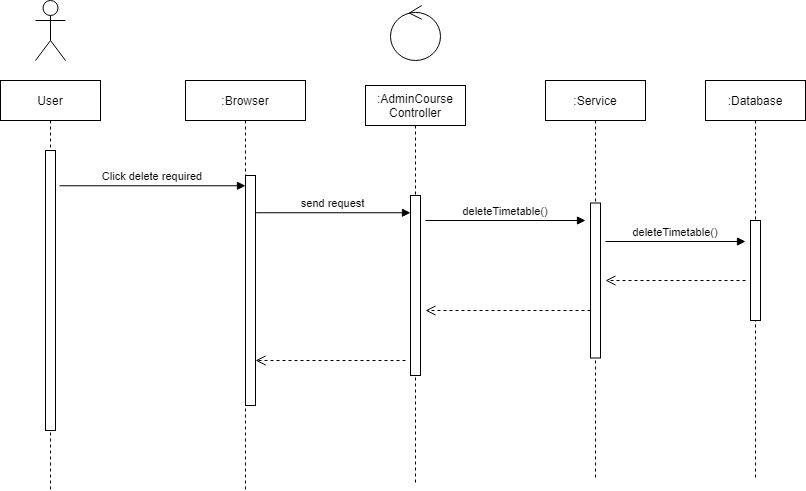
#### 

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DeleteTimetable | - This method allows the Tutor to delete the Timetable of a specific course . |
| 02 | findCourseByIdAndStatusIsTrue | - This method will find and return all courses by id and status is true. |
| 03 | findTimetableByIdAndStatusIsTrue | - This method will find and return all timetables of specific course by id and status is true. |
| 03 | findUserByAuthorizationToken | - Find a User by Authorization Token. |

### 

#### c. Sequence Diagram(s)

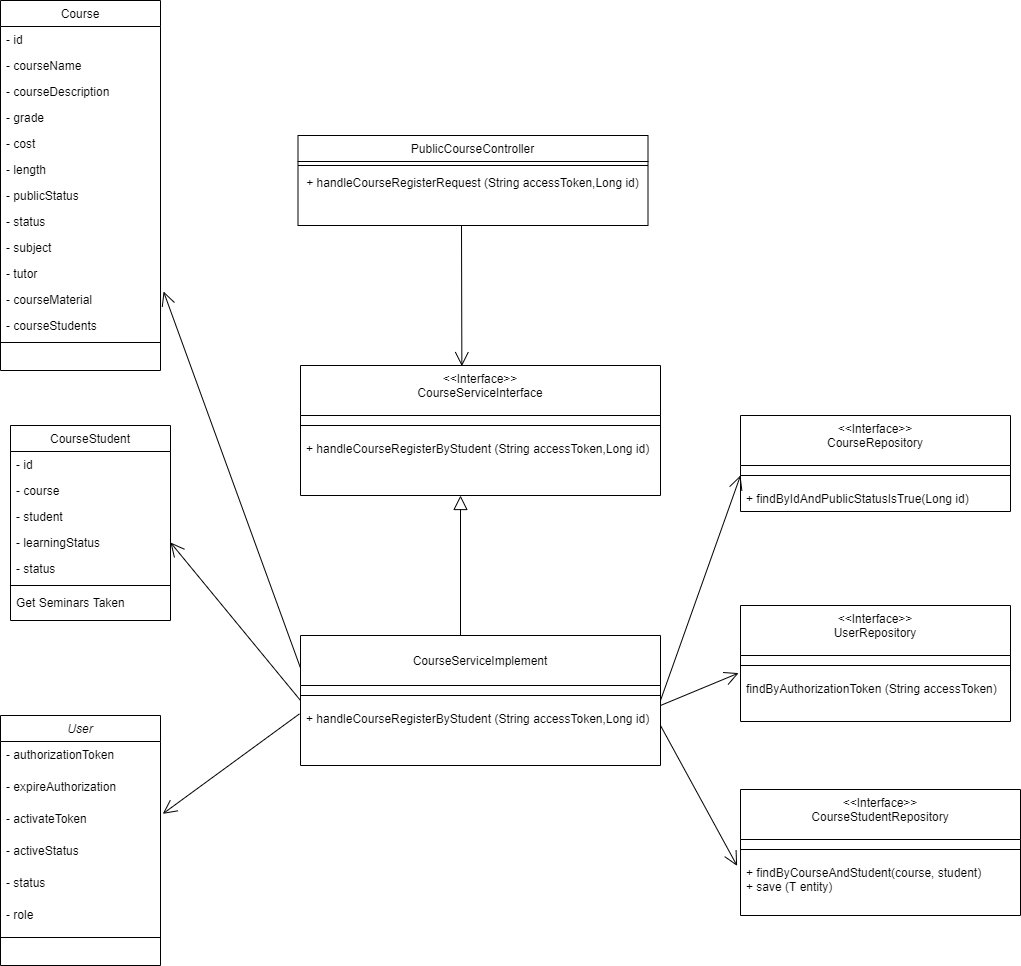


#### d. Database Queries

## 6. Course Registration

### 6.1.  Register course

#### a. Class Diagram

**

#### b. Class Specifications

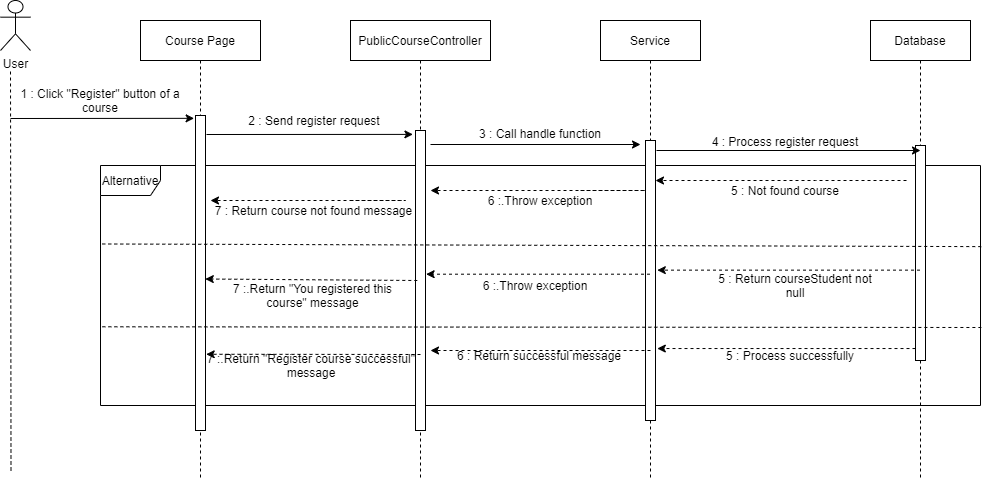
##### PublicCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *registerCourse* | *Call the API of handleCourseRegisterByStudent* |

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *handleCourseRegisterByStudent* | *Add students to a course pending for approval by the tutor or admin.* |
| *02* | *findByIdAndPublicStatusIsTrue* | *Find the course by Id and status is True.* |
| *03* | *findByCourseAndStudent* | *Check if the student has register this course or not.* |

#### c. Sequence Diagram(s)

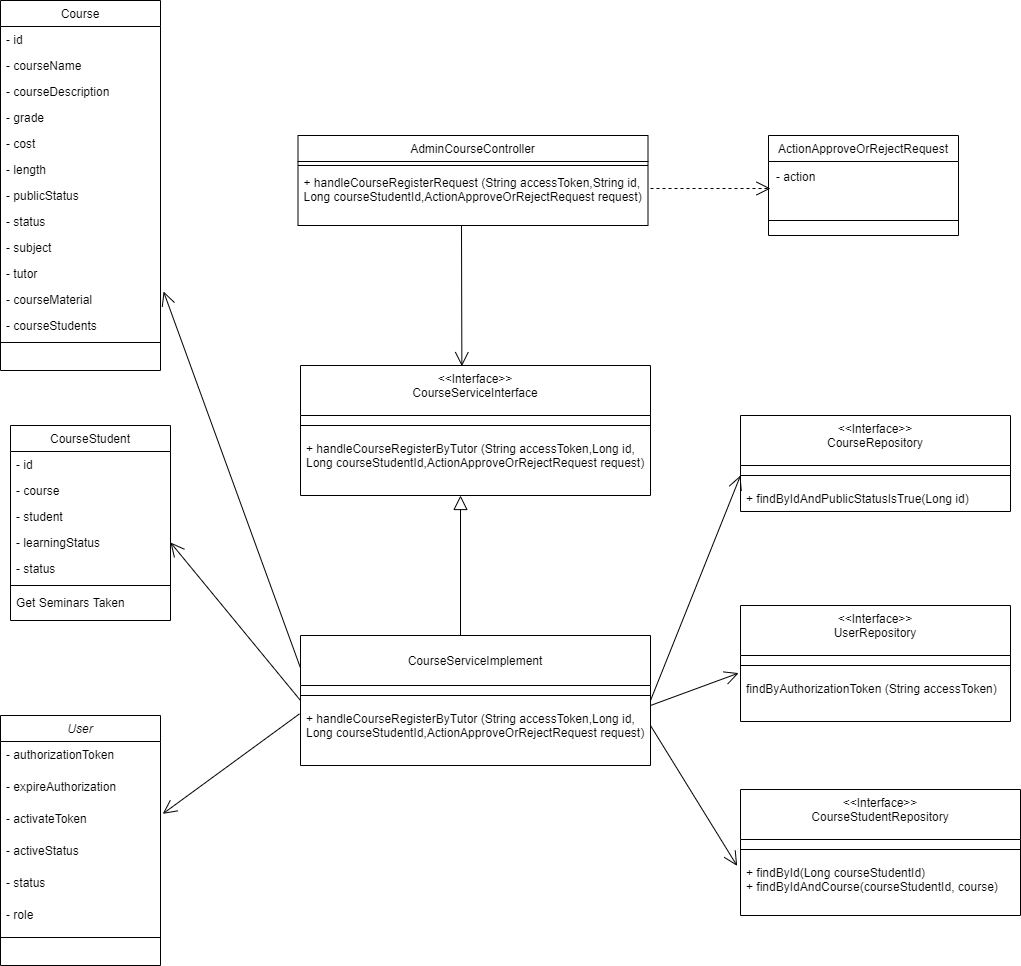
**

#### d. Database queries

### 

### 6.2.  Approve course registration

#### a. Class Diagram



#### b. Class Specifications

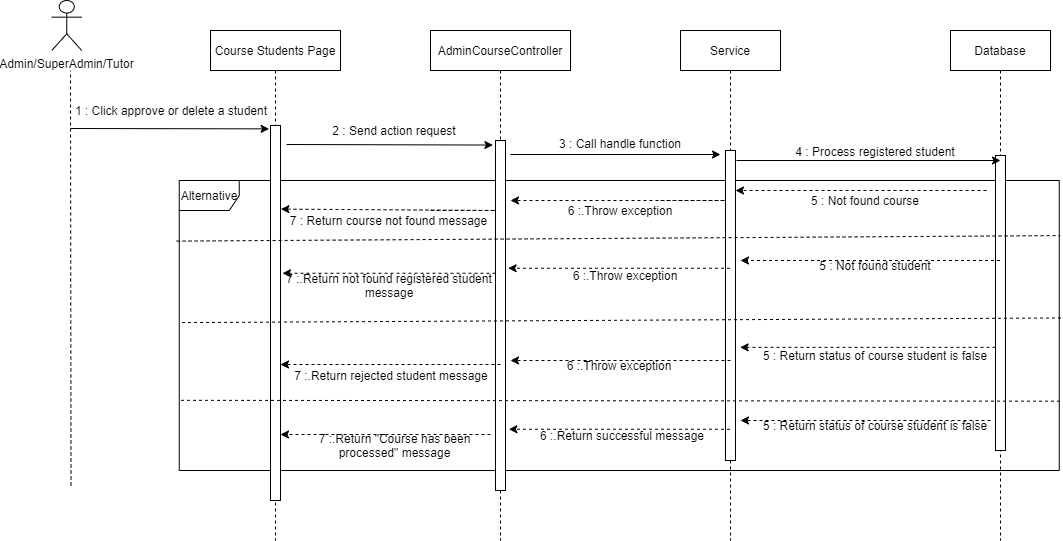
##### AdminCourseController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *handleCourseRegisterRequest* | *Call the API of handleCourseRegisterByTutor* |

##### CourseService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *handleCourseRegisterByTutor* | *Approve or reject the request of a student* |
| *02* | *findByIdAndPublicStatusIsTrue* | *Find a course with Id and status is True.* |
| *03* | *findByIdAndCourse* | *Check if a student has registered for this course or not.* |
| *04* | *findById* | *Check if the student rejects this course or not.* |
| *05* | *setLearningStatus* | *Set status to approve or reject student for this course.* |

#### c. Sequence Diagram(s)

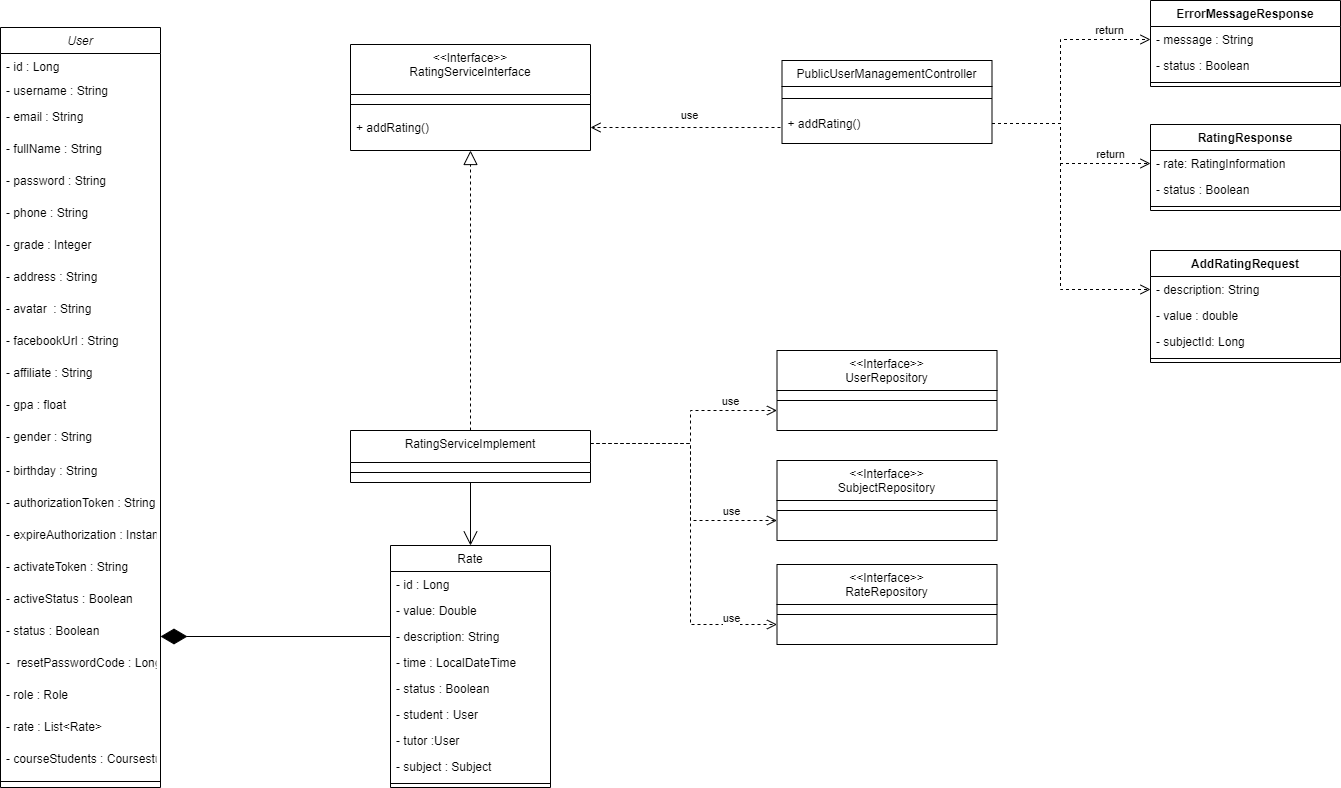


#### d. Database queries

## 7. Rate Management

### 7.1.  Add tutor rating

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of user

##### Rate Class

* Model of rate

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### RatingResponse Class

* Controller API will return this Rating Information format if added successful

##### AddRatingRequest Class

* Controller API will use this request to perform add rating action

##### PublicUserManagementController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *addRating* | *This method will call service to handle add rating request* |

##### RatingServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | addRating | *This method are using to perform add rating* |

##### RatingServiceDetailsImplements Class

* This class will implement the RatingServiceInterface Interface

##### SubjectRepository Interface

* This interface are used to query data on Subject Table

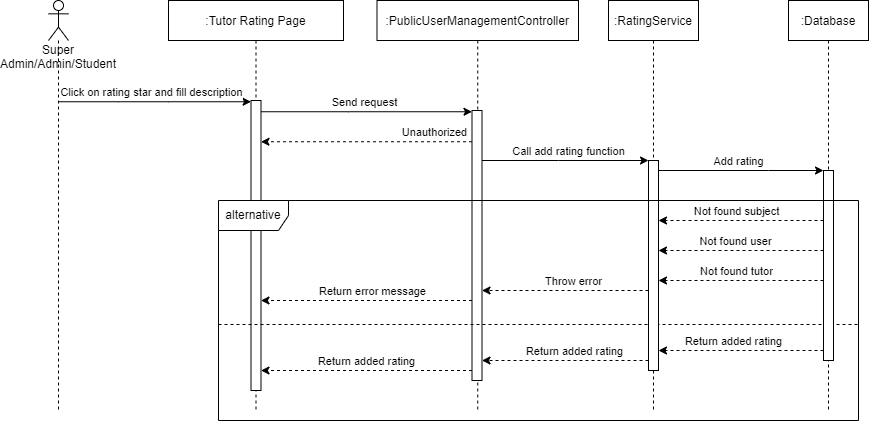
##### UserRepository Interface

* This interface are used to query data on User Table

##### RateRepository Interface

* This interface are used to query data on Rate Table

#### c. Sequence Diagram(s)

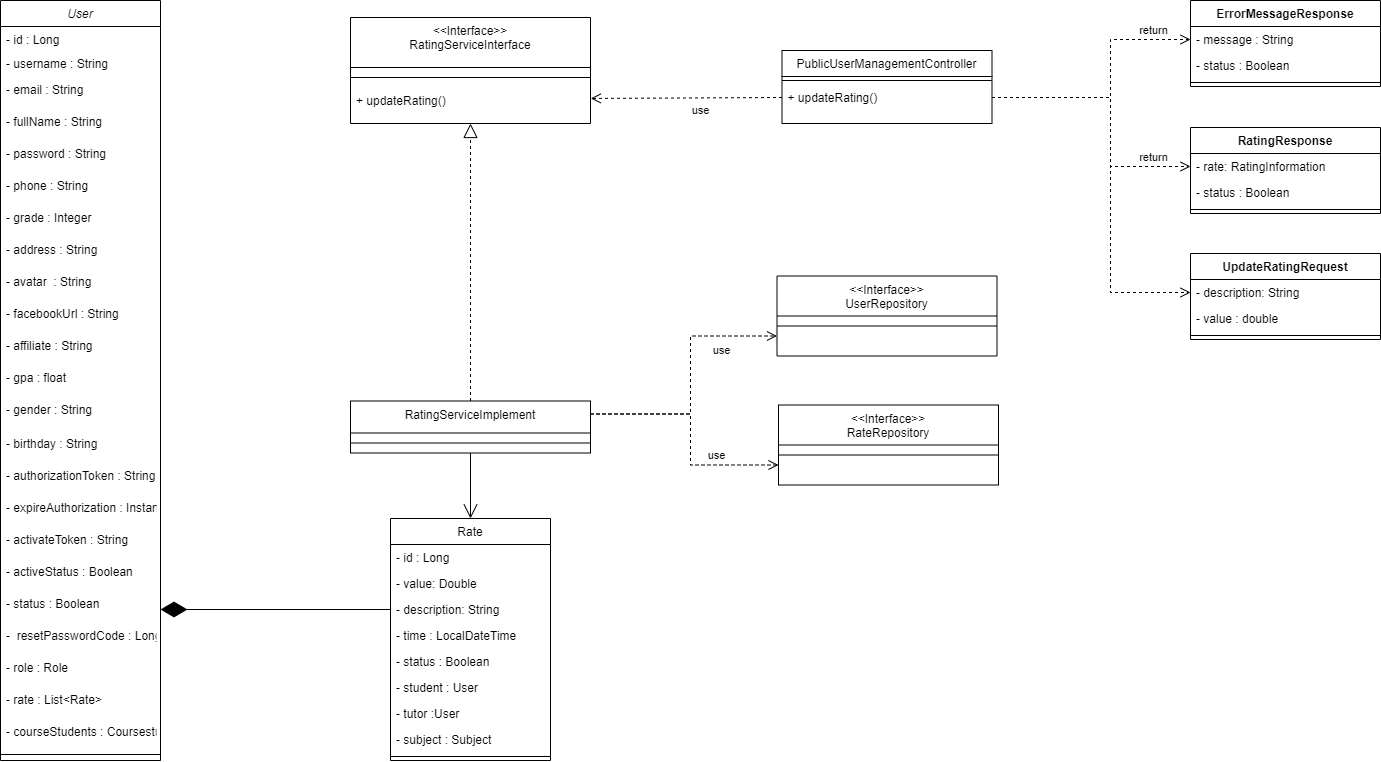


#### d. Database queries

#### 

### 7.2.  Update tutor rating

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of user

##### Rate Class

* Model of rate

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### RatingResponse Class

* Controller API will return this Rating Information format if update successful

##### UpdateRatingRequest Class

* Controller API will use this request to perform add rating action

##### PublicUserManagementController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateRating* | *This method will call service to handle update rating request* |

##### RatingServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | updateRating | *This method are using to perform update rating* |

##### RatingServiceDetailsImplements Class

* This class will implement the RatingServiceInterface Interface

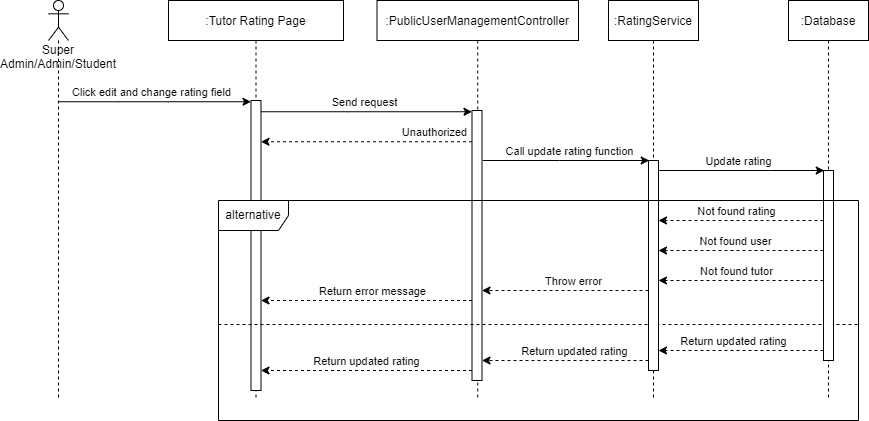
##### UserRepository Interface

* This interface are used to query data on User Table

##### RateRepository Interface

* This interface are used to query data on Rate Table

#### c. Sequence Diagram(s)

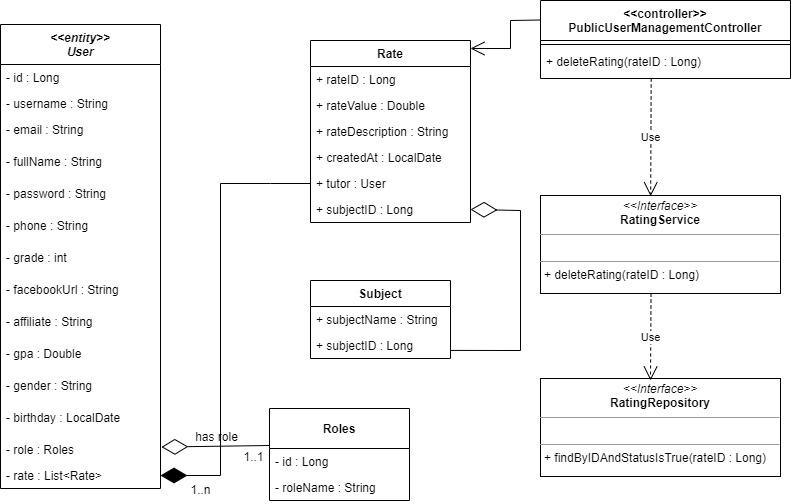


#### d. Database queries

#### 

### 7.3.  Delete tutor rating

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of User.

##### Rate Class

* Model of Rate.

##### Subject Class

* Model of Subject.

##### PublicUserManagementController Class

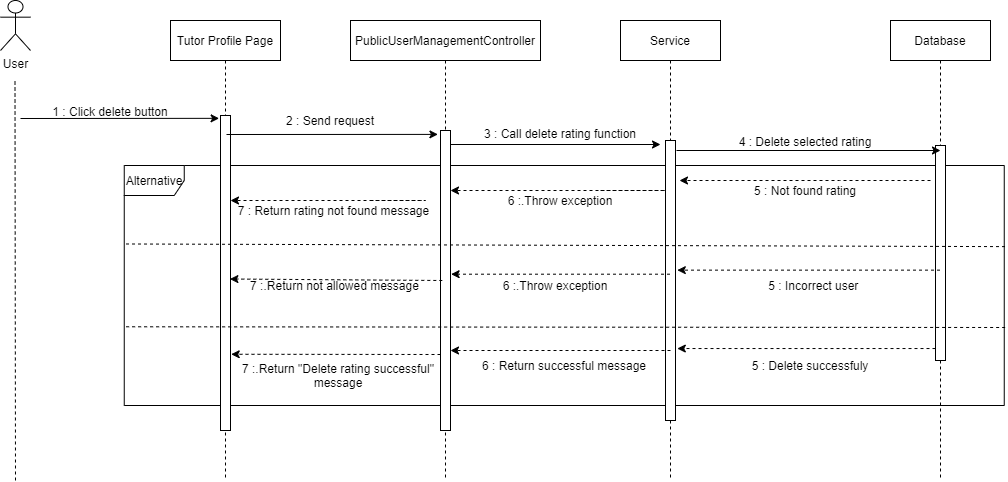
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deleteRating* | *Call the API of deleteRating* |

#### 

##### RatingService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deleteRating* | *This method deletes the selected rating.* |
| *03* | *findByIdAndStatusIsTrue* | *Find rating with Id and status is True.* |

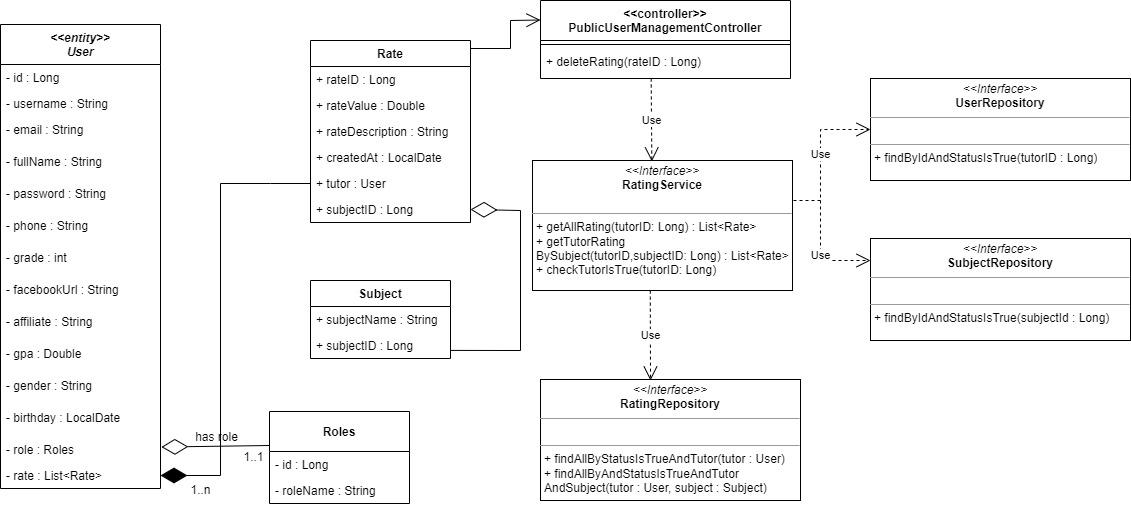
#### c. Sequence Diagram(s)

**

#### d. Database queries

### 7.4.  View tutor rating

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of User.

##### Rate Class

* Model of Rate.

##### Subject Class

* Model of Subject.

##### Roles Class

* Model of Role.

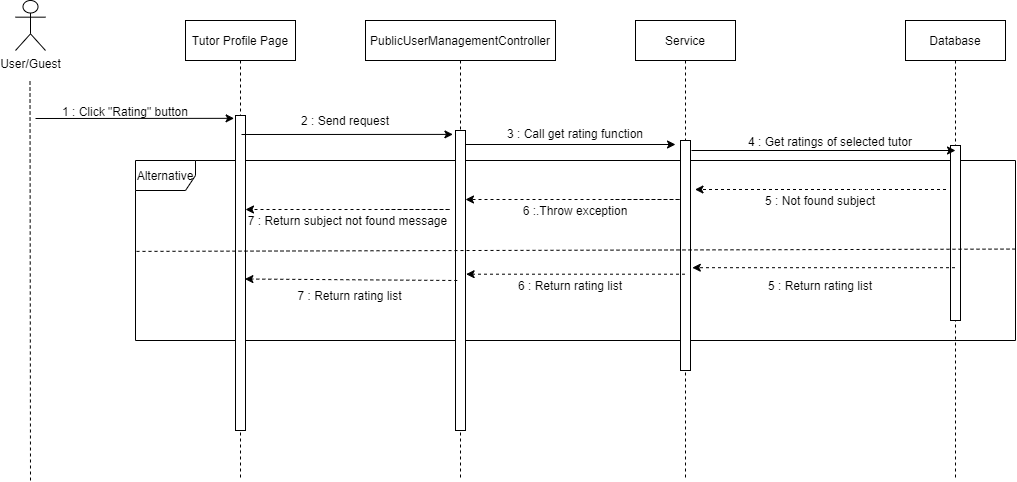
##### PublicUserManagementController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getTutorRatingBySubject* | *Call the API of getAllRating, getTutorRatingBySubject* |

##### RatingService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllRating* | *This function returns a list of ratings.* |
| *02* | *getTutorRatingBySubject* | *This function returns a list of ratings by subject.* |
| *03* | *findAllByStatusIsTrueAndTutorOrderByIdDesc* | *Find and sort all ratings.* |
| *04* | *findAllByStatusIsTrueAndTutorAndSubjectOrderByIdDesc* | *Find and sort all ratings by subject.* |

#### c. Sequence Diagram(s)

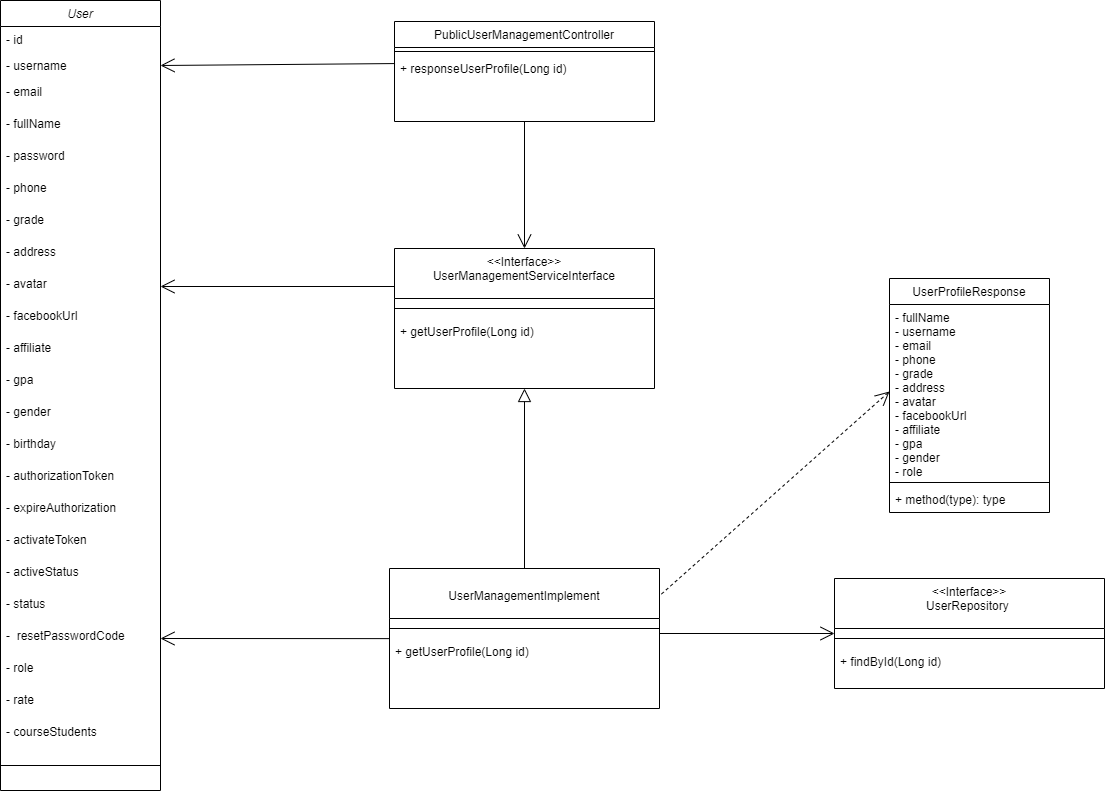
**

#### d. Database queries

## 8.  Profile management

### 8.1.  View profile

#### a. Class Diagram



#### b. Class Specifications

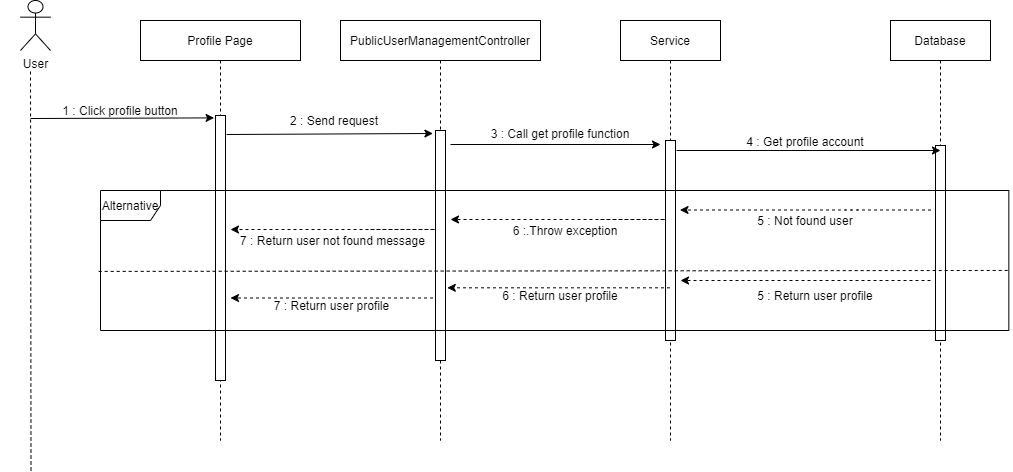
##### PublicUserManagementController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *responseUserProfile* | *Call the API of getUserProfile* |

##### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserProfile* | *This method returns the profile of the user.* |

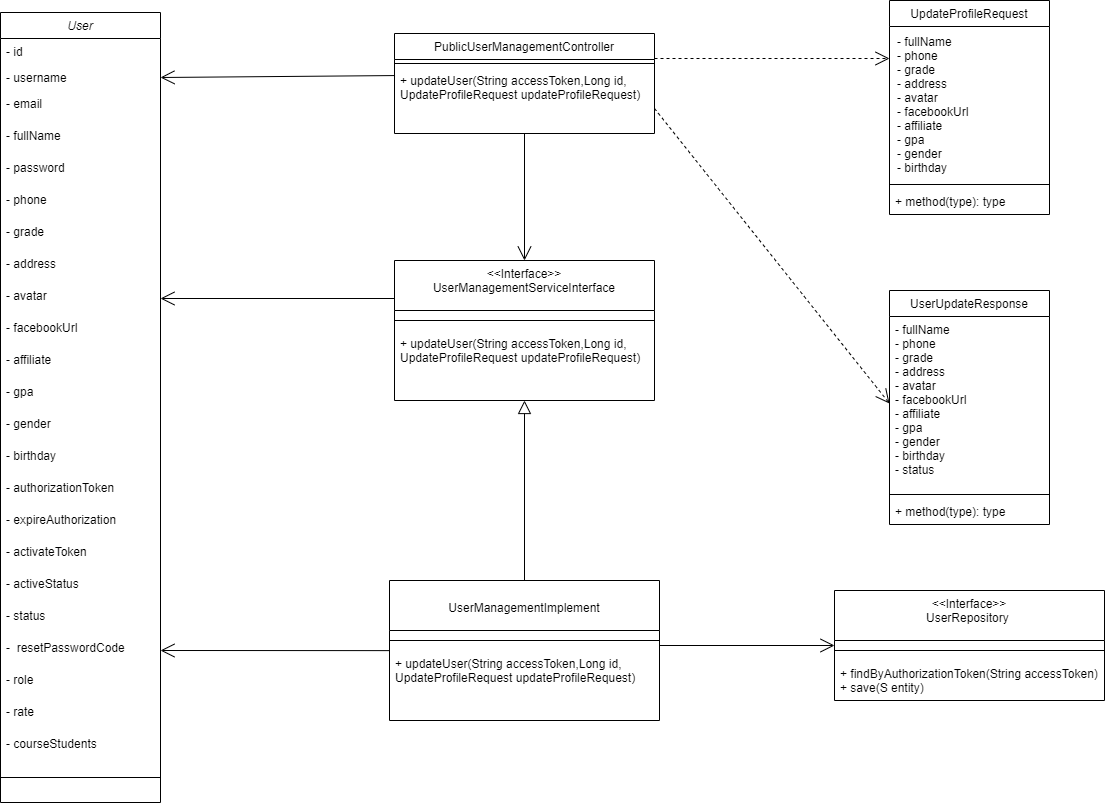
#### c. Sequence Diagram(s)



#### d. Database queries

### 8.2.  Edit profile

#### a. Class Diagram



#### b. Class Specifications

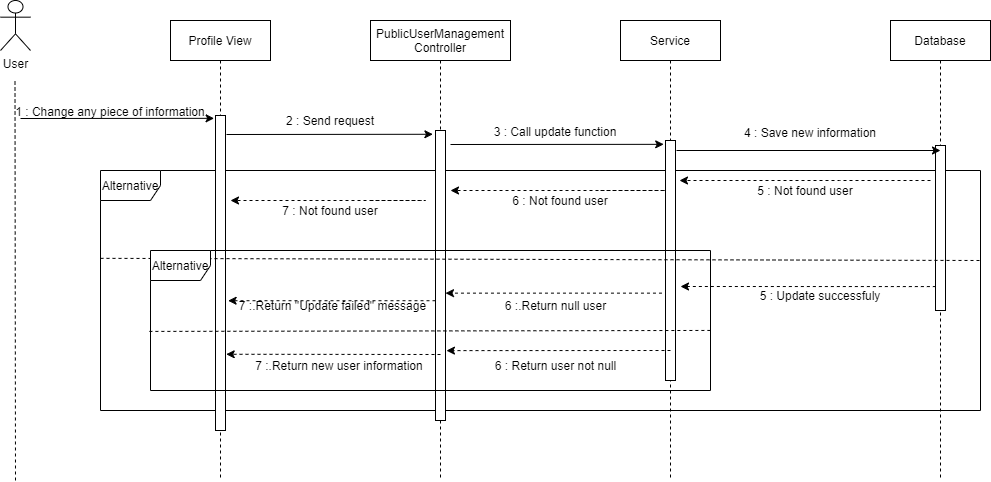
##### PublicUserManagementController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateUser* | *Call the API of updateUser* |

##### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateUser* | *This function changes the pieces of information of the user.* |
| *02* | *findByAuthorizationToken* | *Find the user by AuthorizationToken.* |

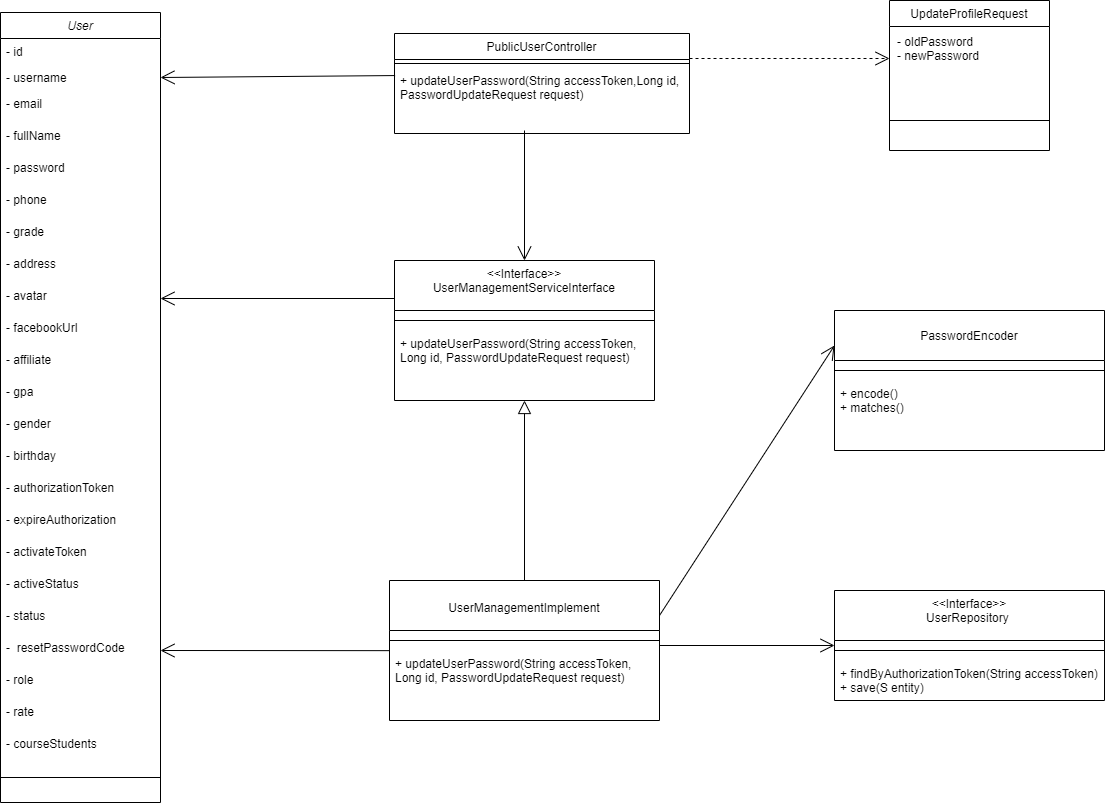
#### c. Sequence Diagram(s)



#### d. Database queries

### 8.3.  Update password

#### a. Class Diagram

**

#### b. Class Specifications

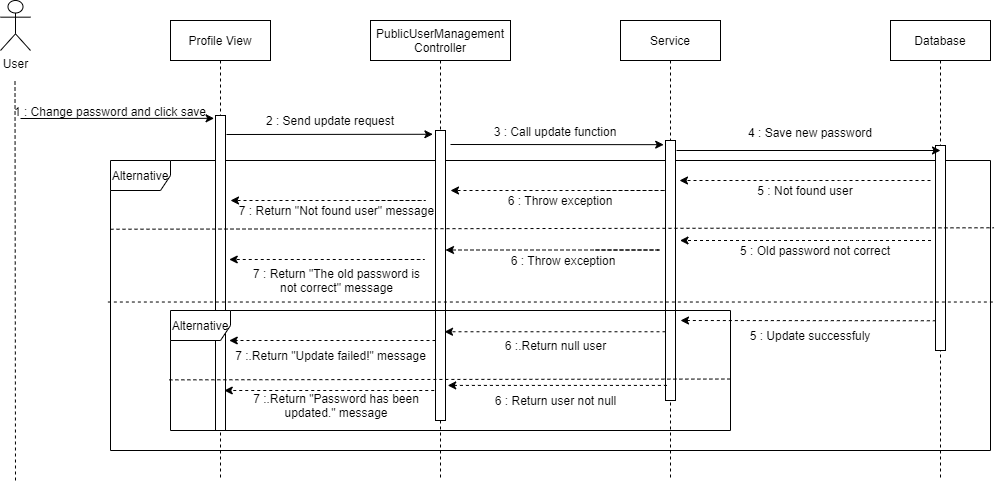
##### PublicUserManagementController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateUserPassword* | *Call the API of updateUserPassword* |

##### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateUserPassword* | *This method changes the password of the user.* |
| *02* | *findByAuthorizationToken* | *Find the user by AuthorizationToken.* |
| *03* | *matches* | *Compare the old password in the database to the old password user typed.* |
| *04* | *encode* | *Encode the new password by SHA-1 algorithm or greater hash.* |

#### c. Sequence Diagram(s)



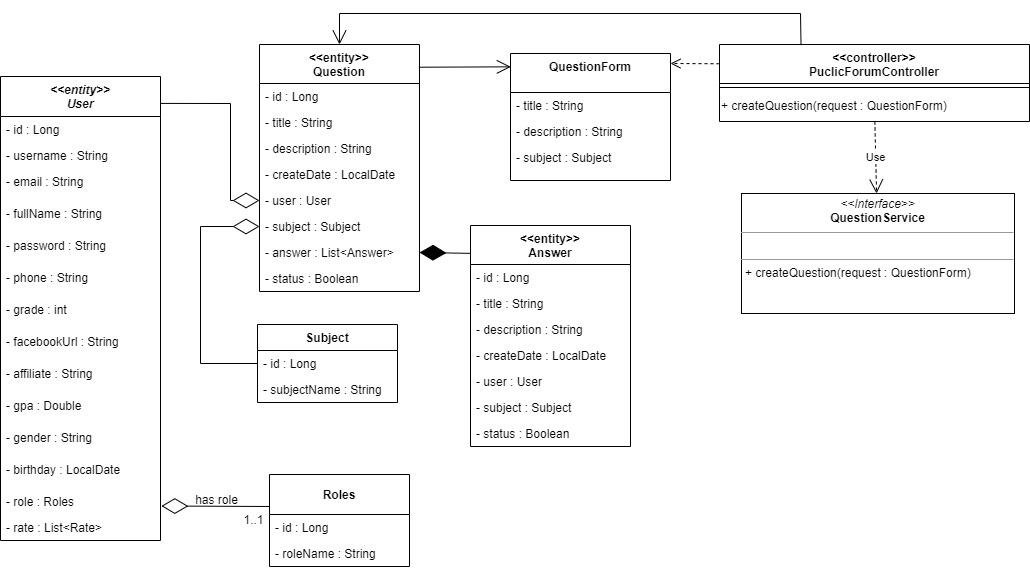
#### d. Database queries

### 9. Community

### 9.1. Question management

### 9.1.1.  Add question

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of User.

##### Question Class

* Model of Question.

##### QuestionRequest Class

* Controller will receive a request with this format.

##### QuestionResponse Class

* Controller will return with this format response.

##### Subject Class

* Model of Subject.

##### PublicForumController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | CreateQuestion | - This method will call the API of CreateQuestion for the forum service. |

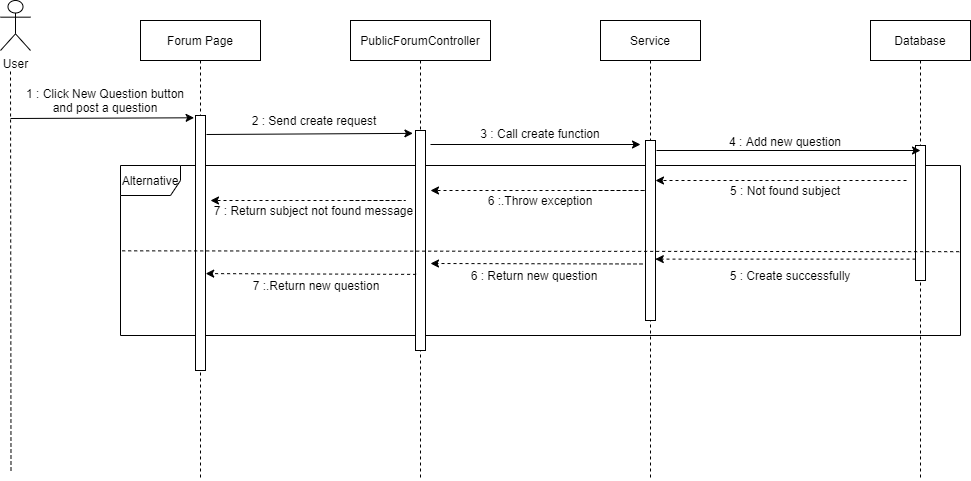
#### 

##### ForumService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | CreateQuestion | - This method allows users with roles Tutor, Admin, SuperAdmin to create questions through QuestionRequest and return objects. |
| 02 | findSubjectById | - Finding a Subject by Id. |
| 03 | findUserByAuthorizationToken | - Find a User by Authorization Token. |

### 

#### c. Sequence Diagram(s)

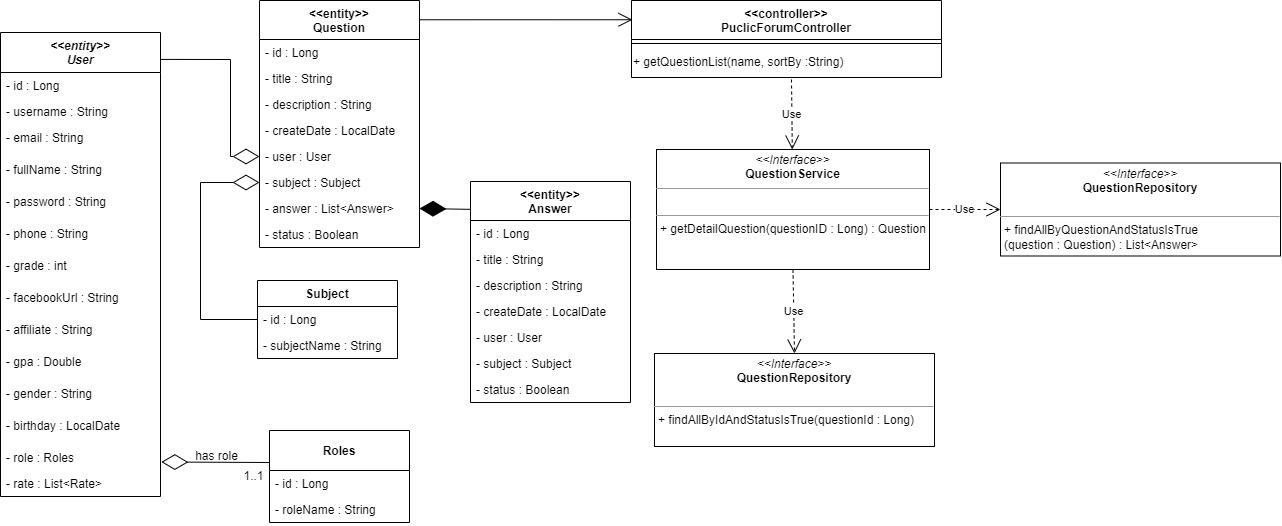
**

#### d. Database queries

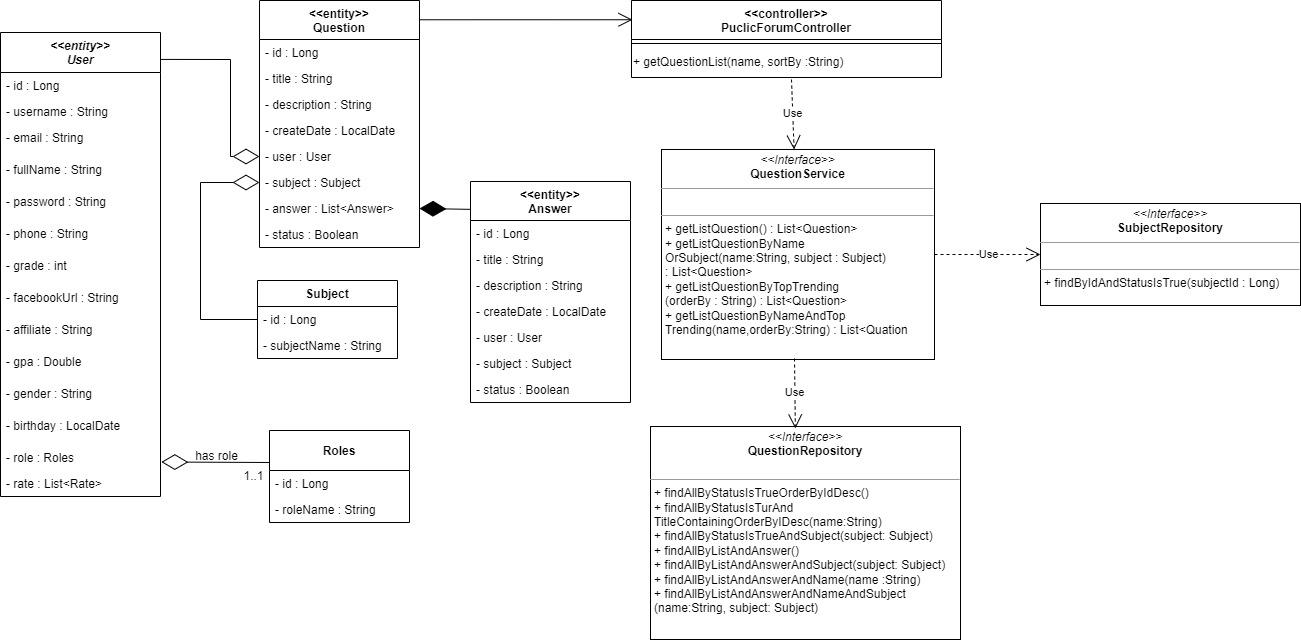
### 9.1.2.  View question

#### a. Class Diagram

* **View detail question**



* **View list questions**



#### b. Class Specifications

##### PublicForumController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getQuestionList | *Call API of getListQuestion, getListQuestionByNameOrSubject* |
| *02* | getDetailsQuestion | *Call API of getDetailsQuestion* |

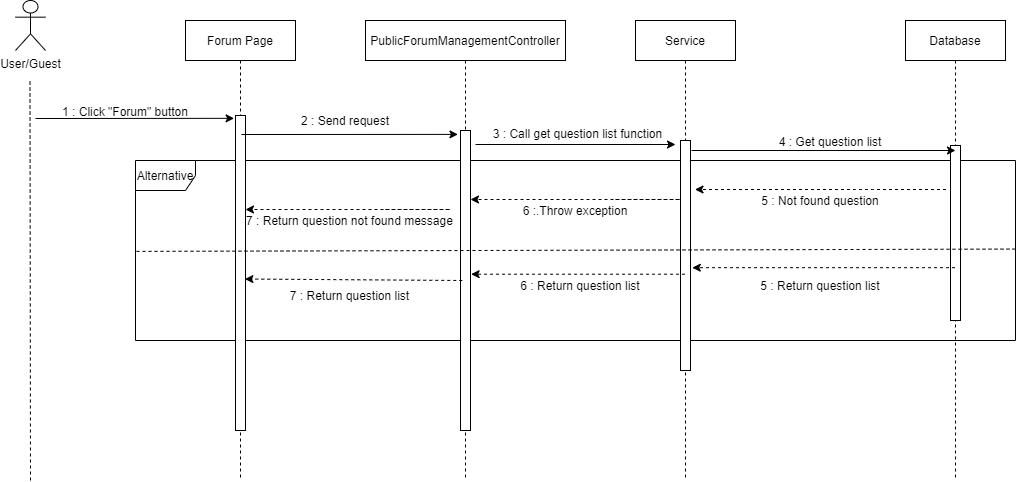
##### ForumService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getListQuestion* | *This method returns a list of all questions.* |
| *02* | *findAllByStatusIsTrueOrderByIdDesc* | *Find and sort all questions.* |
| *03* | *getListQuestionByNameOrSubject* | *This method returns a list of questions by title or subject.* |
| *04* | *findAllByStatusIsTrueAndTitleContainingOrderByIdDesc* | *Find and sort questions by title.* |
| *05* | *findAllByStatusIsTrueAndSubjectOrderByIdDesc* | *Find and sort questions by subject.* |
| *06* | *findAllByStatusIsTrueAndTitleContainingAndSubjectOrderByIdDesc* | *Find and sort questions by title and subject.* |
| *07* | *getDetailsQuestion* | *This method returns details of a question.* |
| *08* | *findByIdAndStatusIsTrue* | *Find question by Id and status is true.* |

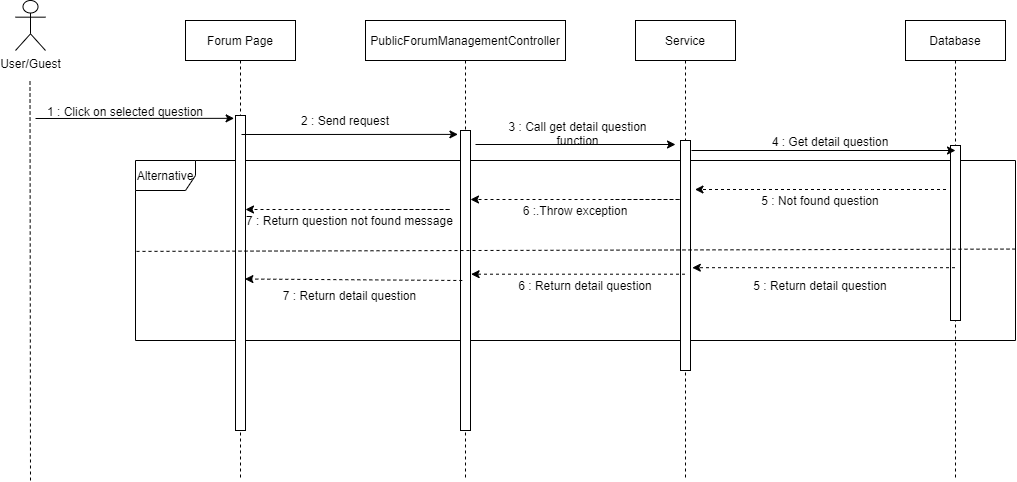
#### 

#### c. Sequence Diagram(s)

* ***view list question***

**

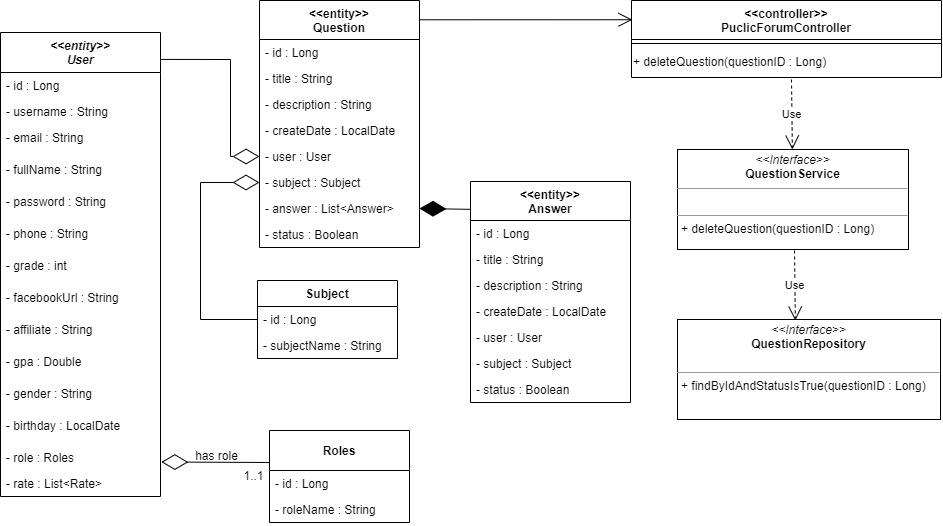
* ***View detail question***

**

#### d. Database queries

### 9.1.3. Delete question

#### a. Class Diagram



#### b. Class Specifications

##### User Class

* Model of User

##### Question Class

* Model of Question

##### Answer Class

* Model of Answer

##### Roles Class

* Model of Roles

##### Subject Class

* Model of Subject

##### PublicForumController Class

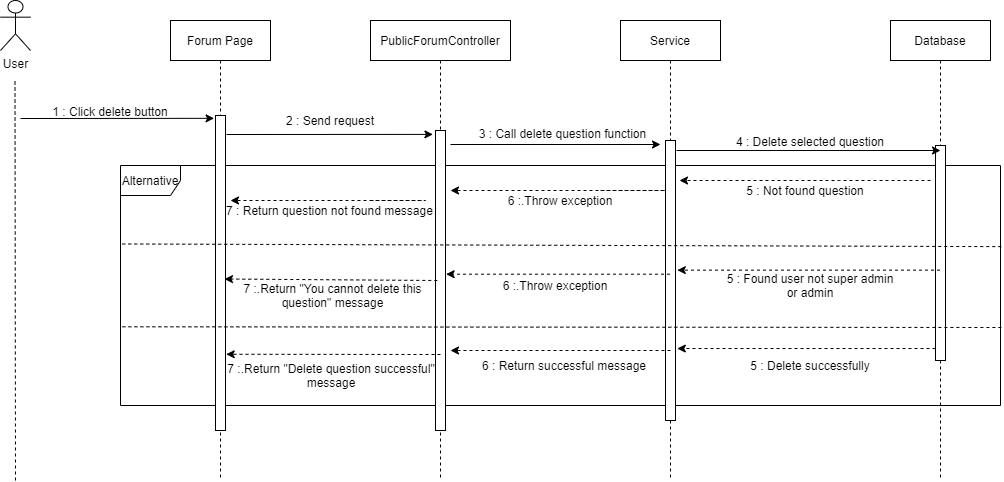
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deleteQuestion* | *Call API of deleteQuestion* |

##### 

##### QuestionService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deleteQuestion* | *This method deletes a question.* |
| *02* | *findByIdAndStatusIsTrue* | *Find the question by Id and status is True.* |
| *03* | *setStatus* | *Change the status of the question to false (soft delete).* |

#### c. Sequence Diagram(s)

**

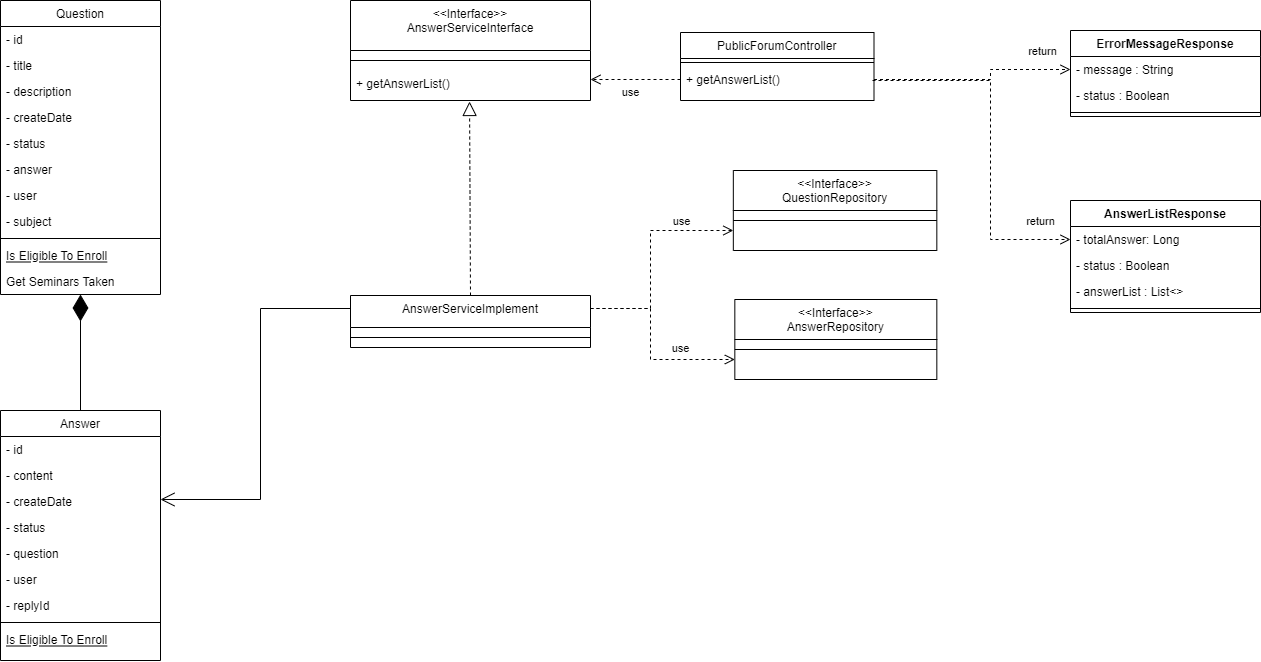
#### d. Database queries

### 

### 9.2. Answer management

### 9.2.1. View answer

#### a. Class Diagram



#### b. Class Specifications

##### Question Class

* Model of question

##### Answer Class

* Model of Answer

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### AnswerListResponse Class

* Controller API will return this Rating Information list format if action perform successful

##### PublicForumController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAnswerList* | *This method will call service to handle view answer request* |

##### AnswerServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAnswerList* | *This method are using to perform get answer list* |

##### AnswerServiceDetailsImplements Class

* This class will implement the AnswerServiceInterface Interface

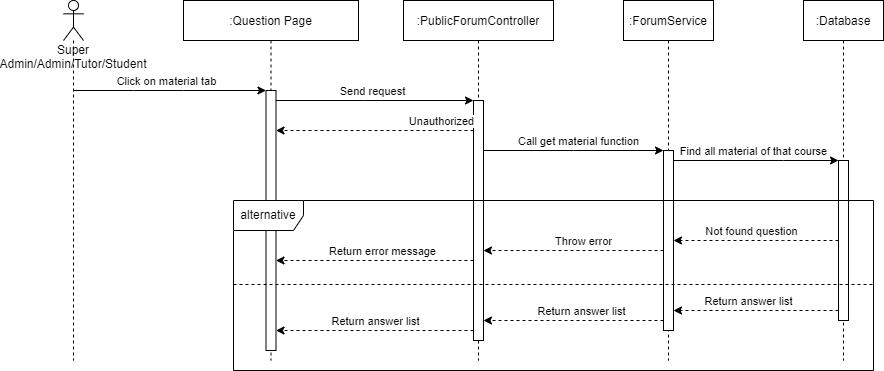
##### QuestionRepository Interface

* This interface are used to query data on Question Table

##### AnswerRepository Interface

* This interface are used to query data on Answer Table

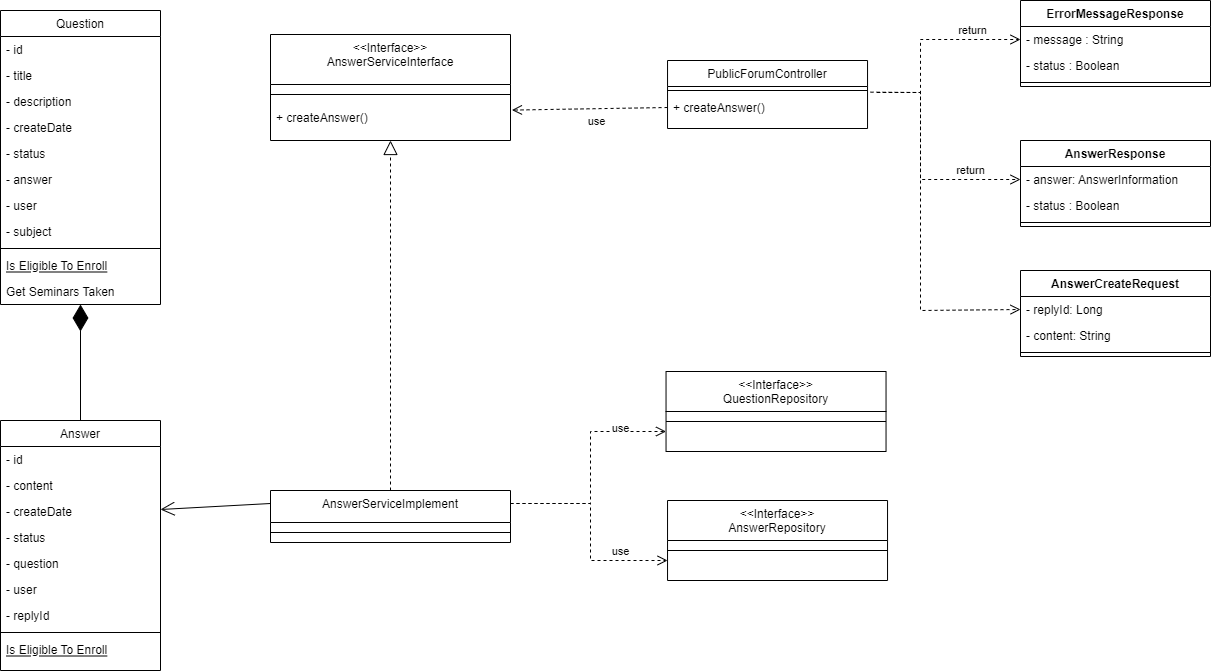
#### c. Sequence Diagram(s)



#### d. Database queries

### 9.2.2. Add answer

#### a. Class Diagram



#### b. Class Specifications

##### Question Class

* Model of question

##### Answer Class

* Model of Answer

##### ErrorMessageResponse Class

* Controller API will return this format if error occur

##### AnswerResponse Class

* Controller API will return this Rating Information format if update successful

##### AnswerCreateRequest Class

* Controller API will use this request to perform add question action

##### PublicForumController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *createAnswer* | *This method will call service to handle create answer request* |

##### AnswerServiceInterface Interface

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *createAnswer* | *This method are using to perform create answer* |

##### AnswerServiceDetailsImplements Class

* This class will implement the AnswerServiceInterface Interface

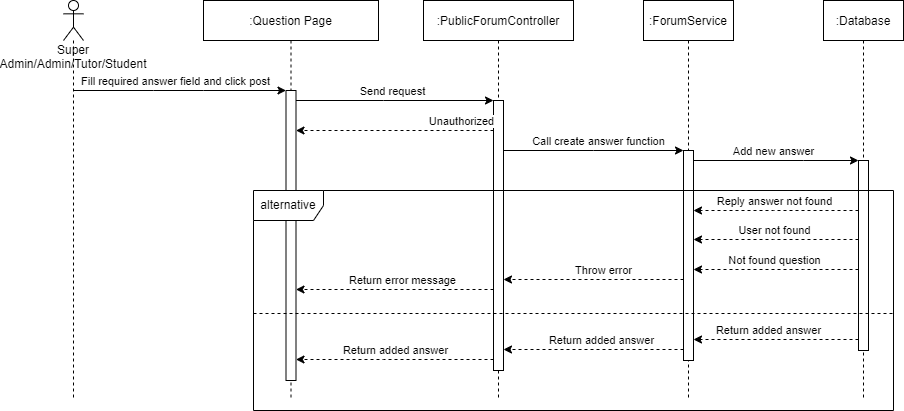
##### QuestionRepository Interface

* This interface are used to query data on Question Table

##### AnswerRepository Interface

* This interface are used to query data on Answer Table

#### c. Sequence Diagram(s)

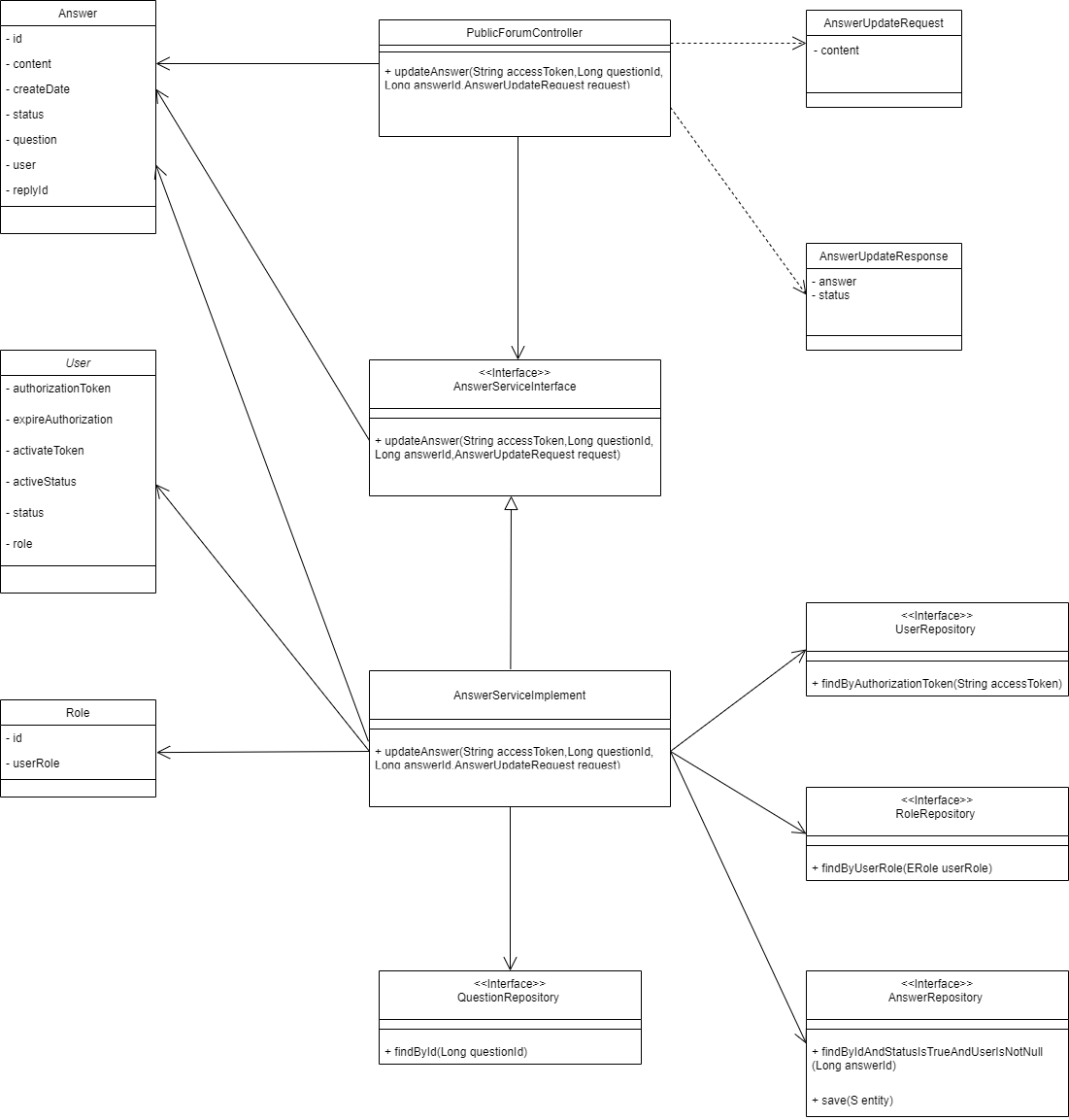


#### d. Database queries

#### 

### 9.2.3. Edit answer

#### a. Class Diagram



#### b. Class Specifications

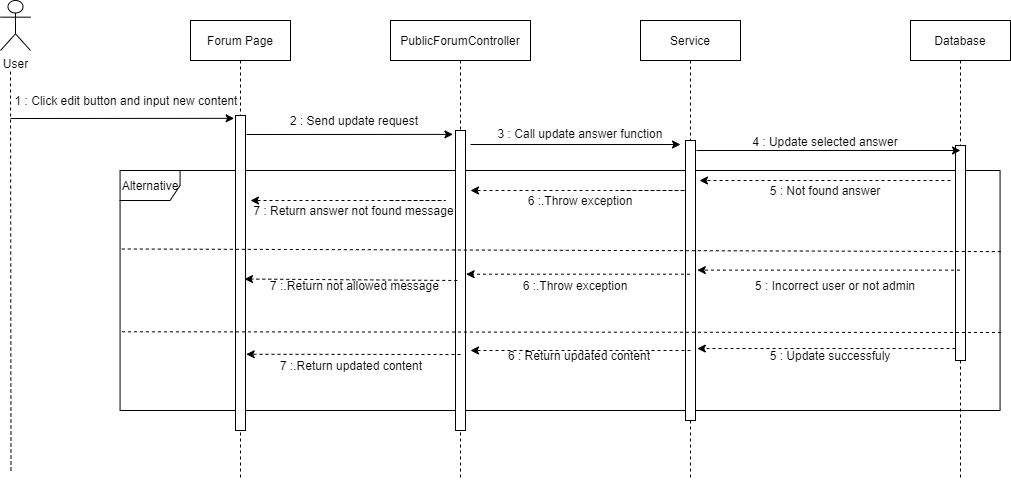
##### PublicForumController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateAnswer* | *Call the API of updateAnswer* |

##### ForumService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateAnswer* | *This function changes the content of the answer* |
| *02* | *findByIdAndStatusIsTrueAndUserIsNotNull* | *Find the answer by Id and status is true and the user is not null.* |
| *03* | *findByUserRole* | *Find the role of the user.* |
| *04* | *setContent* | *Change the content of the old answer with the new content.* |

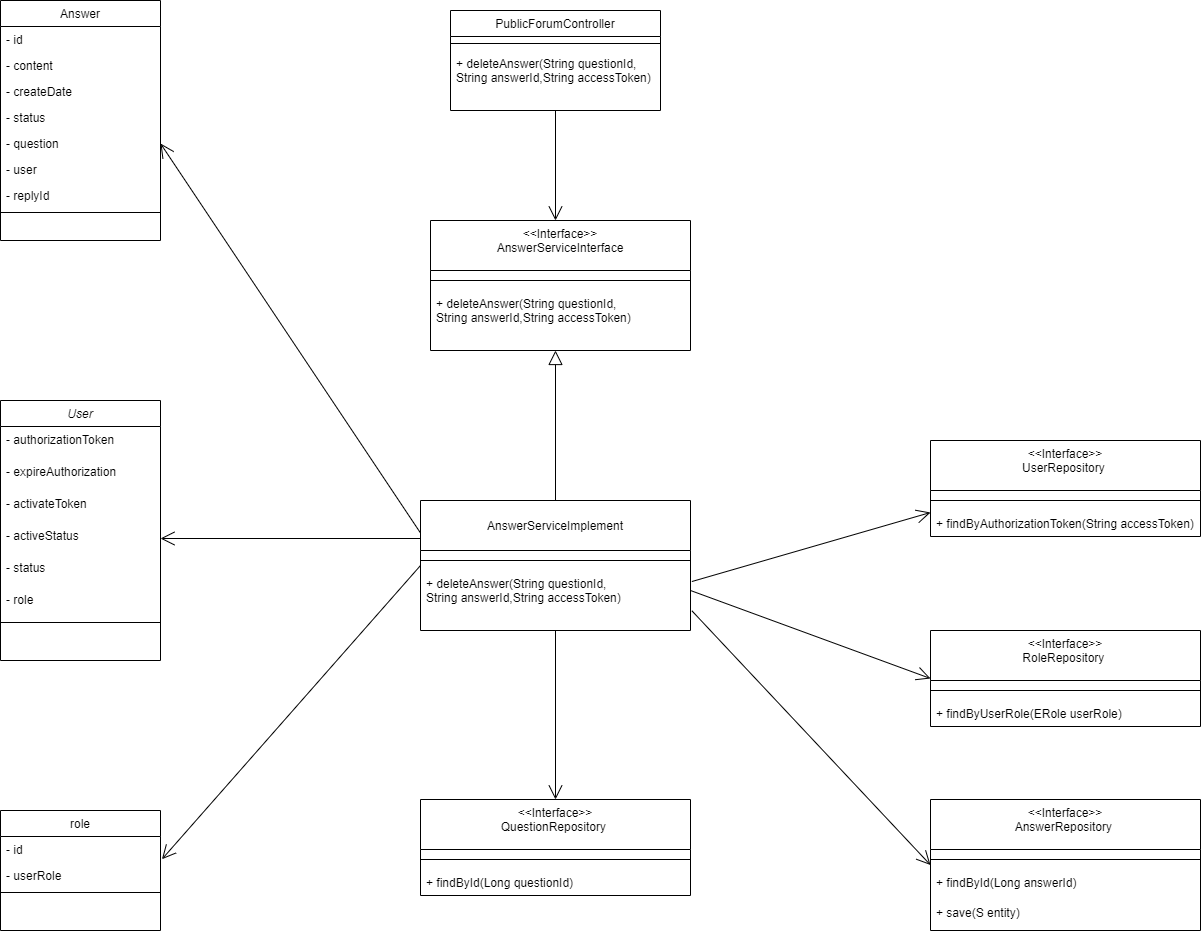
#### c. Sequence Diagram(s)



#### d. Database queries

### 9.2.4. Delete answer

#### a. Class Diagram



#### b. Class Specifications

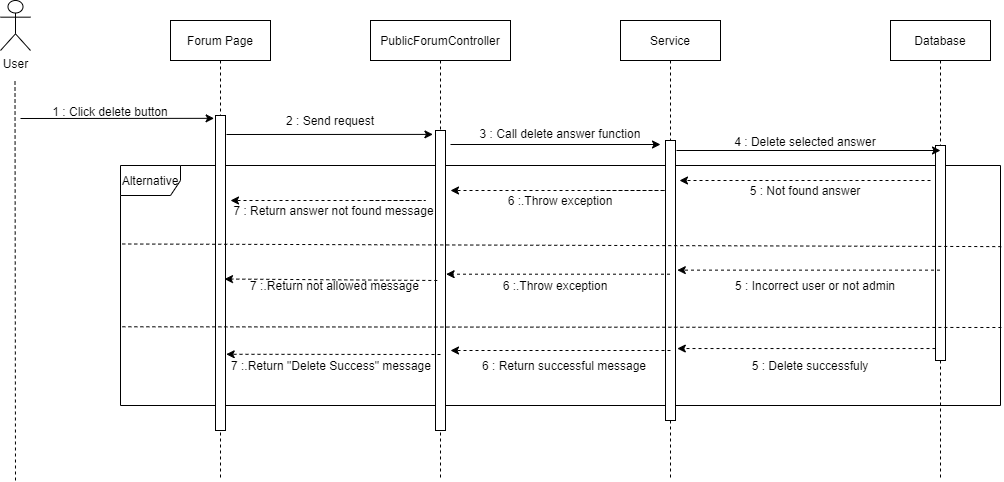
##### PublicForumController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deleteAnswer* | *Call the API of deleteAnswer* |

##### ForumService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deleteAnswer* | *This function deletes the selected answer.* |
| *02* | *findById* | *Find the answer by Id.* |
| *03* | *setStatus* | *Change the status of the answer to false (soft delete).* |

#### c. Sequence Diagram(s)



#### d. Database queries

# III. Database Tables

## 1. users\_account

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | activate\_token | varchar | 255 |  | No |  |  |
| 3 | active\_status | bit | 1 |  | No |  | this status will true if user active email verification |
| 4 | address | nvarchar | 500 |  | No |  |  |
| 5 | affiliate | varchar | 255 |  | No |  |  |
| 6 | authorization\_token | varchar | 255 |  | No |  |  |
| 7 | avatar | varchar | 255 |  | No |  |  |
| 8 | birthday | varchar | 255 |  | No |  |  |
| 9 | email | varchar | 255 |  | No |  |  |
| 10 | expire\_authorization | datetime2 | 7 |  | No |  |  |
| 11 | facebook\_url | varchar | 255 |  | No |  |  |
| 12 | full\_name | nvarchar | 100 |  | No |  |  |
| 13 | gender | varchar | 255 |  | No |  |  |
| 14 | gpa | float | 7 |  | No |  |  |
| 15 | grade | int | 2 |  | No |  |  |
| 16 | password | varchar | 255 |  | No |  |  |
| 17 | phone | varchar | 255 |  | No |  |  |
| 18 | reset\_password\_code | bigint | 8 |  | No |  |  |
| 19 | status | bit | 1 |  | No |  | Not Deleted: true.  Deleted: false. |
| 20 | username | varchar | 255 | Yes | Yes |  |  |

## 2. user\_roles

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | user\_id | bigint | 8 | Yes | Yes | PK,FK |  |
| 2 | role\_id | bigint | 8 |  | Yes | PK,FK |  |

## 

## 3. roles

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | user\_role | varchar | 255 |  | No |  |  |

## 

## 4. rate

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | description | nvarchar | 4000 |  | No |  |  |
| 3 | status | bit | 1 |  | No |  | Not Deleted: true.  Deleted: false. |
| 4 | time | datetime2 | 7 |  | No |  |  |
| 5 | value | float | 7 |  | No |  |  |
| 6 | student\_id | bigint | 8 |  | No | FK |  |
| 7 | subject\_id | bigint | 8 |  | No | FK |  |
| 8 | tutor\_id | bigint | 8 |  | No | FK |  |

## 5. subject

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | subject\_name | varchar | 255 |  | No |  |  |

## 

## 6. question

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | description | nvarchar | 4000 |  | No |  |  |
| 3 | status | bit | 1 |  | No |  | Not Deleted: true.  Deleted: false. |
| 4 | created\_date | date | 7 |  | No |  |  |
| 5 | title | bit | 4000 |  | Yes |  |  |
| 6 | user\_id | bigint | 8 |  | No | FK |  |
| 7 | subject\_id | bigint | 8 |  | No | FK |  |

## 7. answer

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | [content] | nvarchar | 4000 |  | No |  |  |
| 3 | status | bit | 1 |  | No |  | Active: true.  Deleted: false. |
| 4 | created\_date | date | 7 |  | No |  |  |
| 5 | question\_id | bigint | 8 |  | Yes | FK |  |
| 6 | user\_id | bigint | 8 |  | No | FK |  |
| 7 | reply\_id | bigint | 8 |  | No |  |  |

## 8. course

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | cost | float | 20 |  | Yes |  |  |
| 3 | course\_description | nvarchar | 4000 |  | No |  |  |
| 4 | course\_name | nvarchar | 4000 |  | No |  |  |
| 5 | grade | int | 2 |  | Yes |  |  |
| 6 | length | int |  |  | Yes |  |  |
| 7 | public\_status | bit | 1 |  | No |  |  |
| 8 | status | bit | 1 |  | No |  | Not Deleted: true.  Deleted: false. |
| 9 | subject\_id | bigint | 8 |  | No | FK |  |
| 10 | tutor\_id | bigint | 8 |  | No | FK |  |

## 9. course\_student

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | learning\_status | bit | 1 |  | No |  |  |
| 3 | status | bit | 1 |  | No |  | Not Deleted: true.  Deleted: false. |
| 4 | course\_id | bigint | 8 | Yes | No | FK |  |
| 5 | student\_id | bigint | 8 | Yes | No | FK |  |

## 10. course\_material

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | description | nvarchar | 4000 |  | No |  |  |
| 3 | file\_attach | varchar | 255 |  | No |  |  |
| 4 | link\_share | varchar | 255 |  | No |  |  |
| 5 | status | bit | 1 |  | No |  | Not Deleted: true.  Deleted: false. |
| 6 | title | nvarchar | 255 |  | No |  |  |
| 7 | course\_id | bigint | 8 | Yes | No | FK |  |

## 11. course\_timetable

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | bigint | 8 | Yes | Yes | PK |  |
| 2 | day | int | 2 |  | No |  |  |
| 3 | end\_time | time | 7 |  | No |  |  |
| 4 | start\_time | time | 7 |  | No |  |  |
| 5 | status | bit | 1 |  | No |  | Not Deleted: true.  Deleted: false. |
| 6 | course\_id | bigint | 8 | Yes | No | FK |  |