

**BOOKIE - EBOOK SYSTEM**

**Software Design Document**

– Hanoi, September 2022 –

Table of Contents

[I. Overview 3](#_heading=h.gjdgxs)

[1. Code Packages/Namespaces 3](#_heading=h.30j0zll)

[II. Code Designs 4](#_heading=h.1fob9te)

[1. Homepage 4](#_heading=h.3znysh7)

[a. Class Diagram 4](#_heading=h.2et92p0)

[b. Class Specifications 4](#_heading=h.tyjcwt)

[c. Sequence Diagram(s) 6](#_heading=h.3dy6vkm)

[d. Database queries 6](#_heading=h.1t3h5sf)

[2. Signup 7](#_heading=h.4d34og8)

[a. Class Diagram 7](#_heading=h.2s8eyo1)

[b. Class Specifications 7](#_heading=h.17dp8vu)

[c. Sequence Diagram(s) 8](#_heading=h.3rdcrjn)

[d. Database queries 8](#_heading=h.26in1rg)

[3. Login 10](#_heading=h.lnxbz9)

[a. Class Diagram 10](#_heading=h.35nkun2)

[b. Class Specifications 10](#_heading=h.1ksv4uv)

[c. Sequence Diagram(s) 11](#_heading=h.44sinio)

[d. Database queries 11](#_heading=h.2jxsxqh)

[4. Logout 11](#_heading=h.z337ya)

[a. Class Diagram 11](#_heading=h.3j2qqm3)

[b. Class Specifications 11](#_heading=h.1y810tw)

[c. Sequence Diagram(s) 12](#_heading=h.4i7ojhp)

[d. Database queries 12](#_heading=h.2xcytpi)

[5. Forget password 12](#_heading=h.1ci93xb)

[a. Class Diagram 12](#_heading=h.3whwml4)

[b. Class Specifications 12](#_heading=h.2bn6wsx)

[c. Sequence Diagram(s) 14](#_heading=h.147n2zr)

[d. Data queries 14](#_heading=h.23ckvvd)

[6. View Admin Profile 15](#_heading=h.ihv636)

[a. Class Diagram 15](#_heading=h.32hioqz)

[b. Class Specifications 15](#_heading=h.1hmsyys)

[c. Sequence Diagram(s) 16](#_heading=h.41mghml)

[d. Database queries 16](#_heading=h.2grqrue)

[7. Update Admin Profile 17](#_heading=h.vx1227)

[a. Class Diagram 17](#_heading=h.3fwokq0)

[b. Class Specifications 17](#_heading=h.1v1yuxt)

[c. Sequence Diagram(s) 18](#_heading=h.4f1mdlm)

[d. Database queries 18](#_heading=h.2u6wntf)

[8. Forget password 18](#_heading=h.19c6y18)

[a. Class Diagram 18](#_heading=h.3tbugp1)

[b. Class Specifications 18](#_heading=h.28h4qwu)

[c. Sequence Diagram(s) 20](#_heading=h.nmf14n)

[d. Data queries 20](#_heading=h.37m2jsg)

[9. Code Packages/Namespaces 21](#_heading=h.1mrcu09)

[10. Profile 21](#_heading=h.46r0co2)

[a. Class Diagram 21](#_heading=h.2lwamvv)

[b. Class Specifications 22](#_heading=h.111kx3o)

[c. Sequence Diagram(s) 22](#_heading=h.3l18frh)

[11. Edit profile 22](#_heading=h.206ipza)

[a. Class Diagram 22](#_heading=h.4k668n3)

[b. Class Specifications 23](#_heading=h.2zbgiuw)

[c. Sequence Diagram(s) 24](#_heading=h.1egqt2p)

[d. Database queries 24](#_heading=h.3ygebqi)

[12. View security 24](#_heading=h.2dlolyb)

[a. Class Diagram 24](#_heading=h.sqyw64)

[b. Class Specifications 24](#_heading=h.3cqmetx)

[c. Sequence Diagram(s) 25](#_heading=h.1rvwp1q)

[13. Change password 26](#_heading=h.4bvk7pj)

[a. Class Diagram 26](#_heading=h.2r0uhxc)

[b. Class Specifications 26](#_heading=h.1664s55)

[c. Sequence Diagram(s) 27](#_heading=h.3q5sasy)

[d. Database queries 27](#_heading=h.25b2l0r)

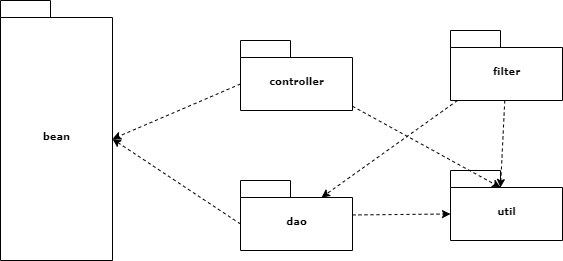
[III. Database Design 28](#_heading=h.kgcv8k)

[1. Database Schema 28](#_heading=h.34g0dwd)

[2. Table Description 28](#_heading=h.1jlao46)

# I. Overview

## 1. Code Packages/Namespaces



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

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | DAO | Used to contain files that are related to Database Connected Files |
| 02 | Controller | Used to contain files that are directly control the system |
| 03 | util | Used to contain files that helps the controllers to manage the system |
| 04 | filter | Used to contain files whose purpose are user authorization and encoding the web |

# II. Code Designs

## 1. Homepage

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

**- BookDAO class**

| No | Method | Description |
| --- | --- | --- |
| 01 | getNumberBook | Return an integer which is quantity of books |
| 02 | getWeeklySaleBooks | Return a list contains 4 that has sale in week to display in homepage |
| 03 | getFeaturedBooks | Random selected 15 books in all book to display in homepage |
| 04 | getFavouriteBooks | Get top 3 books that has highest user choose to display in homepage |

**- Book class**

| No | Method | Description |
| --- | --- | --- |
| 01 | Book(int id, String title, String author, int categoryid, float rating, int favourite, float price, boolean issale, String image, String description, int views) | Get all information of a books for details |
| 02 | Book(int id, String title, String author, float price, boolean issale, int favourite, String image) | Get general information of a books for display on homepage |

**- Home class**

| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | - ArrayList to take a list from getFeaturedBooks() from Book  - ArrayList to take a list from getWeeklySaleBooks() from Book  - ArrayList to take a list from getFavouriteBooks() from Book |
| 02 | doGet() | Doing processRequest |
| 03 | doPost() | Doing processRequest |

**- DBContext class**

| No | Method | Description |
| --- | --- | --- |
| 01 | getConnection() | Connecting with database |

### c. Sequence Diagram(s)

Diagram

Description automatically generated

### d. Database queries

- getFeaturedBooks(): “select \* from [Book]”

- getWeeklySaleBooks(): "select \* from [Book] where [is\_sale] = 1"

- getFavoriteBooks(): “select top(3)\* from BOOK order by favourite desc”

## 2. Signup

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

**- UserDAO class**

| No | Method | Description |
| --- | --- | --- |
| 01 | getUser | Get username and password |
| 02 | generatePass | Encode users’ password |
| 03 | checkDupEmail | Checking that user’s email is exist or not |
| 04 | checkDupUsername | Checking that user’s username is exist or not |
| 05 | editProfile | Allow user to edit their information |
| 06 | changePassword | Allow user to change their password |
| 07 | getNumberUser | Get Total Number |
| 08 | getAllUser | Taking a list contains all user includes their information |
| 09 | getUserName | Taking user’s ursername |
| 10 | updateUser | Update information of user |
| 11 | checkOldPass | Checking user’s old password when they want to change their one. |

**- User class**

| No | Method | Description |
| --- | --- | --- |
| 01 | User(int id, String name, String gender, String dob, String email, String phone, String address, String username, String password, boolean is\_super) | Get all information of user for detail |
| 02 | User(String name, String gender, String dob, String email, String phone, String address, String username, String password) | Get general information of user |

**- DBContext class**

| No | Method | Description |
| --- | --- | --- |
| 01 | getConnection() | Connecting with database |

### c. Sequence Diagram(s)

Diagram, timeline

Description automatically generated

### 

### d. Database queries

- getUser(): "Select \* from [User] where (username=? OR email=?) AND password=?"

- checkDupEmail(String email): "select \* from [User] where email='" + email + "'"

- checkDupUsername(String key): "select \* from [User] where username='" + key + "'"

- createNewUser(String name, String gender, String dob, String email, String phone, String username, String password): "insert [User] ([fullname],[gender] ,[dob] , [email], [phone],[username], [password])"

+ " values (N'" + name + "',"

+ "N'" + (gender.equals("Male") ? 1 : 0) + "',"

+ "CAST(N'" + dob+ "' AS Date),"

+ "N'" + email + "',"

+ "N'" + phone + "',"

+ "N'" + username + "',"

+ "N'" + password + "')"

- editProfile(int userid, String name, String gender, String dob, String email, String phone, String address): "update [User] set "

+ " [fullname] = N'" + name+ "' AND"

+ " [gender] = " + (gender.equals("male") ? 1 : 0) + " AND"

+ " [dob] = N'" + dob + "' AND"

+ " [phone] = N'" + phone + "' AND"

+ " [address] = N'"+ address+"'"

+ "where [id] = "+userid

- changePassword(int userid, String newPassword): "update [User] set " + " [password] = N'" +newPassword + "'"+ "where [id] = "+userid

- getNumberUser(): "select count([id]) from [User] where [is\_super] = 'False'"

- getAllUsers(): "Select \* from [User] where [is\_super] = 0"

- getUsername(int userid): "select [username] from [User] where [id] = "+userid

- updateUser(int id, String name, String phone, String address): "update [User] set "

+ " [fullname] = N'" +name + "'"

+ ", [phone] = N'"+phone+"'"

+ ", [address] = N'"+address+"'"

+ "where [id] = "+id

- checkOldPass(int id,String oldpass): "select \* from [User] where [id] = "+id+" AND [password]=N'"+oldpass+"'"

## 3. Login

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

#### LoginController

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *doGet()* | *Handles the HTTP <code>GET</code> method.* |
| *2* | *doPost()* | *Handles the HTTP <code>POST</code> method.* |

#### UserDAO

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *getUser()* | *Method used to check if an account record existed in database and get all data of that account.Inputs are username and password.Return an <Code>model.User</Code> represents the user’s data* |

#### DBContext

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *getConnection* | *Method used to connect to the database. Return a <Code>java.sql.Connection</Code> represents a connection to the database.* |

### c. Sequence Diagram(s)

Diagram, schematic

Description automatically generated

### d. Database queries

Select \* from [User] where (username=? OR email=?) AND password=?

## 4. Logout

### a. Class Diagram

**A, Class diagram**

| LogoutController |
| --- |
| processRequest(request: HttpServletRequest, response: HttpServletResponse)  doGet()  doPost() |

### b. Class Specifications

**- Logout Controller**

| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | - Destroy session  - Redirect to Homepage |
| 02 | doGet() | Doing processRequest |
| 03 | doPost() | Doing processRequest |

### c. Sequence Diagram(s)

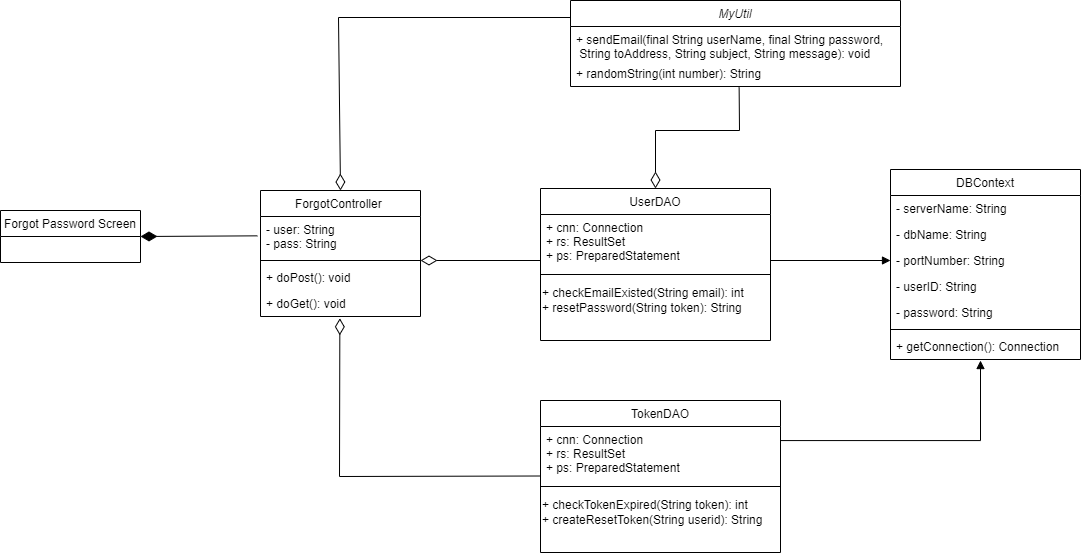
Diagram

Description automatically generated with medium confidence

## d. Database queries

## 5. Forget password

### a. Class Diagram



### b. Class Specifications

#### ForgetController

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *doGet()* | *Handles the HTTP <code>GET</code> method.* |
| *2* | *doPost()* | *Handles the HTTP <code>POST</code> method.* |

#### UserDAO

#### Class Methods

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *checkEmailExisted* | *Method used to check if an email of a User record existed in the database and get id data of that account. Input is email. Return an Integer represents the account‘s id.* |
| *2* | *resetPassword* | *Method used to update password of a User record if the token input is still in expiration time. Return a String represents the new user’s password* |

#### TokenDAO

#### Class Methods

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *createResetToken* | *Method used to insert a token with expiration time to a Token record . Input is the user's id. Return the generated token as a String* |
| *2* | *checkTokenExpired* | *Method used to check a token valid if the token input is still in expiration time. Return an integer represents userId of that token* |

#### 

#### DBContext

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *getConnection* | *Method used to connect to the database. Return a connection to the database* |

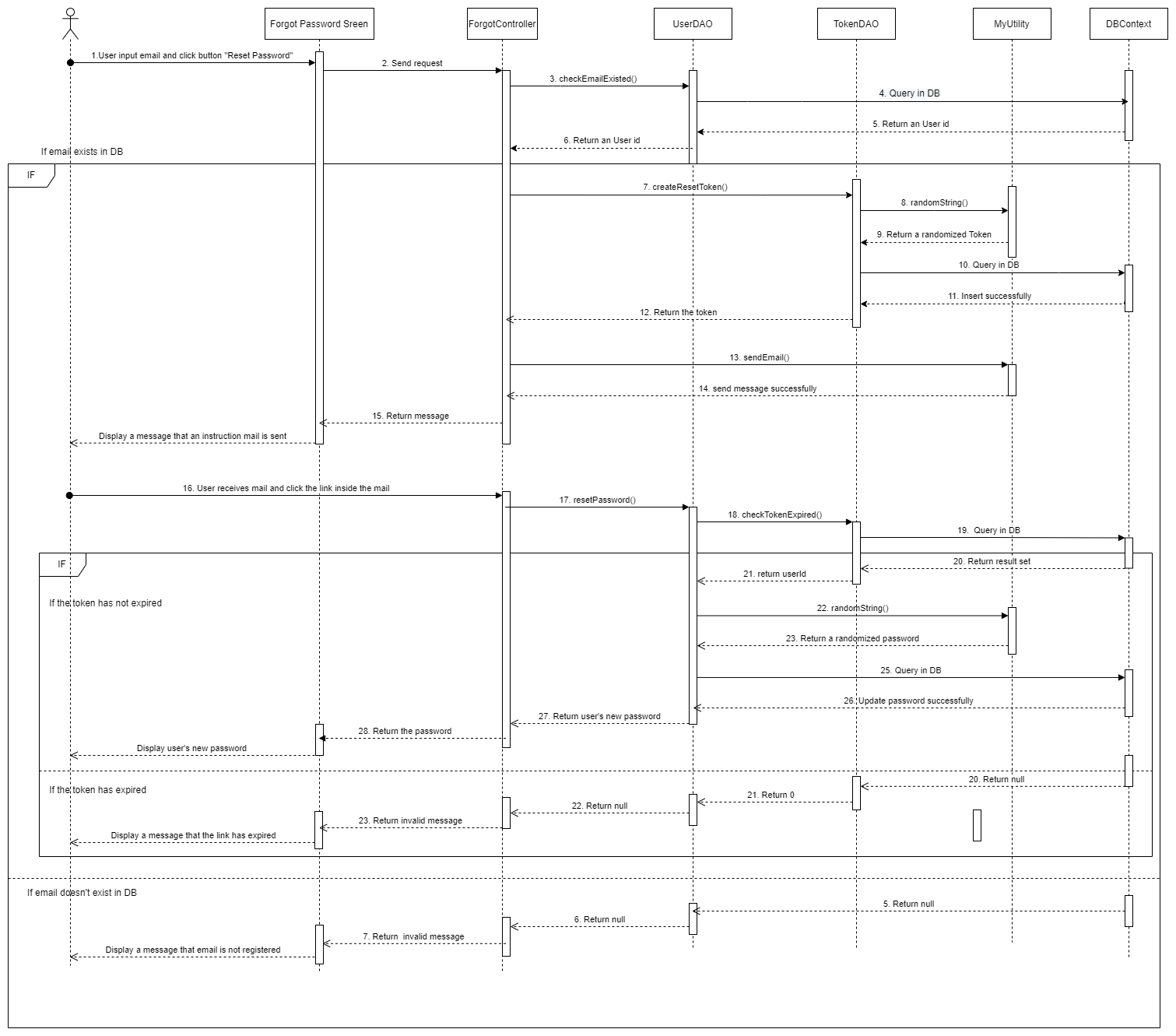
#### MyUtil

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *sendEmail* | *Method used to send an email of a reset instruction. Inputs include recipient’s email and message .* |
| *2* | *randomString* | *Method used to generate a randomized any-number-character string. Return that string.* |

### c. Sequence Diagram(s)

### 

**

### d. Data queries

* **Get id of a record with specific email from User table**

SELECT id

FROM [User]

WHERE [email] = ?

* **Insert a record with specific user id into Token table**

INSERT INTO [Token]

([userId]

, [token

], [expiredDate])

VALUES ( ?

, ?

, ?)

* **Update password of a record in User table**

UPDATE [User]

SET [password] = ?

WHERE [id] = ?

## 6. View Admin Profile

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

#### LoginController

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *doPost()* | *Handles the HTTP <code>POST</code> method.* |

#### EditController

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *doGet()* | *Handles the HTTP <code>GET</code> method.* |

#### UserDAO

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *getUser()* | *Method used to check if an account record existed in database and get all data of that account.Inputs are username and password.Return an <Code>model.User</Code> represents the user’s data* |

#### DBContext

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *getConnection* | *Method used to connect to the database. Return a <Code>java.sql.Connection</Code> represents a connection to the database.* |

### c. Sequence Diagram(s)

Diagram

Description automatically generated

### d. Database queries

Select \* from [User] where (username=? OR email=?) AND password=?

## 7. Update Admin Profile

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

#### EditController

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *doGet()* | *Handles the HTTP <code>GET</code> method.* |
| *2* | *doPost()* | *Handles the HTTP <code>POST</code> method.* |

#### UserDAO

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *editProfile()* | *Method used to let admin update the data of that account.Inputs are admin details.Update those to database.* |

#### DBContext

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *getConnection* | *Method used to connect to the database. Return a <Code>java.sql.Connection</Code> represents a connection to the database.* |

### c. Sequence Diagram(s)

Diagram

Description automatically generated

### d. Database queries

update[User] set

[email] = ?,

[fullname] = ?,

[phone] = ?,

[address] = ?

where [id] = ?

## 8. Forget password

### a. Class Diagram

### b. Class Specifications

#### ForgetController

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *doGet()* | *Handles the HTTP <code>GET</code> method.* |
| *2* | *doPost()* | *Handles the HTTP <code>POST</code> method.* |

#### UserDAO

#### Class Methods

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *checkEmailExisted* | *Method used to check if an email of a User record existed in the database and get id data of that account. Input is email. Return an Integer represents the account‘s id.* |
| *2* | *createResetToken* | *Method used to insert token of a ResetToken record with expiration time. Input is the user's id. Return the generated token as a String* |
| *3* | *resetPassword* | *Method used to update password of a User record if the token input is still in expiration time. Return a String represents the new user’s password* |
|  |  |  |

#### DBContext

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *getConnection* | *Method used to connect to the database. Return a connection to the database* |

#### MyUtil

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *sendEmail* | *Method used to send an email of a reset instruction. Inputs include recipient’s email and message .* |
| *2* | *randomString* | *Method used to generate a randomized any-number-character string. Return that string.* |

### c. Sequence Diagram(s)

### d. Data queries

* **Get id of a record with specific email from User table**

SELECT id

FROM [User]

WHERE email = ?

* **Insert a record with specific user id into ResetToken table**

INSERT INTO [ResetToken] ([uId], [token], [expiredDate])

VALUES ( ? , ? , ?)

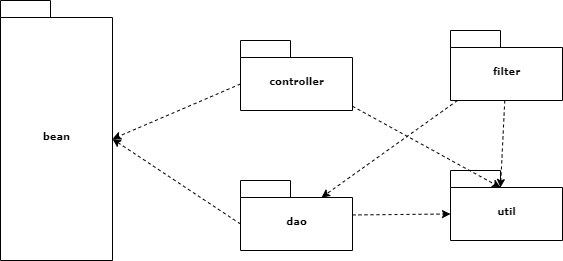
* **Update password of a record with specific token in User table**

UPDATE [User]

SET [password] = ?

WHERE [id] = (select [uid] from [ResetToken] where [token] = ? )

## 9. Code Packages/Namespaces

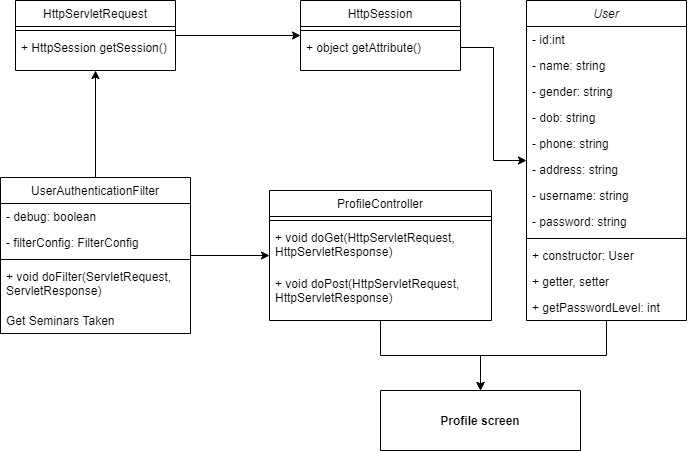


***Package descriptions & package class naming conventions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | DAO | Used to contain files that arre related to Database Connected Files |
| 02 | controller | Used to contain files that are directly control the system |
| 03 | Utilities | Used to contain files that helps the controllers to manage the system |
| 04 | Filter | Used to contain files whose purpose are user authorization and encoding the web |

## 10. Profile

### a. Class Diagram



### b. Class Specifications

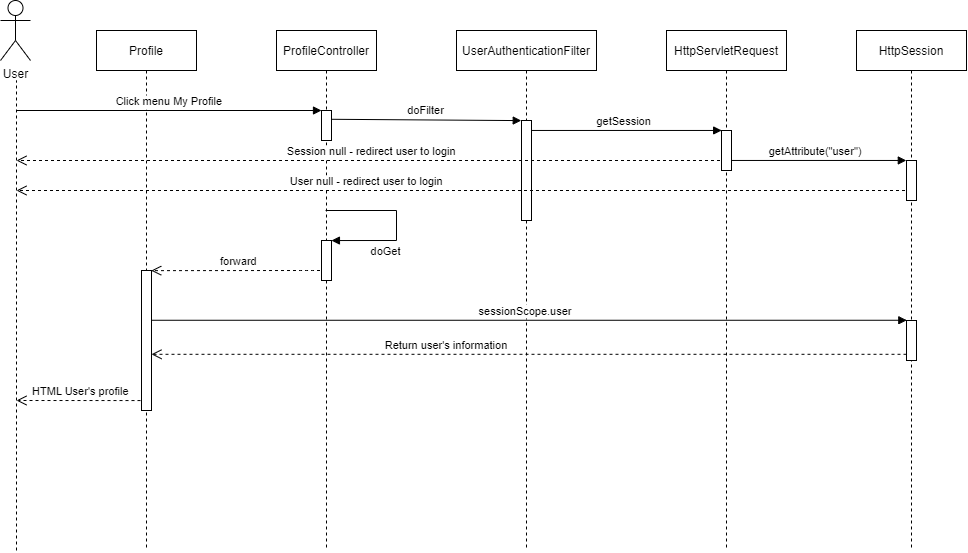
#### Controller.User.ProfileController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Handles the HTTP <code>GET</code> method. |
| 02 | doPost | Handles the HTTP <code>POST</code> method. |

#### Filter.UserAuthenticationFilter

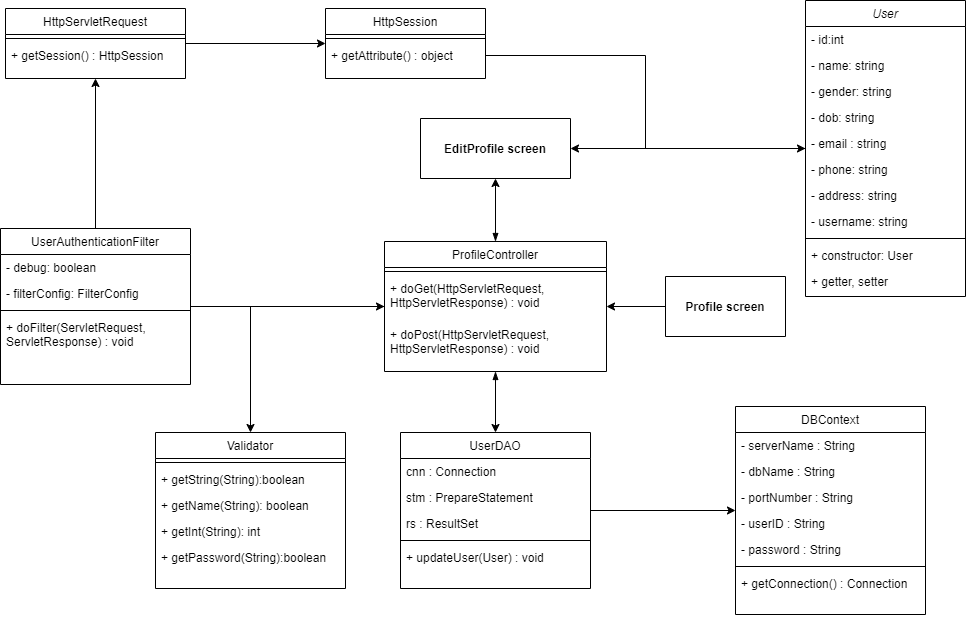
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doFilter | Handles the HTTP <code>GET</code> method. |

### c. Sequence Diagram(s)



## 11. Edit profile

### a. Class Diagram



### b. Class Specifications

#### Controller.User.ProfileController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Handles the HTTP <code>GET</code> method. |
| 02 | doPost | Handles the HTTP <code>POST</code> method. |

#### Filter.UserAuthenticationFilter

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doFilter | Handles the HTTP <code>GET</code> method. |

#### Model.User

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getId | Get ID of user |
| 02 | getName | Get name of user |
| 03 | getGender | Get gender of user |
| 04 | getPhone | Get phone number of user |
| 05 | getAddress | Get address of user |
| 06 | getEmail | Get email of user |
| 07 | setName | Set name of user |
| 08 | setPhone | Set phone number of user |
| 09 | setAddress | Set address of user |
| 10 | setGender | Set gender of user |

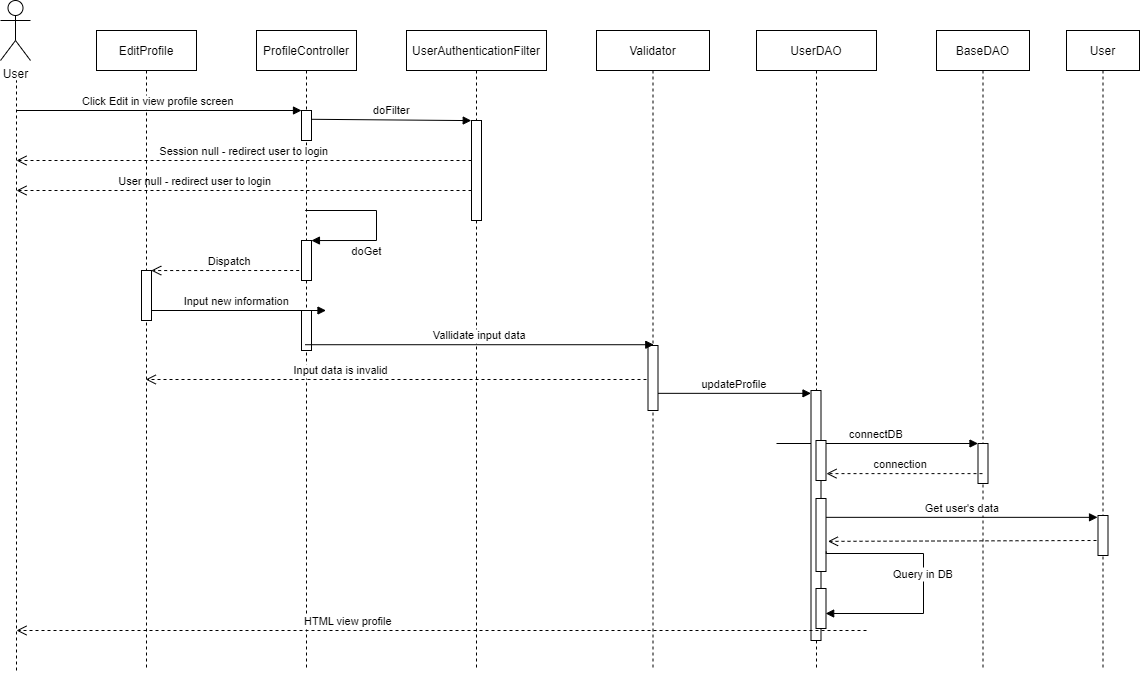
#### Context.UserDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateProfile | Update information of user in database |

#### Model.Validator

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getString | Check if string is empty |
| 02 | getName | Check if name is empty or contains digit or so long |

### c. Sequence Diagram(s)

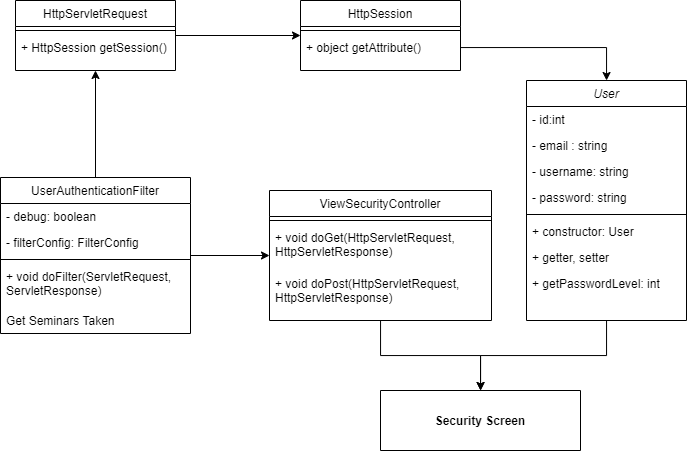


### d. Database queries

update [User] set [fullname] = ?, [phone] = ?, [address] = ?, [gender] = ?, [dob] = ? where [id] = ?

## 12. View security

### a. Class Diagram



### b. Class Specifications

#### Controller.User.ProfileController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Handles the HTTP <code>GET</code> method. |
| 02 | doPost | Handles the HTTP <code>POST</code> method. |

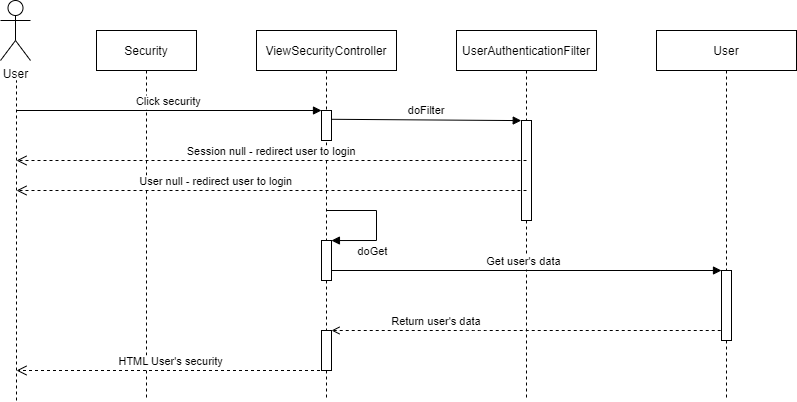
#### Filter.UserAuthenticationFilter

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doFilter | Handles the HTTP <code>GET</code> method. |

#### Model.User

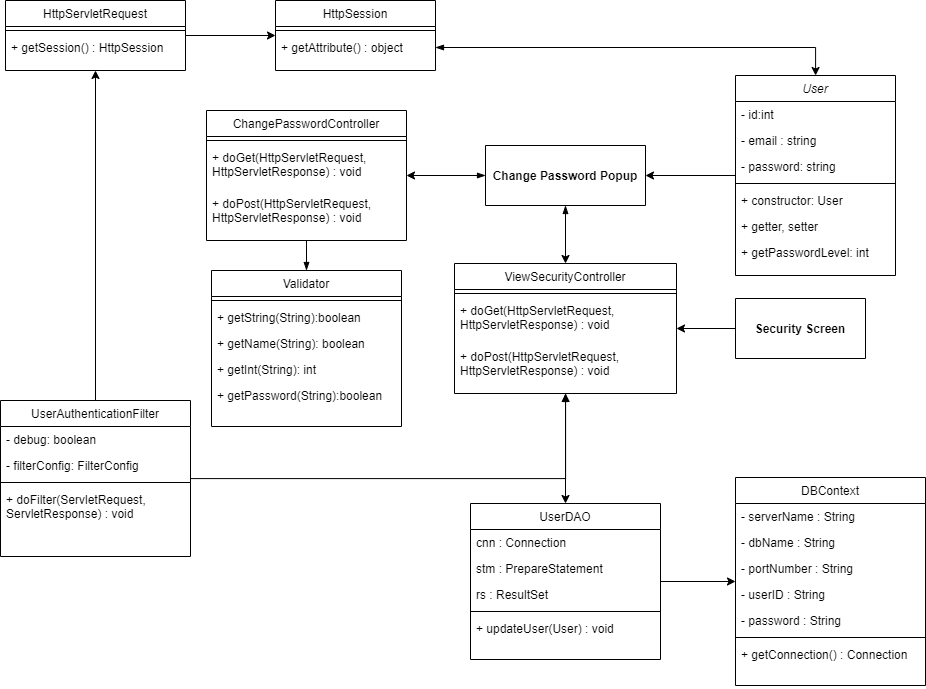
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getEmail | Get email of user |
| 02 | getPasswordLevel | Get level of password |

### c. Sequence Diagram(s)



## 13. Change password

### a. Class Diagram



### b. Class Specifications

#### Controller.User.ProfileController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Handles the HTTP <code>GET</code> method. |
| 02 | doPost | Handles the HTTP <code>POST</code> method. |

#### Filter.UserAuthenticationFilter

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doFilter | Handles the HTTP <code>GET</code> method. |

#### Model.User

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getPassword | Get password of user |
| 02 | getId | Get ID of user |

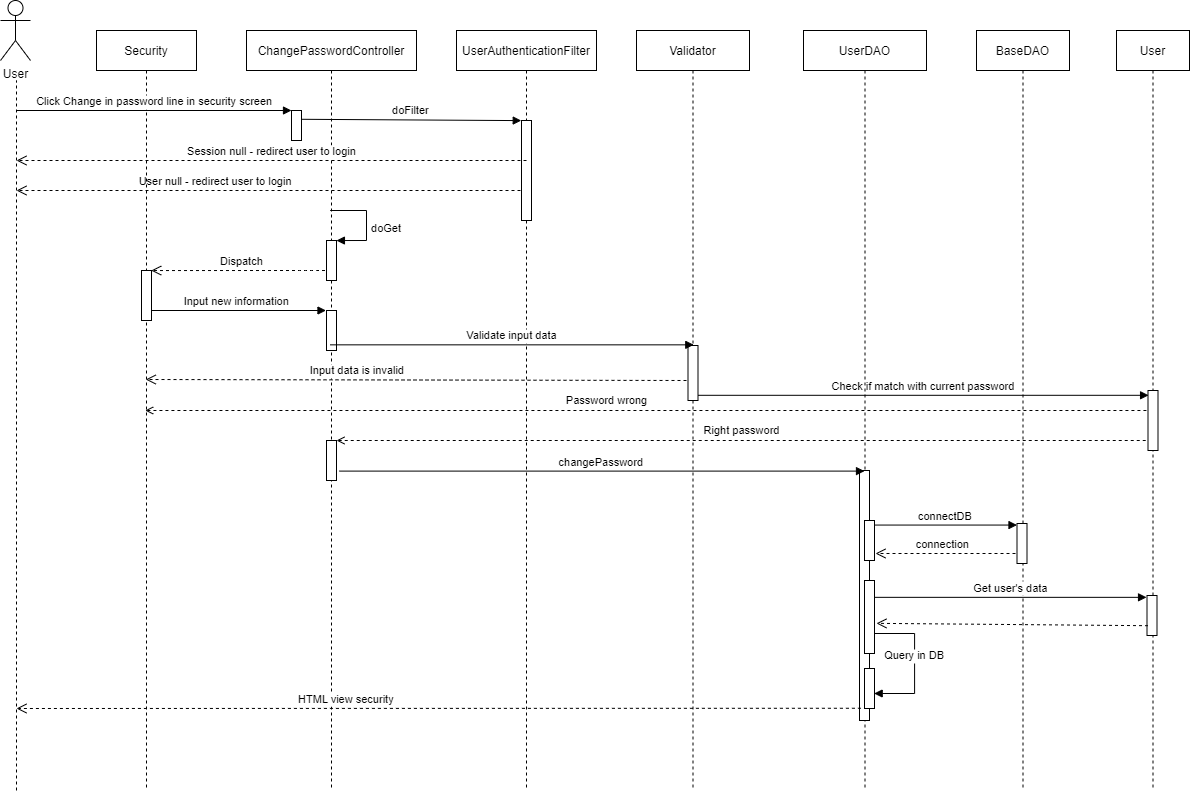
#### Context.UserDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | changePassword | Change password of user by id and new password |

#### Model.Validator

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getString | Check if string is empty |
| 02 | getPassword | Check if password is valid |

### c. Sequence Diagram(s)



### d. Database queries

update [User] set [password] = ? where [id] = ?

# III. Database Design

## 1. Database Schema

Diagram

Description automatically generated

## 2. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | Book | - Primary key: ID  - Foreign key :<none>  - Attribute: tittle, author, categoryID, rating, favourite, price, is\_sale, image, description, views |
| 02 | User | - Primary key: ID  - Foreign key: <none>  - Attribute: fullname, gender, dob, email, phone, address,username, password, is\_super |
| 03 | Favourite | - Primary key: uID  - Foreign key: uID(User: ID), bID(Book: ID)  - Attribute: <none> |
| 04 | Category | - Primary key: ID  - Foreign key: ID(Book: categoryID)  - Attribute: name |
| 05 | Star | - Primary key: bID  - Foreign key: bID(Book: ID)  - Attribute: <none> |
| 06 | ResetToken | - Primary key: uID  - Foreign key: uID(Book: ID)  - Attribute: token, expired data |
| 07 | Volumn | - Primary key: bID, vol  - Foreign key: bID(Book: ID)  - Attribute: vol\_name |
| 08 | Chapter | - Primary key: bID, vol, chapter  - Foreign key: bID(Book: ID), vol(Volumn: vol)  - Attribute: chapter\_name, content |