#### 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* |

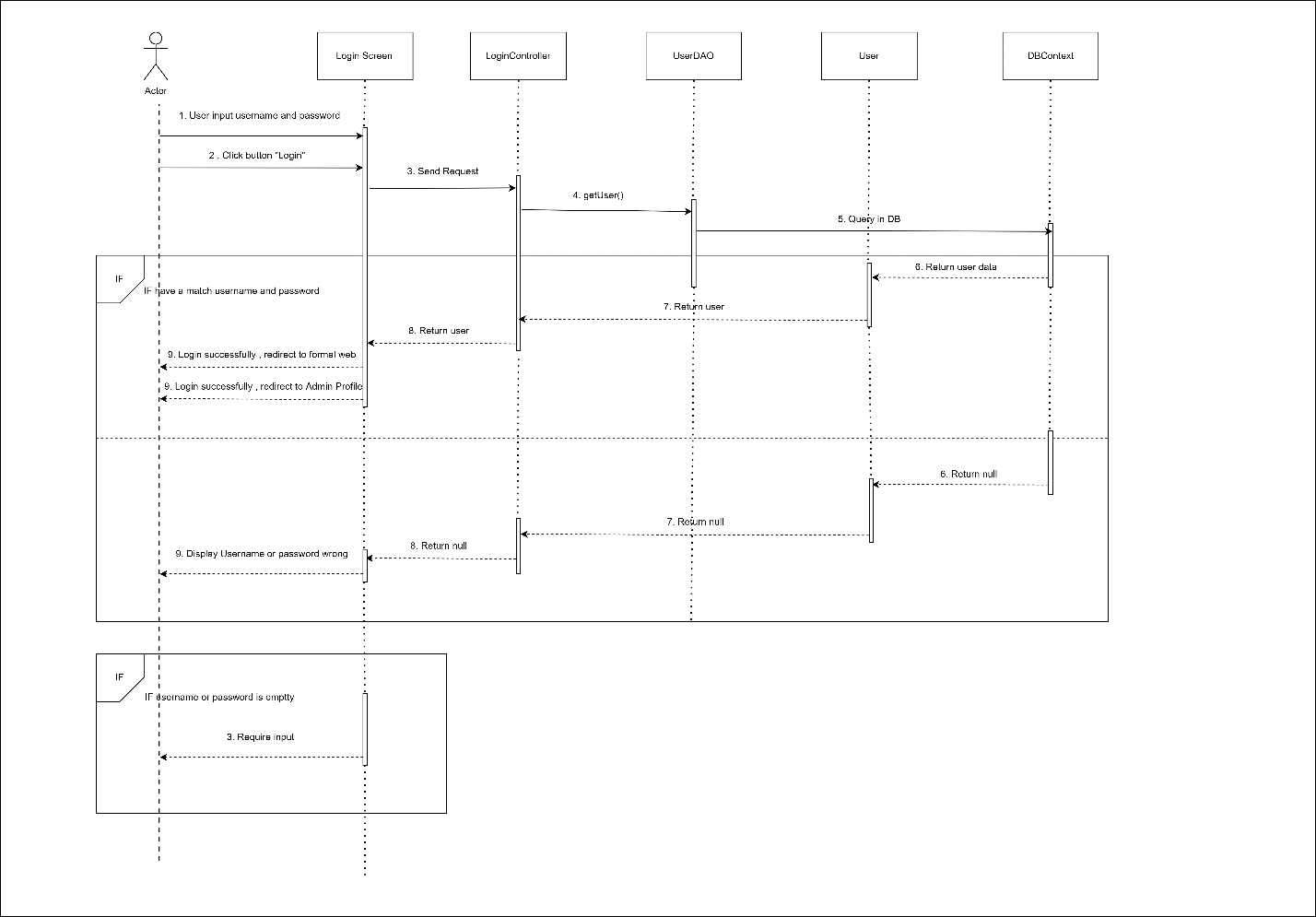
#### DBContext

***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.* |

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

Diagram

Description automatically generated

View Profile

#### 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* |

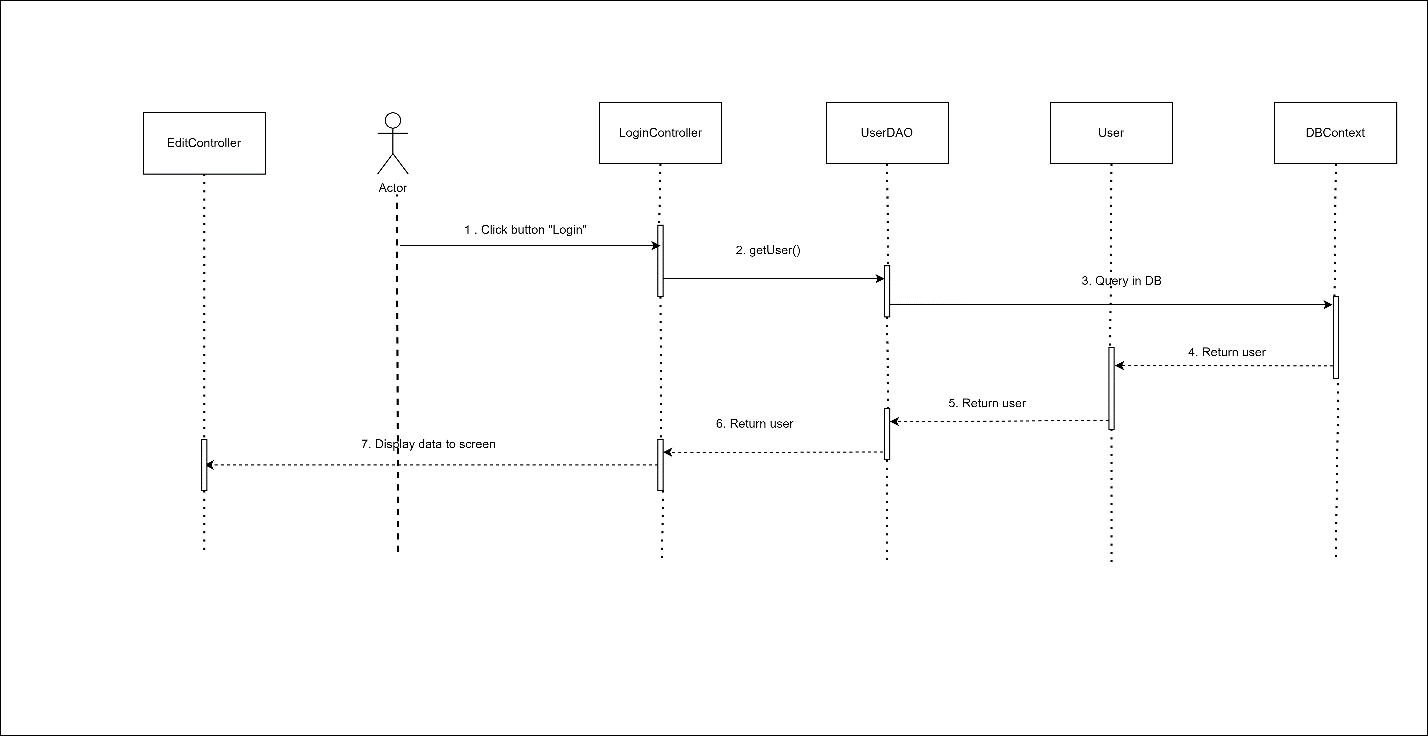
#### DBContext

***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.* |

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

Diagram

Description automatically generated

Update ad min profile

#### 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.* |

update[User] set

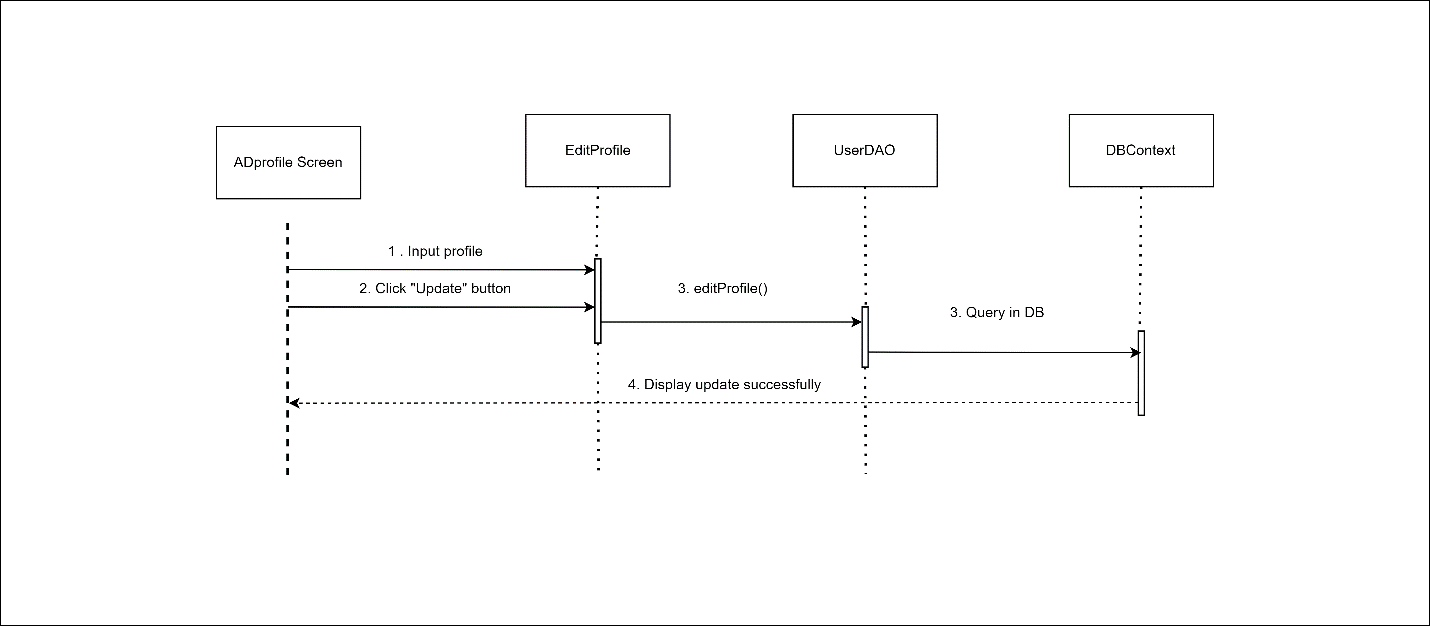
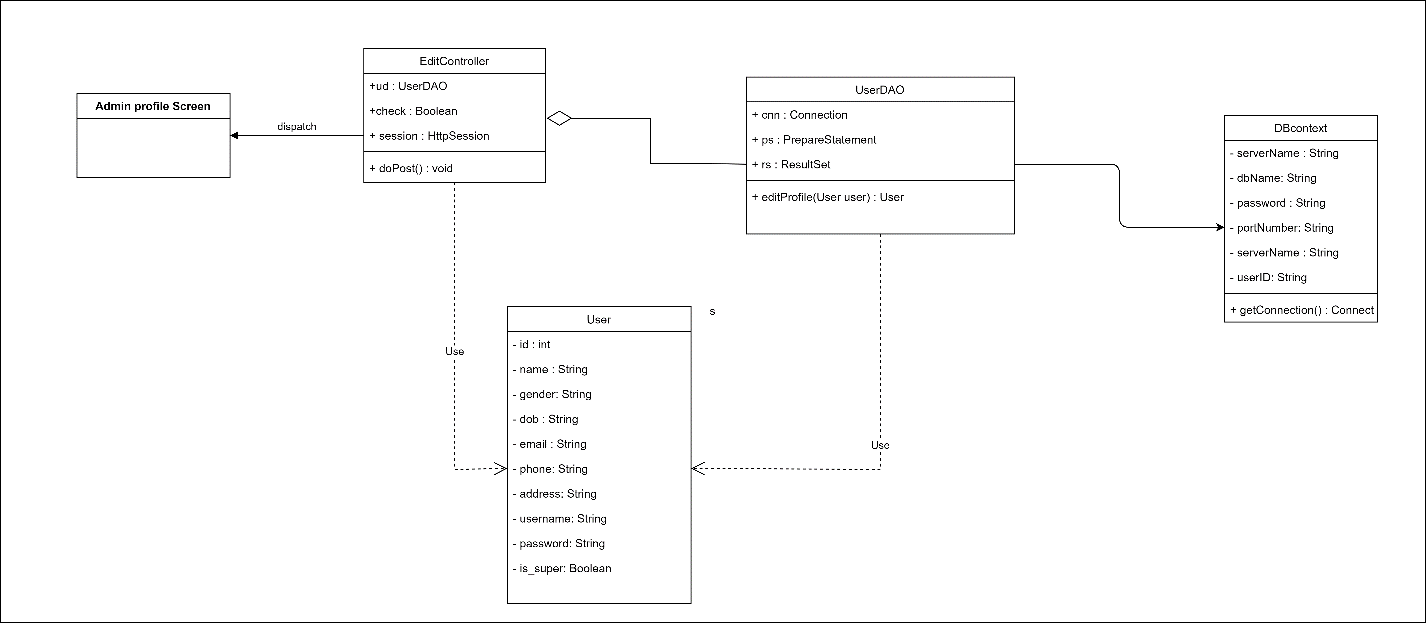
[email] = ?,

[fullname] = ?,

[phone] = ?,

[address] = ?

where [id] = ?



|  |  |
| --- | --- |
| # | Screen (prioritize left to right) |
| #1 | Home, User Register, Login, View Profile, View List Book, Logout, |
| #2 | View Book’s Description, Update Profile, Admin’s dashboard, Change Password |
| #3 | Forget Password, Security Status, Change email |

1. **Homepage**

**A, Class diagram**

Diagram

Description automatically generated

**B, Class Specification**

**- 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 |

**- BookDAO 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 hompage |

**- 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**

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 Specification**

**- 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**

**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. Logout**

**A, Class diagram**

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

**B, Class specification**

**- 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**

Diagram

Description automatically generated with medium confidence

**D, Database queries**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Login

● Function trigger: Click “Login” on pop up home page

● Function description:

o Actor: Admin , User

o Users that have not yet registered cannot log in. They must contact the education unit to create a new account.

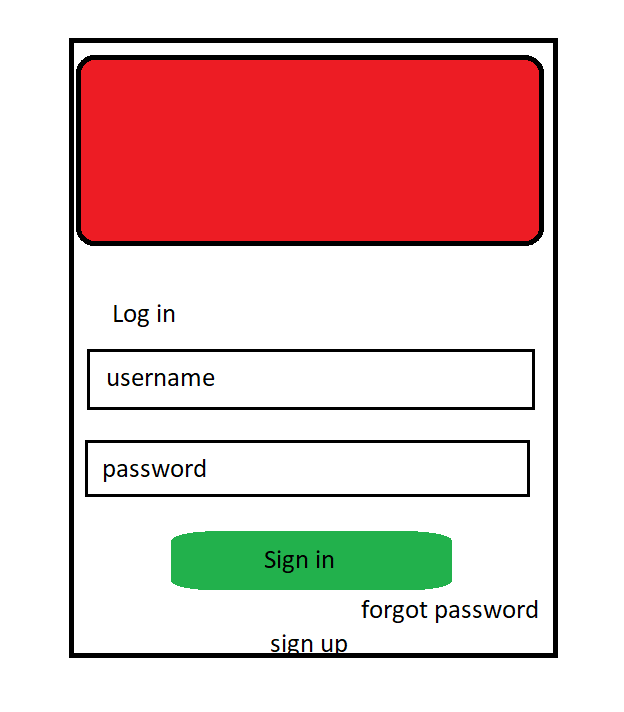
● Function detail:

o Users shall login to the system by using the username and password which is registered.

o Users shall be able to use system functionalities and be directed to the formal screen after successful login.

o The Users will receive “errors” to the login screen if the username or password is invalid.

● Screen layout:



View profile

● Function trigger: Dash board page click “ Profile”

● Function description:

o Actor: Admin

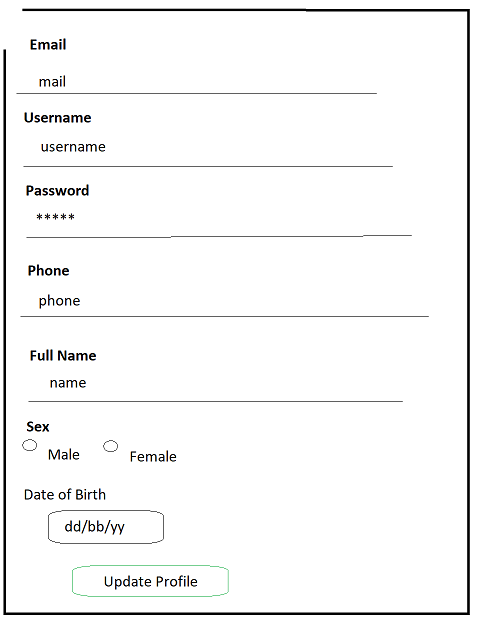
o Admin can view account’s profile

● Function detail:

o The profile of Admin account will display on profile screen

o Admin can view all those details

● Screen layout:



View profile

● Function trigger: From Profile page

● Function description:

o Actor: Admin

o Admin can update profile need to

● Function detail:

o There is a list of data of admin account.

o Admin can check and rewrite what they want

o Click the “Update” button to save the new details

o Notice update successfully

● Screen layout:

Graphical user interface, text

Description automatically generated