**SMART PRESCRIPTION APPLICATION**

Software Design

By

**Mr. Natthakan Kaeokanpai 552115020**

**Mr. Phithiwat   Sitthitun     552115051**

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Dr.Prompong Sugunnasil**

**Document History**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Document Name** | **Version** | **Status** | **Date** | **Viewable** | **Editable** | **Responsible** |
| **Documents** | | | | | | |
| Smart prescription application \_Software design | Software design \_V0.1 | Reviewed | December 10th 2015 | NAT,PHI, PROM | NAT,PHI | NAT,PHI |
| Smart prescription application \_Software design | Software design \_V1.0 | Released | February 20th 2016 | NAT,PHI, PROM | NAT,PHI | NAT,PHI |
| Smart prescription application \_Software design | Software design \_V.1.2 | Reviewed | March 10th 2016 | NAT,PHI, PROM | NAT,PHI | NAT,PHI |

*Acronym*

NAT = Natthakan Kaeokanpai

PHI = Phithiwat Sitthitun

PROM = Prompong Sugunnasil

Content

[Chapter One | Introduction 4](#_Toc428464160)

[1.1 Objective 4](#_Toc428464161)

[1.2 Project Scope 4](#_Toc428464162)

[1.3Acronyms and Definitions 5](#_Toc428464163)

[1.3.1 Acronyms 5](#_Toc428464164)

[1.3.2Definitions 6](#_Toc428464165)

[Chapter Two | System Architecture 7](#_Toc428464167)

[Chapter Four | Database Design 18](#_Toc428464168)

[4.1 ER Diagram 18](#_Toc428464169)

[Chapter Five | Sequence Diagram 20](#_Toc428464170)

[Chapter Six | User Interface Design 50](#_Toc428464171)

# Chapter One | Introduction

## 1.1 Objective

The objective of the Software Design Document for “Smart prescription application” both Android mobile application and web application are to provide a description and explain of the design system including activity following requirements specification and before programming. This document is based on the project proposal, project plan and software requirement specification and help developers to understand how to work for this application

**1.2 Project Scope**

“Smart prescription application” is an android application and web application that uses the electronic prescription in type of QR code that use for doctor, patient, and pharmacist. Smart prescription application require web application for generate QR code and android application to scan QR code pass built-in camera of a mobile phone to capture the QR code converts the QR code into text and numeric. Pharmacist can understand the prescription with name and detail of drug that doctor give to patient.

“Smart prescription application” can also can collect and report information of all prescriptions to the Food and Drug Administration, Ministry of Public Health organization or (FDA) for observing unusually dispensation in Thailand including report drug allergy that have been found in prescription of most of user that be allergic.

The main features of “Smart prescription application” consist of:

**Feature#1: Prescription system**

**Description:** This feature supports the doctor to manage patient’s profile. The doctor can create, delete, search, update and view patient’s profile on the web application.

**Feature#2: Account management system**

**Description:** This feature supports the administrator to manage user’s profile.

The administrators can create, delete, search, update, and view user’s profile on the web application.

**Feature#3: Authentication system**

**Description:** This feature supports accessing to the system and exiting from the system Doctor, FDA, patients, pharmacists and administrator can login to the system and log-out from the system.

**Feature#4: Prescription verification system**

**Description:** This feature supports the patient to view QR code on the mobile application and pharmacist can verify QR code by scan QR code function and pharmacist also can notify the time of dispensation to patient’s prescription on the mobile application.

**Feature#5: Report allergy system**

**Description:** This feature supports the patient to view QR code on the mobile application and pharmacist can verify QR code by scan QR code function.

**Feature#6: Allergy drug summary report**

**Description:** This feature support the FDA can view the report from the doctor.

## 1.3Acronyms and Definitions

### 1.3.1 Acronyms

SDD Software Design Document

CD Class Diagram

URS User Requirement Specification

UI User Interface

SD Sequence Diagram

### 1.3.2Definitions

**Class diagram**

A type of static structure diagram that describes the structure of a system by showing the system's [classes](https://en.wikipedia.org/wiki/Class_(computer_science)), their attributes, operations (or methods), and the relationships among objects.[1]

**Feature**

Activity after input parameters and output the processing to user interface. It explain about functionality of an application used for requirements analysis, coding, design, and testing

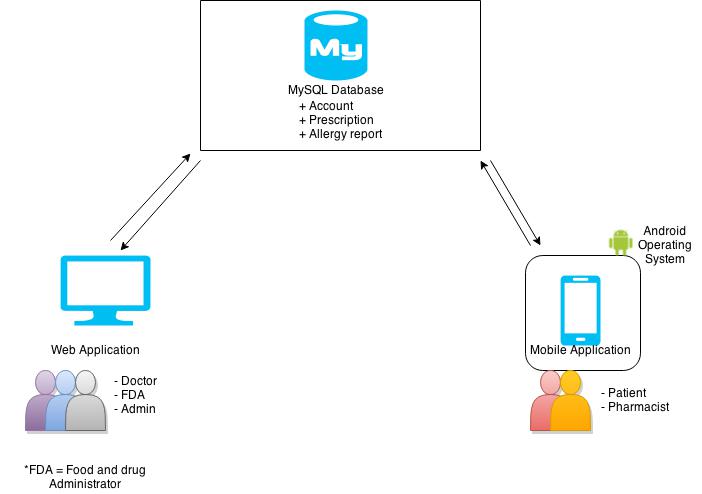
**Sequence diagram**

A sequence diagram is an [interaction diagram](http://en.wikipedia.org/wiki/Interaction_diagram) that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence.

**UML**

is standard notation for the modeling of real-world objects as a first step in developing an object-oriented design methodology [2]

# Chapter Two | System Architecture



*Figure 1: System Architecture (Overview)*

The architecture of “Smart Prescription Application”, show in the figure 10, consists of two parts. The first part is the web application. In this system, there are three types of user doctor, FDA and admin. The users can use web application via a web browser by access to the web application by login. Doctors can create the patient’s profiles and allergy report to MySQL database. The FDAs can view the allergy report on the web application. An admin can create and manage account of doctor and FDA to MySQL database.

The second part is the mobile application Smart Prescription Application stand for android operating system. There have two types of users; patients and pharmacists. Patients can view the QR code on the mobile application. Pharmacist can verify QR code and get the information of patient’s profile by scan QR code function from database on the mobile application.

**MySQL**

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL

**Yii**

Yii is a high-performance PHP framework best for developing Web 2.0 applications.Yii comes with rich features: MVC, DAO/Active Record, I18N/L10N, caching, authentication and role-based access control, scaffolding, testing, etc. It can reduce your development time significantly.

**Android Studio**

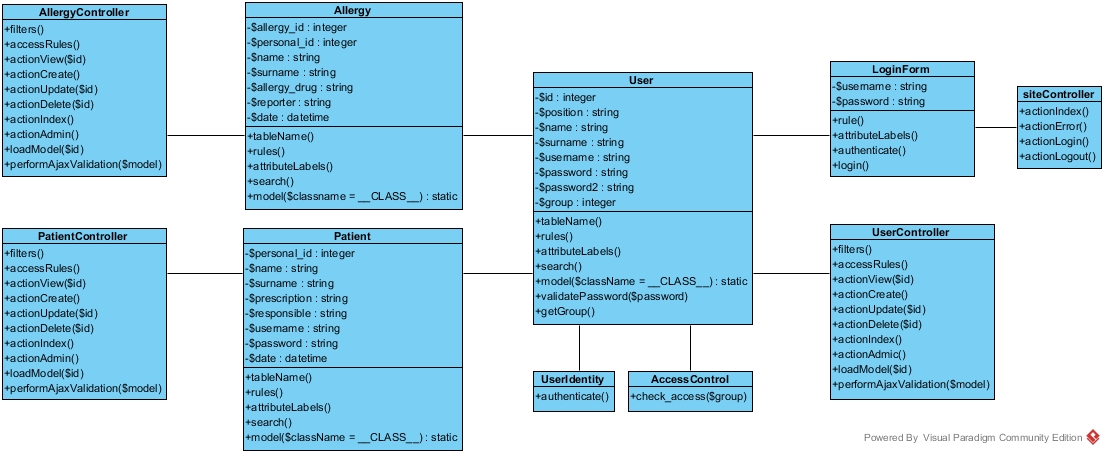
Android Studio is the official IDE for Android application development, based on IntelliJ IDEA. On top of the capabilities, you expect from IntelliJ.

**QR code system**

QR code (abbreviated from Quick Response Code) is the trademark for a type of [matrix barcode](https://en.wikipedia.org/wiki/Matrix_barcode) (or two-dimensional [barcode](https://en.wikipedia.org/wiki/Barcode)) first designed for the [automotive industry in Japan](https://en.wikipedia.org/wiki/Automotive_industry_in_Japan). A barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte / binary, and [kanji](https://en.wikipedia.org/wiki/Kanji)) to efficiently store data; extensions may also be used.

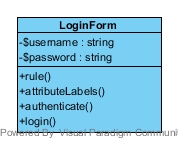
**Chapter Three | Detailed Design**

3.1 Class Diagram of smart prescription application

****

## 3.2 Class Description

3.2.1 Class-01 LoginForm



Description:

LoginForm class is the data structure for keeping user login form data. It is used by the 'login' action of 'SiteController'.

Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Type** |
| 1 | $username | Username of users | String |
| 2 | $password | Password of users | String |

Methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | rules() | Declares the validation rules. The rules state that username and password are required, and password needs to be authenticated. | $username, $password | Array |
| 2 | authenticate() | Authenticates the password. This is the 'authenticate' validator as declared in rules(). | $password | Boolean |
| 3 | login() | Logs in the user using the given username and password in the model. @return boolean whether login is successful | $username, $password | Boolean |

3.2.2 Class-02 SiteController



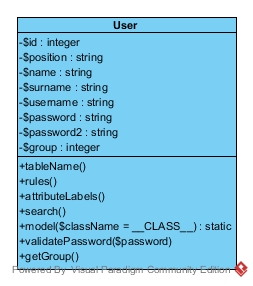
Description:

This class declares the class-based actions.

Methods :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | actionIndex() | This is the default 'index' action that is invoked when an action is not explicitly requested by users. | - | - |
| 2 | actionError() | This is the action to handle external exceptions. | - | - |
| 3 | actionLogin() | This is the action to displays the login page and handle the login in the web application. | - | - |
| 4 | actionLogout() | This is the action to logs out the current user and redirect to homepage. | - | homeUrl |

3.2.4 Class-04 User



Description:

This is the model class for table "tbl\_user". The followings are the available columns in table 'tbl\_user': @property integer $id, @property string position, @property string $name, @property string surname, @property string $username, @property string $password, @property string password2, @property integer $group.

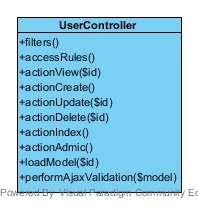
Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Type** |
| 1 | $id | Id of user | Integer |
| 2 | $position | The position of user | String |
| 3 | $name | The name of user | String |
| 4 | $surname | The surname of user | String |
| 5 | $username | The username of user | String |
| 6 | $password | The password of user | String |
| 7 | $password2 | The confirm password of user | String |
| 8 | $group | The groups of user and they are 1, 2, and 4. 1 is Admin. 2 is Doctor. 4 is FDA. | Integer |

Method:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | tableName() | This method is declaring table name in database that used in model. | - | String |
| 2 | rule() | This method defines rules of model attributes that receive user input | $position, $name, $surname, $username, $password, $password2, $group | Array |
| 4 | attributeLabel() | This method to shows attribute labels about information of user that customized. | $position, $name, $surname, $username, $password, $password2, $group | Array |
| 5 | search() | This method is retrieves a list of models based on the current search/filter conditions.  Typical usecase: - Initialize the model fields with values from filter form. - Execute this method to get CActiveDataProvider instance which will filter models according to data in model fields. - Pass data provider to CGridView, CListView or any similar widget. | - | CActiveDataProvider |
| 6 | model($className=\_\_CLASS\_\_) | This method is handle the model class and exact method in all CActiveRecord descendants. | $className | - |
| 7 | validatePassword($password) | This method is password validation of user. | $password | Boolean |
| 8 | getGroup() | This method is used to getting the group of user for separate the type of user. | $group | group |

3.2.5 Class-05 UserController



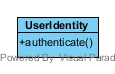
Description:

This class use for controls and manages (create, update and delete) the user’s profile and information. This class controls the user by admin.

Methods :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | filters() | This method is performs access control for CRUD operations and only allow deletion via POST request | - | Array |
| 2 | accessRules() | This method is specifies the access control rules and this method is used by the 'accessControl' filter | - | Array |
| 3 | actionView($id) | Displays a particular model (‘view’ page) by using ID of user. | $id | Array |
| 4 | actionCreate() | This method is used to creating the new data of user into database, the array of data will validated by AJAX validation. If creation is successful, the browser will be redirected to the 'view' page. | - | Array |
| 5 | actionUpdate($id) | This method is used to updating the data of user from the database by using id of user, the array of data will validated by AJAX validation. If update is successful, the browser will be redirected to the 'view' page. The user be updated. | $id | Array |
| 6 | actionDelete($id) | This method is used to deleting the data of user from the database by using id of user. If deletion is successful, the browser will be redirected to the 'admin' page. | $id | Boolean |
| 7 | actionIndex() | This method is used to providing the list all models and handle the model classes. | - | Array |
| 8 | actionAdmin() | This method is used to managing all models of user by admin and managing search() on model class. | - | Array |
| 9 | loadModel($id) | This method is used to returning the data model based on the primary key given in the GET variable. If the data model is not found, an HTTP exception will be raised. | $id | Array |
| 10 | performAjaxValidation($model) | This method is used to performing the AJAX validation, the model is true will be validated. | $model | Boolean |

3.2.6 Class-06 UserIdentity



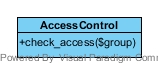
Description:

The UserIdentity class represents the data needed to identity a user. It contains the authentication method that checks if the provided data can identify the user.

Methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | authenticate() | This method is the authentication method that checks if the provided data can identify the user. | $username | Boolean |

3.2.7 Class-07 AccessControl



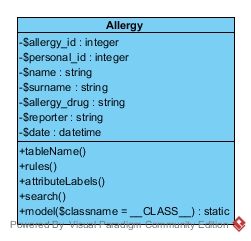
Description :

This class is used to controlling the user that can access to the authenticate web pages. The user will be checked by type of user that inform $group of user.

Methods :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | check\_access($group) | This method is used to checking the types of user that can access into the authenticate web pages. For example, FDA cannot access into the manage user page. | $group | Boolean |

3.2.10 Class-10 Allergy



Description :

This is the model class for table "tbl\_allery". The followings are the available columns in table 'tbl\_allergy': @property integer $allergy\_id, @property integer $personal\_id, @property string $name, @property string $surname, @property string $allergy\_drug, @property string $reporter, and @property datetime $date.

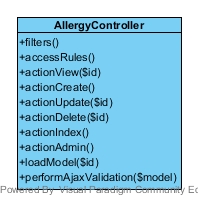
Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Type** |
| 1 | $allergy\_id | This attribute is ID of the allergy report. | Integer |
| 2 | $personal\_id | This attribute is personal ID of patient. | Integer |
| 3 | $name | This attribute is name of patients. | String |
| 4 | $surname | This attribute is surname of patients. | String |
| 5 | $allergy\_drug | This attribute is information of allergy drug. | String |
| 6 | $reporter | This attribute is name of reporter. | String |
| 7 | $date | This attribute is date time of allergy report created. | datetime |

Methods :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | tableName() | This method is declaring table name in database that used in model. | - | String |
| 2 | rule() | This method defines rules of model attributes that receive user input | $personal\_id, $name, $surname, $allergy\_drug, $reporter, $date | Array |
| 4 | attributeLabel() | This method to shows attribute labels about information of allergy that customized. | $allergy\_id,  $personal\_id,  $name, $surname, $allergy\_drug, $reporter, $date | Array |
| 5 | search() | This method is retrieves a list of models based on the current search/filter conditions.  Typical usecase: - Initialize the model fields with values from filter form. - Execute this method to get CActiveDataProvider instance which will filter models according to data in model fields. - Pass data provider to CGridView, CListView or any similar widget. | - | CActiveDataProvider |
| 6 | model($className=\_\_CLASS\_\_) | This method is handle the model class and exact method in all CActiveRecord descendants. | $className | - |

3.2.11 Class-11 AllergyController



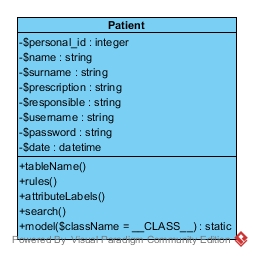
Description :

This class use for controls and manages (create, update and delete) the information of the allergy and some patients. This class controls the allergy report’s information by doctor.

Methods :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | filter() | This method is performs access control for CRUD operations and only allow deletion via POST request | - | Array |
| 2 | accessRules() | This method is specifies the access control rules and this method is used by the 'accessControl' filter | - | Array |
| 3 | actionView($allergy\_id) | Displays a particular model by using ID of allergy report. | $allergy\_id | Array |
| 4 | actionCreate() | This method is used to creating the new data of allergy report into database, the array of data will validated by AJAX validation. If creation is successful, the browser will be redirected to the 'view' page. | - | Array |
| 5 | actionUpdate($allergy\_id) | This method is used to updating the data of user from the database by using id of allergy report, the array of data will validated by AJAX validation. If update is successful, the browser will be redirected to the 'view' page. The allergy report be updated. | $allergy\_id | Array |
| 6 | actionDelete($allergy\_id) | This method is used to deleting the data of allergy report from the database by using ID of allergy report. If deletion is successful, the browser will be redirected to the 'admin' page. | $allergy\_id | Boolean |
| 7 | actionIndex() | This method is used to providing the list all models and handle the model classes. | - | Array |
| 8 | actionAdmin() | This method is used to managing all models of allergy report by doctor and managing search() on model class. | - | Array |
| 9 | loadModel($allergy\_id) | This method is used to returning the data model based on the primary key given in the GET variable. If the data model is not found, an HTTP exception will be raised. | $allergy\_id | Array |
| 10 | performAjaxValidation($model) | This method is used to performing the AJAX validation, the model is true will be validated. | $model | Boolean |

3.2.12 Class-12 Patient



Description :

This is the model class for table "tbl\_patient". The followings are the available columns in table 'tbl\_patient': @property string $personal\_id, @property string $name, @property string $surname, @property string $prescription, @property string $responsible, @property string $username, @property string $password. @property datetime $date,

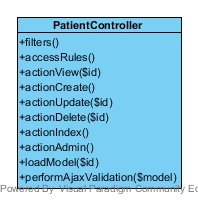
Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Type** |
| 1 | $personal\_id | This attribute is personal id of the patient. | Integer |
| 2 | $name | This attribute is name of the patient. | String |
| 3 | $surname | This attribute is surname of patients. | String |
| 4 | $prescription | This attribute is information of prescription. | String |
| 5 | $responsible | This attribute is name of doctor who are the response that prescription. | String |
| 6 | $username | This attribute is username of patient that doctor create for patient to use the prescription. | String |
| 7 | $password | This attribute is password of patient. | String |
| 8 | $date | This attribute is date time of patient profile created. | datetime |

Methods :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | tableName() | This method is declaring table name in database that used in model. | - | String |
| 2 | rule() | This method defines rules of model attributes that receive user input | $personal\_id, $name, $surname, $prescription, $responsible $username, $password, $date | Array |
| 3 | relations() | This method defines the relations of rules and the relations of class that automatically generated. | - | Array |
| 4 | attributeLabel() | This method to shows attribute labels about information of patient that customized. | $personal\_id,  $name, $surname, $prescription, $responsible, $username, $password, $date | Array |
| 5 | search() | This method is retrieves a list of models based on the current search/filter conditions.  Typical usecase: - Initialize the model fields with values from filter form. - Execute this method to get CActiveDataProvider instance which will filter models according to data in model fields. - Pass data provider to CGridView, CListView or any similar widget. | - | CActiveDataProvider |
| 6 | model($className=\_\_CLASS\_\_) | This method is handle the model class and exact method in all CActiveRecord descendants. | $className | - |

3.2.13 Class-13 PatientController



Description :

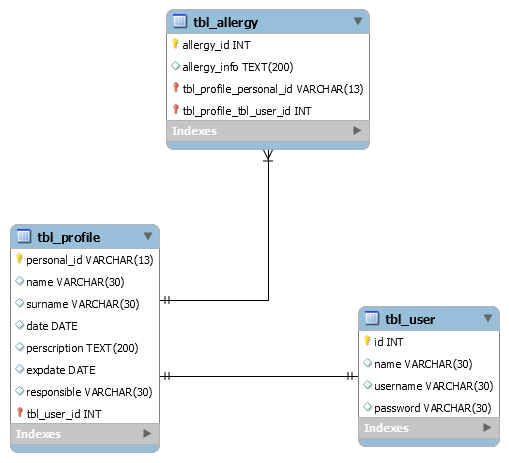
This class use for controls and manages (create, update and delete) the information of patients. This class controls the patient’s information by doctor.

Methods :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Parameter Lists** | **Return Type** |
| 1 | filter() | This method is performs access control for CRUD operations and only allow deletion via POST request | - | Array |
| 2 | accessRules() | This method is specifies the access control rules and this method is used by the 'accessControl' filter | - | Array |
| 3 | actionView($personal\_id) | Displays a particular model of patient profile by using | $personal\_id | Array |
| 4 | actionCreate() | This method is used to creating the new data of patient into database, the array of data will validated by AJAX validation. If creation is successful, the browser will be redirected to the 'view' page. | - | Array |
| 5 | actionUpdate($personal\_id) | This method is used to updating the data of patient from the database by using personal id of patient, the array of data will validated by AJAX validation. If update is successful, the browser will be redirected to the 'view' page. The allergy report be updated. | $personal\_id | Array |
| 6 | actionDelete($personal\_id) | This method is used to deleting the data of patient from the database by using personal ID of patient. If deletion is successful, the browser will be redirected to the 'admin' page. | $personal\_id | Boolean |
| 7 | actionIndex() | This method is used to providing the list all models and handle the model classes. | - | Array |
| 8 | actionAdmin() | This method is used to managing all models of patient by doctor and managing search() on model class.. | - | Array |
| 9 | loadModel($personal\_id) | This method is used to returning the data model based on the primary key given in the GET variable. If the data model is not found, an HTTP exception will be raised. | $personal\_id | Array |
| 10 | performAjaxValidation($model) | This method is used to performing the AJAX validation, the model is true will be validated. | $model | Boolean |

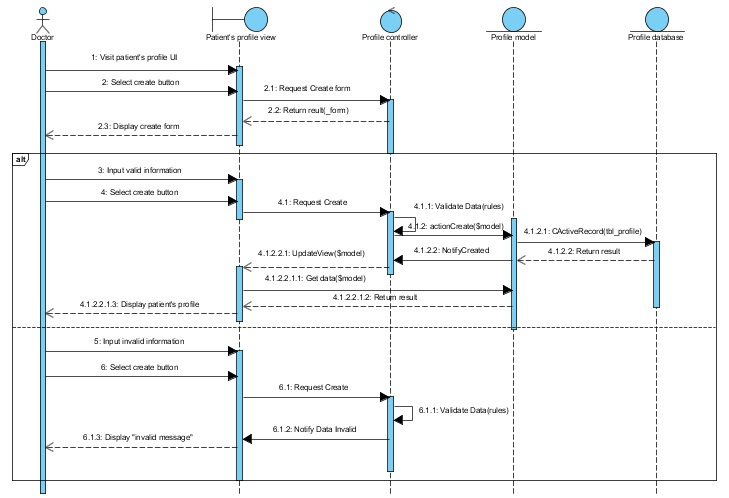
# Chapter Four | Database Design

## 4.1 ER Diagram



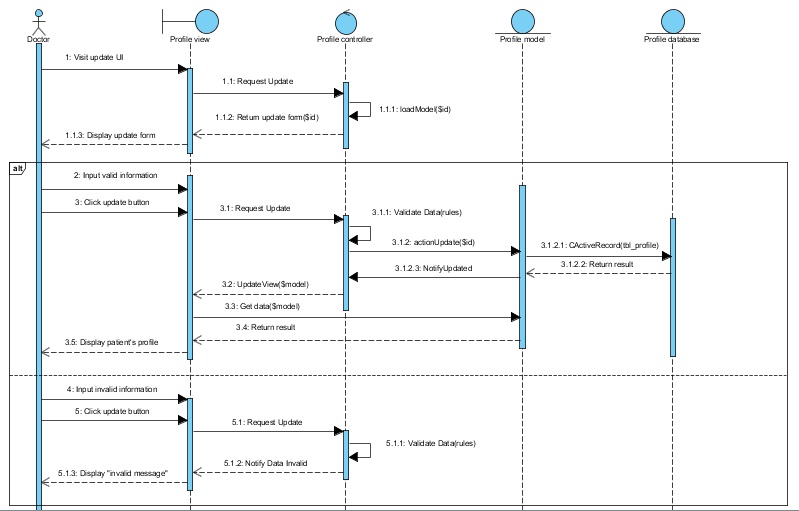
# Chapter Five | Sequence Diagram

SD-01: Doctors can create patient’s profiles on the web application.



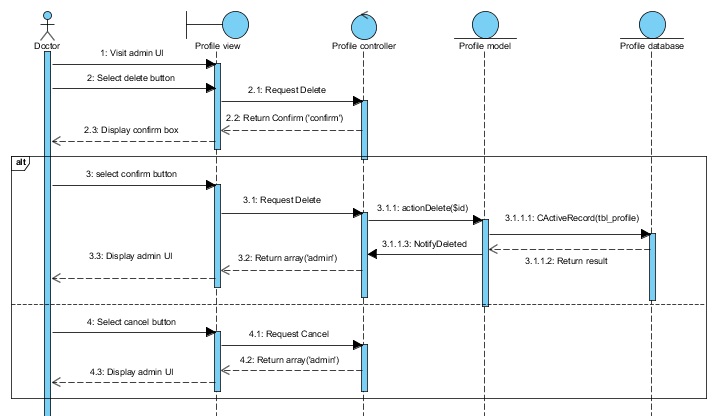
*Figure 1: Sequence Diagram of SD-01*

SD-02: Doctors can update patient’s profiles on the web application.



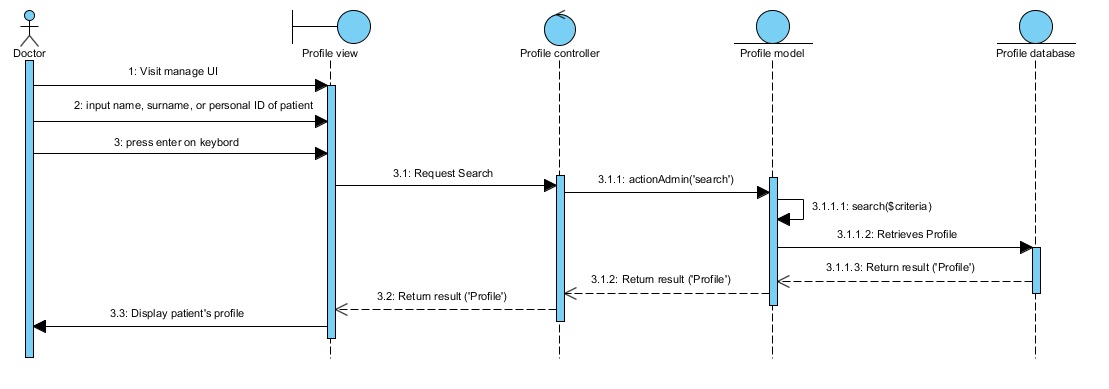
*Figure 1: Sequence Diagram of SD-01*

SD-03: Doctors can delete patient’s profiles on the web application.



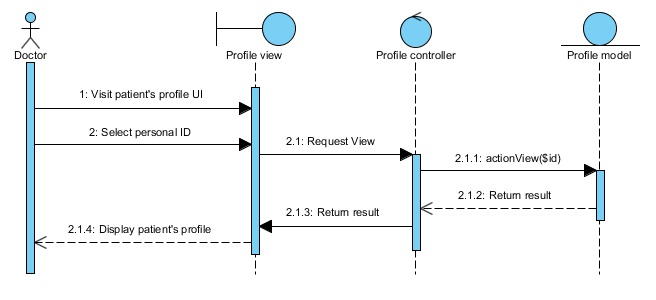
*Figure 1: Sequence Diagram of SD-01*

SD-04: Doctors can search patient’s profiles on the web application.



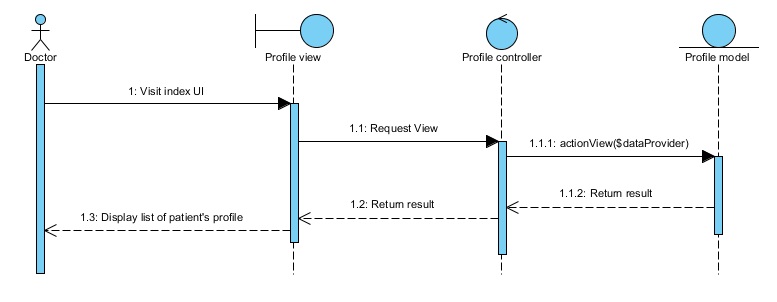
*Figure 1: Sequence Diagram of SD-01*

SD-05: Doctors can view patient’s profiles on the web application.

****

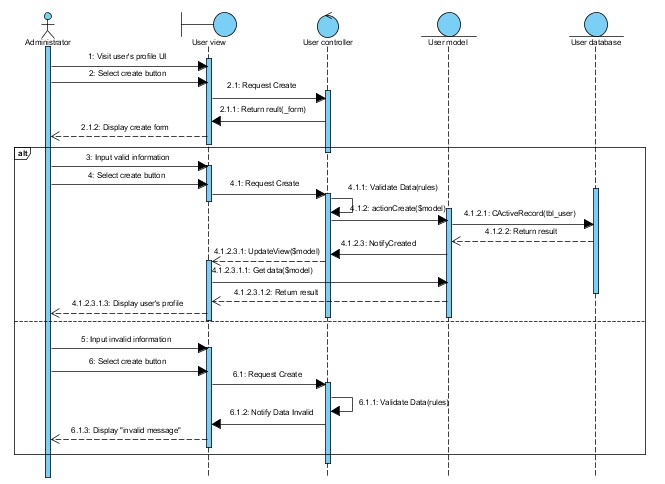
*Figure 1: Sequence Diagram of SD-01*

SD-06: Doctors can view a list of patient’s profiles on the web application.



*Figure 1: Sequence Diagram of SD-01*

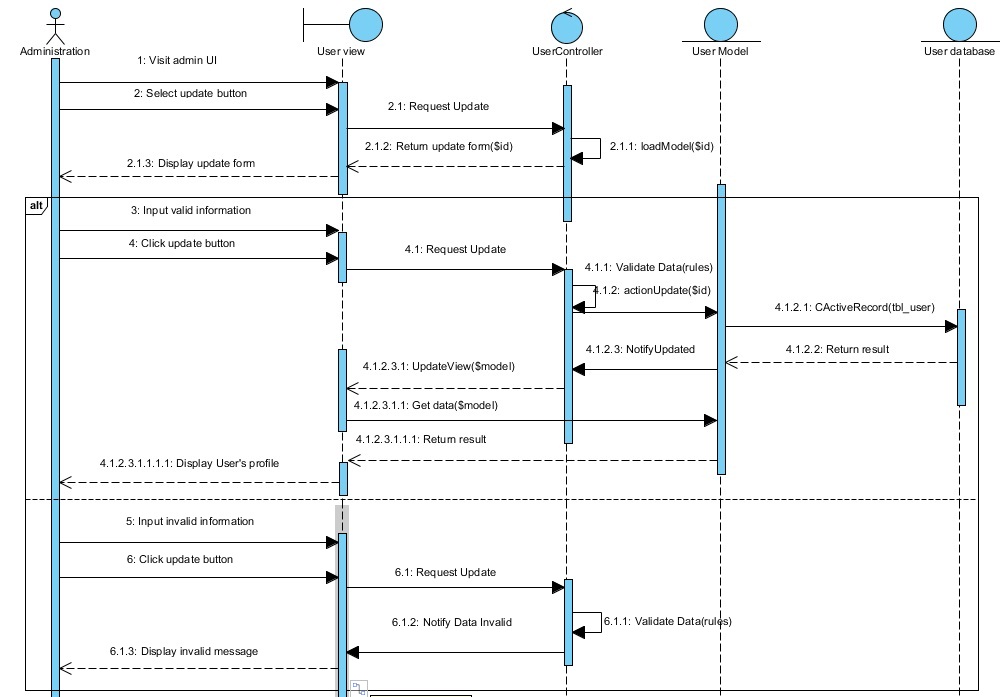
SD-07: Administrations can create user’s profiles on the web application.



*Figure 1: Sequence Diagram of SD-01*

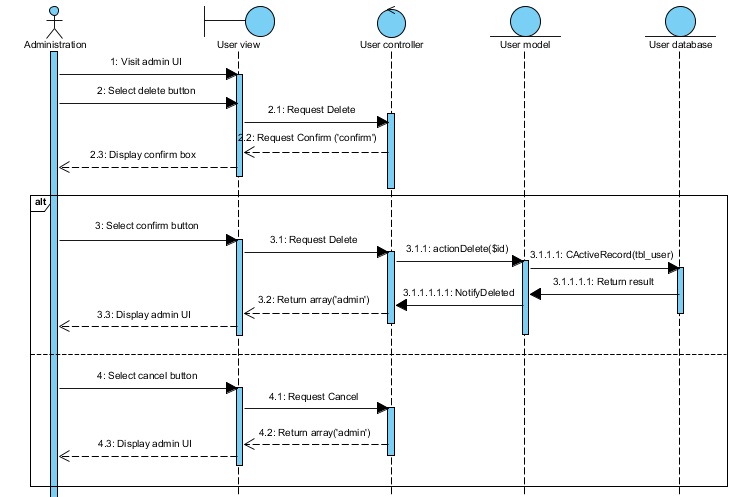
\

SD-08: Administrations can update user’s profiles on the web application.



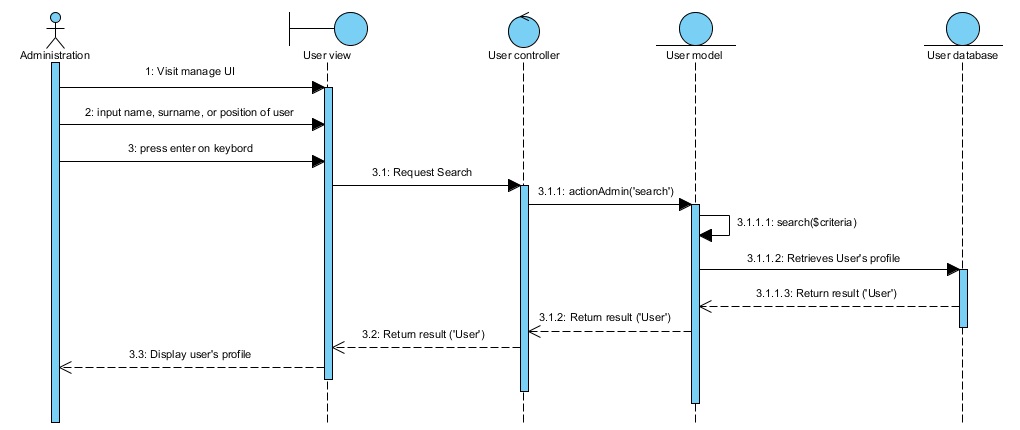
*Figure 1: Sequence Diagram of SD-01*

SD-09: Administrations can delete user’s profiles on the web application.



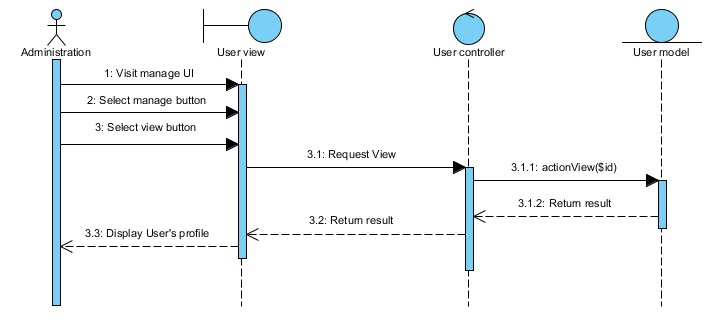
*Figure 1: Sequence Diagram of SD-01*

SD-10: Administrations can search user’s profiles on the web application.



*Figure 1: Sequence Diagram of SD-01*

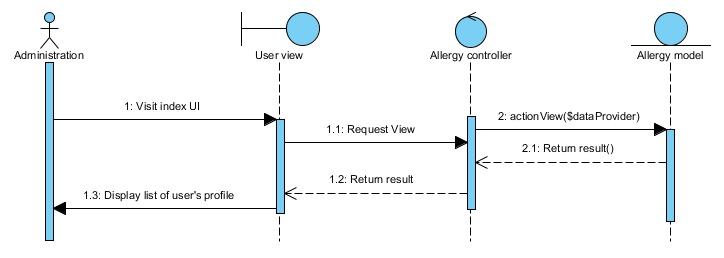
SD-11: Administrations can view user’s profile on the web application.



\

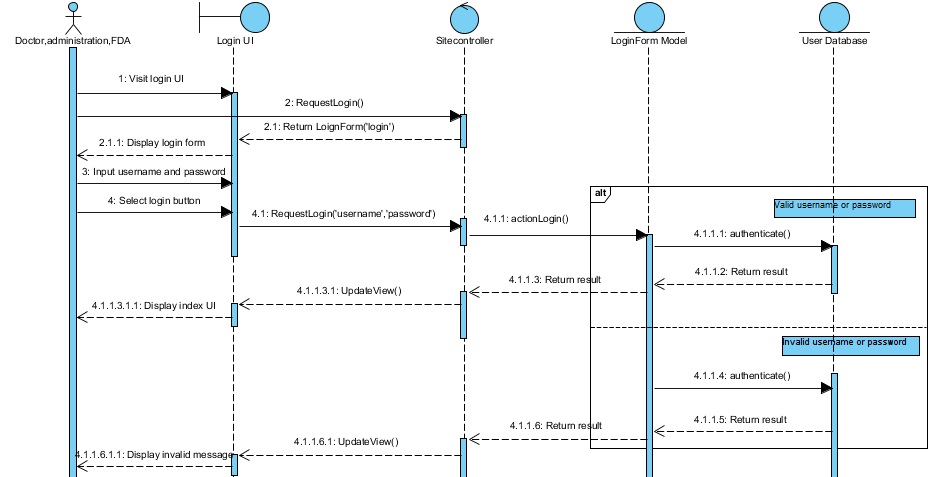
*Figure 1: Sequence Diagram of SD-01*

SD-12: Administrations can view a list of user’s profiles on the web application.



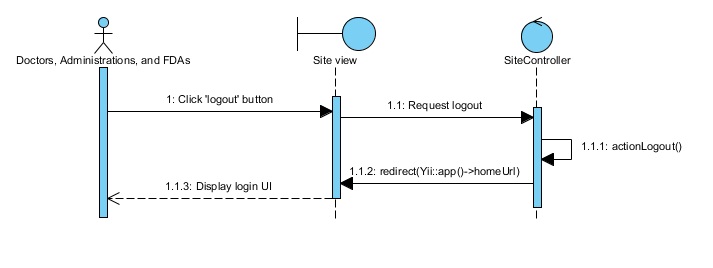
*Figure 1: Sequence Diagram of SD-01*

SD-13: Doctors, Administrations, and FDAs can login to the web application.



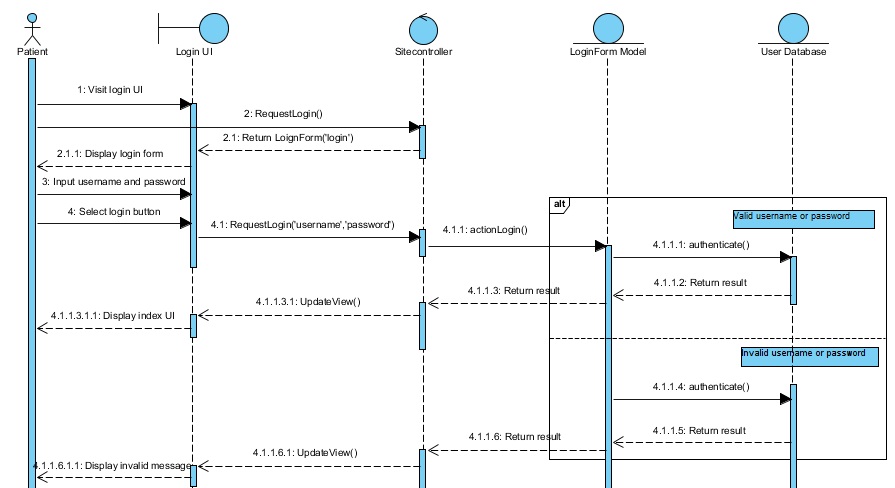
*Figure 1: Sequence Diagram of SD-01*

SD-14: Doctors, Administrations, and FDAs can logout from the web application.



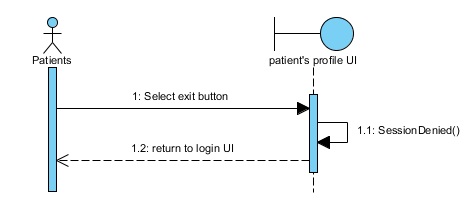
*Figure 1: Sequence Diagram of SD-01*

SD-15: Patients can login to the mobile application.



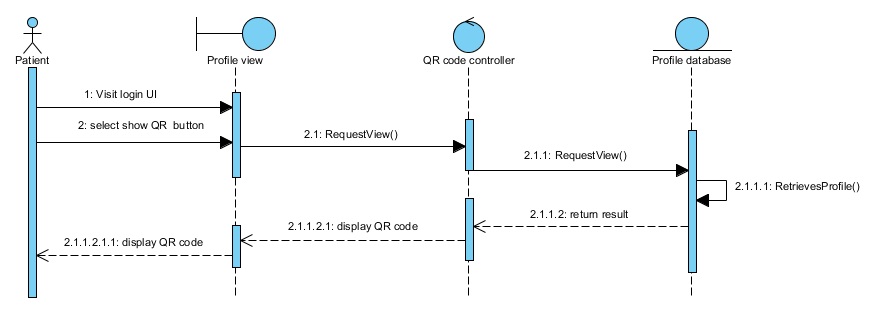
*Figure 1: Sequence Diagram of SD-01*

SD-16: Patients can logout from the mobile application.

****

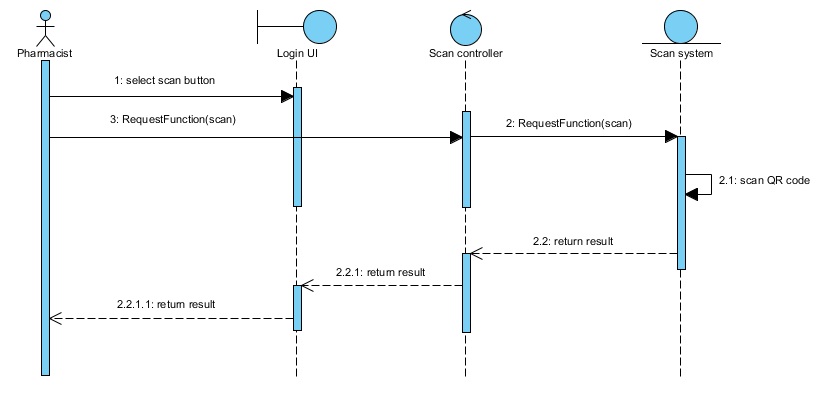
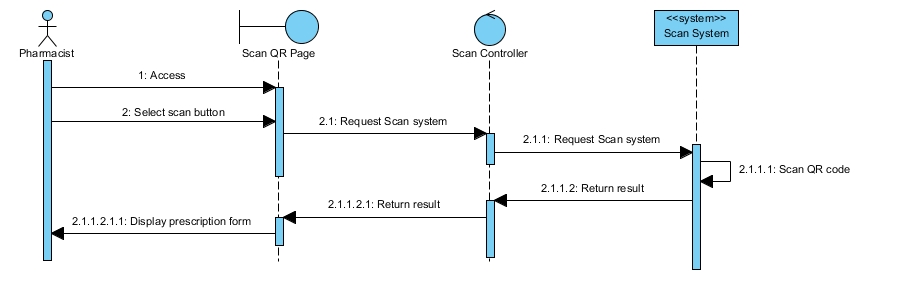
*Figure 1: Sequence Diagram of SD-01*

SD-17: Patients can view the QR code on the mobile application.



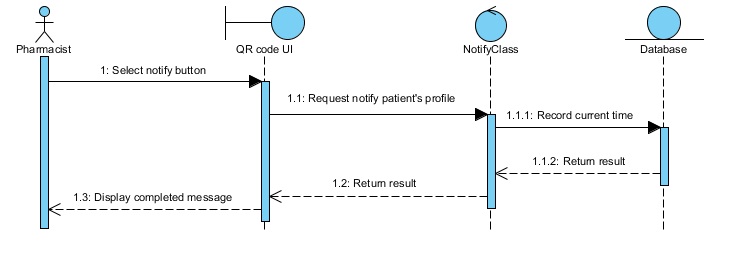
*Figure 1: Sequence Diagram of SD-01*

SD-18: Pharmacists can scan the QR code on the mobile application.

****

*Figure 1: Sequence Diagram of SD-01*

SD-19: Pharmacists can notify the time of dispensation to patient’s profile on the mobile application.



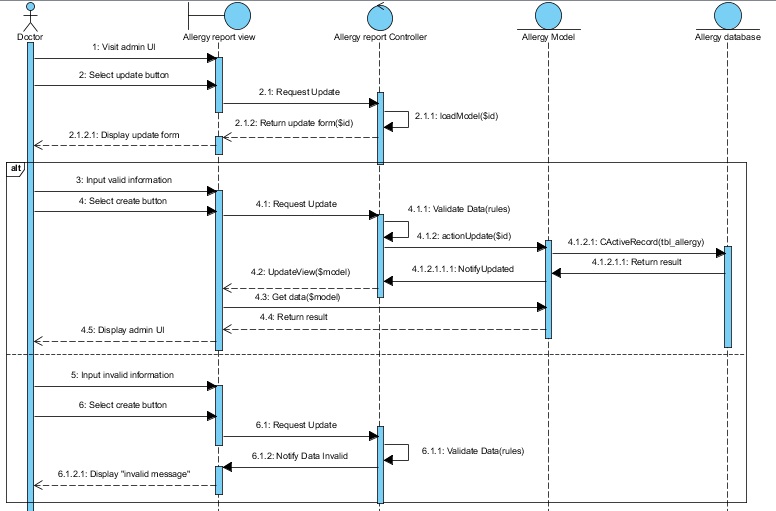
*Figure 1: Sequence Diagram of SD-01*

SD-20: Doctors can create allergy reports on the web application.

# .

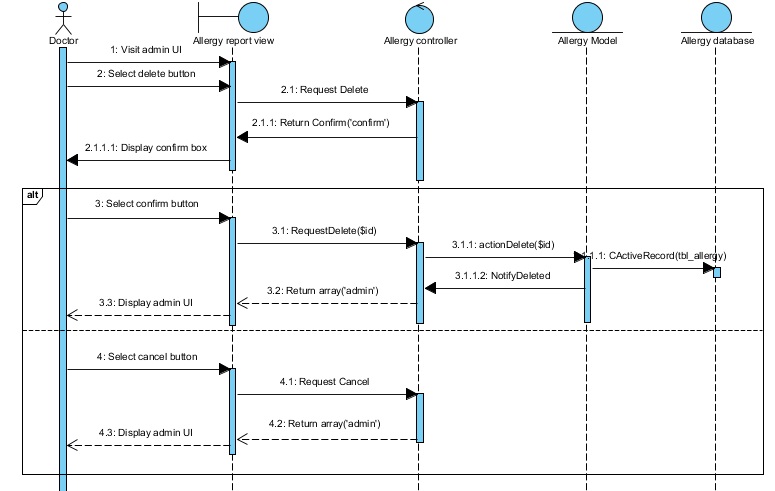
*Figure 1: Sequence Diagram of SD-01*

SD-21: Doctors can update allergy reports on the web application



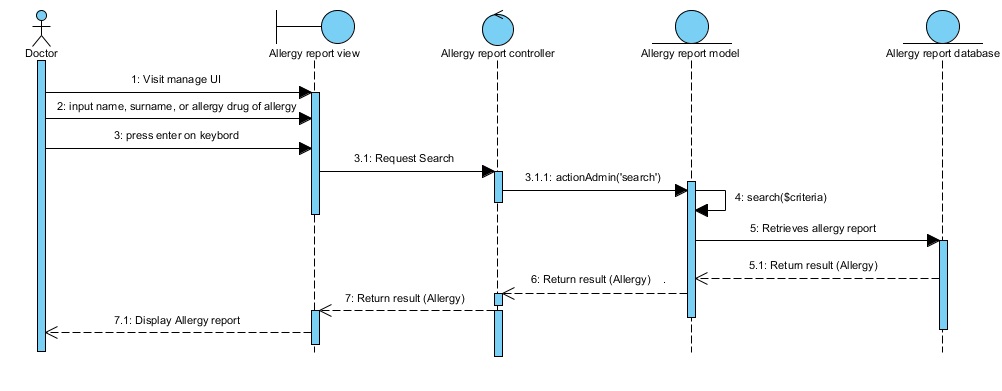
*Figure 1: Sequence Diagram of SD-01*

SD-22: Doctors can delete allergy reports on the web application.



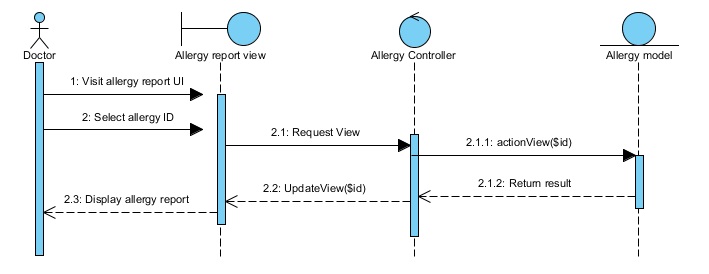
*Figure 1: Sequence Diagram of SD-01*

SD-23: Doctors can search allergy reports on the web application.

****

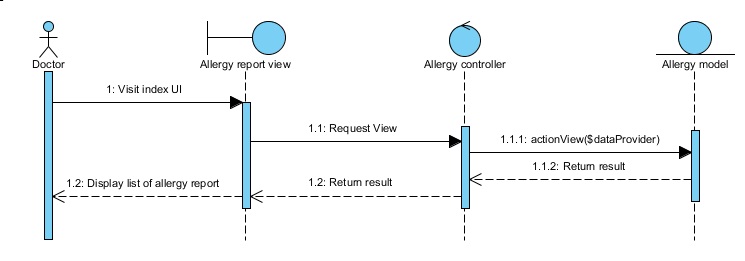
*Figure 1: Sequence Diagram of SD-01*

SD-24: Doctors can view allergy drug reports on the web application.



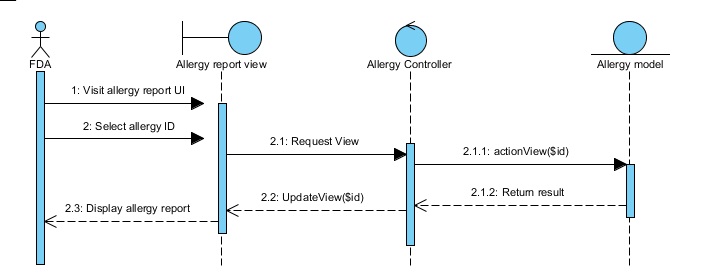
*Figure 1: Sequence Diagram of SD-01*

SD-25: Doctors can view a list of allergy drug reports on the web application.



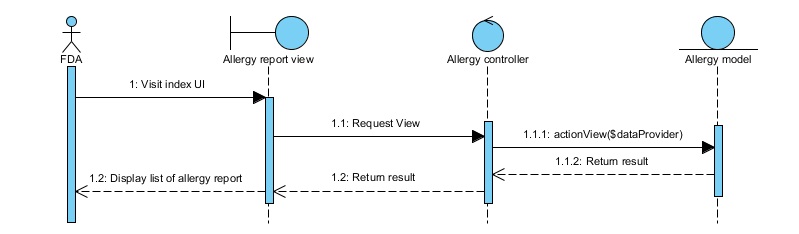
*Figure 1: Sequence Diagram of SD-01*

SD-26: FDAs can view allergy reports on the web application.



*Figure 1: Sequence Diagram of SD-01*

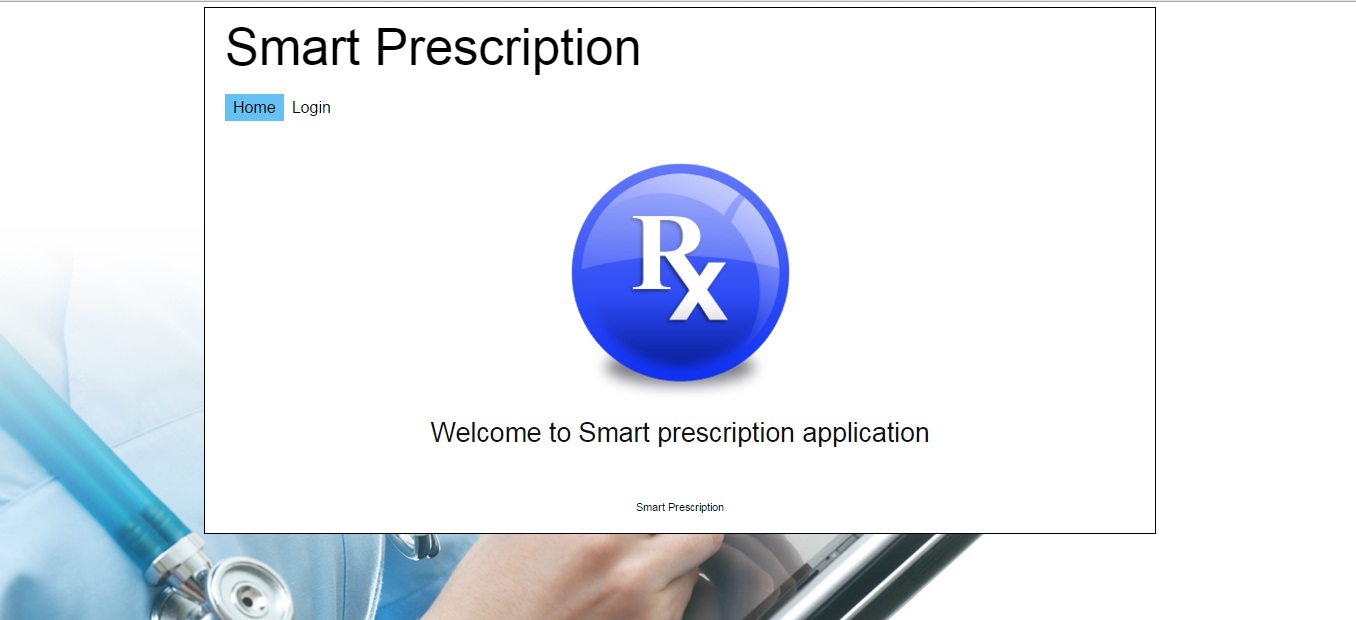
SD-27: FDAs can view a list of allergy reports on the web application.



*Figure 1: Sequence Diagram of SD-01*

# Chapter Six | User Interface Design

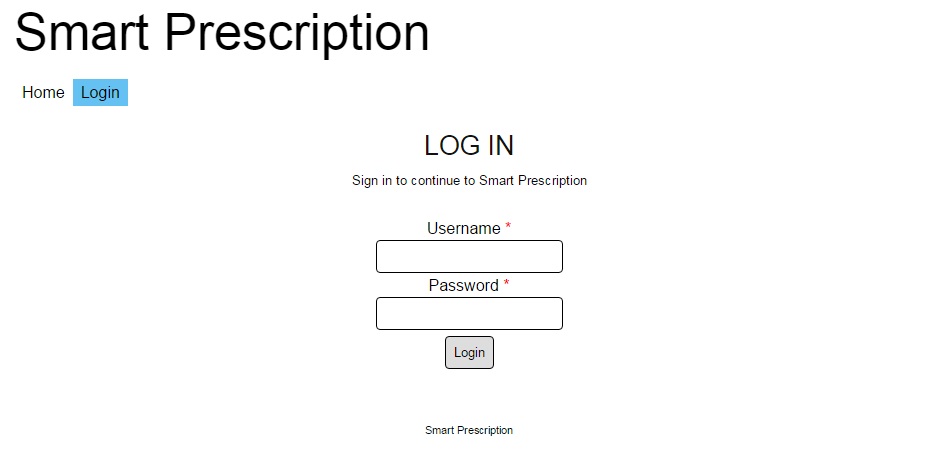
**UI-01: Home page**



*Figure 44: User Interface of “Home page”*

URS-13: Doctors, Administrations, and FDAs can login to the web application.

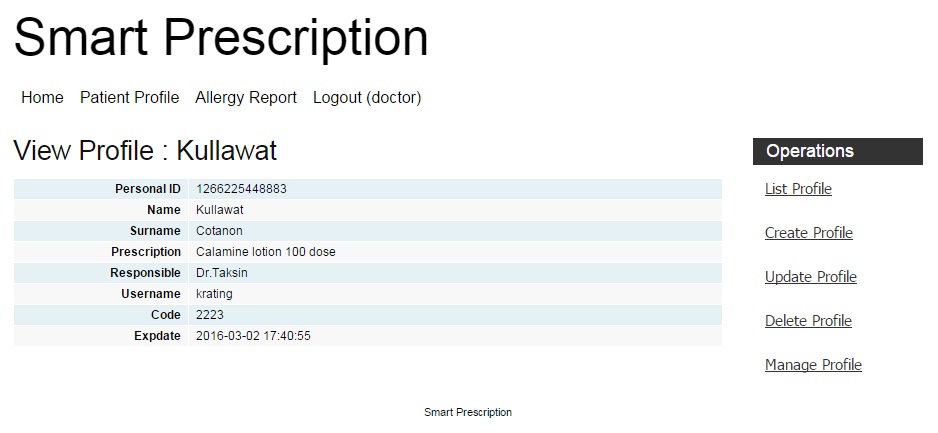
**UI-02 Login page**



*Figure 44: User Interface of “Login page”*

URS-13: Doctors, Administrations, and FDAs can login to the web application.

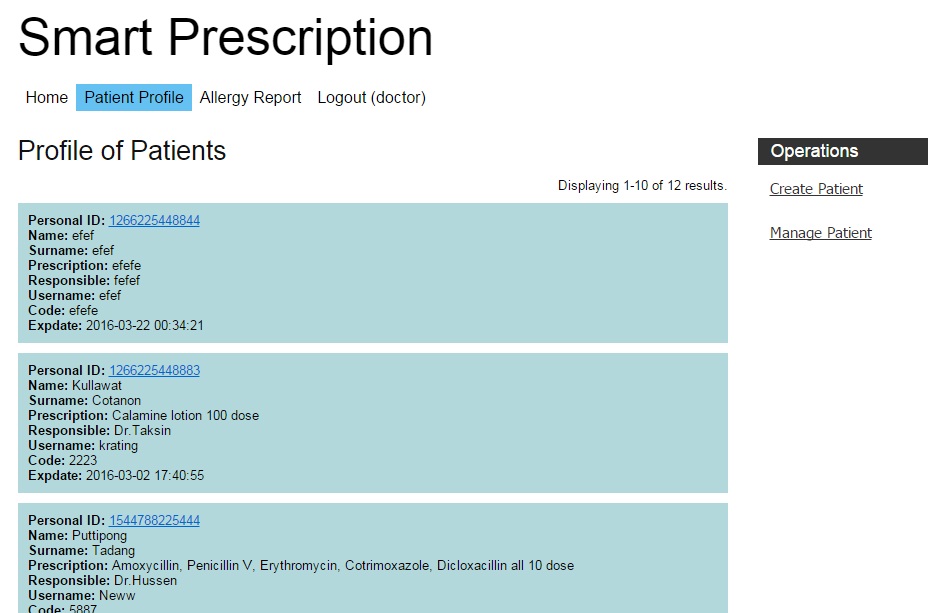
**UI-03: Patient’s profile page**



*Figure 44: User Interface of “Patient’s profile page”*

URS-5: Doctors can view patient’s profiles on the web application.

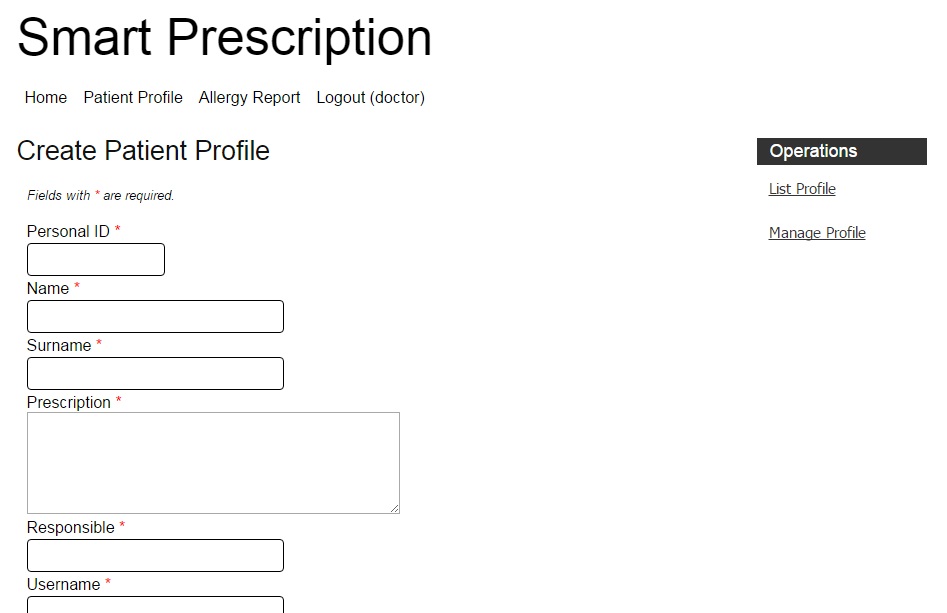
**UI-04: List of patient’s profile page**



*Figure 44: User Interface of “Patient’s profile list page”*

URS-6: Doctors can view a list of patient’s profiles on the web application.

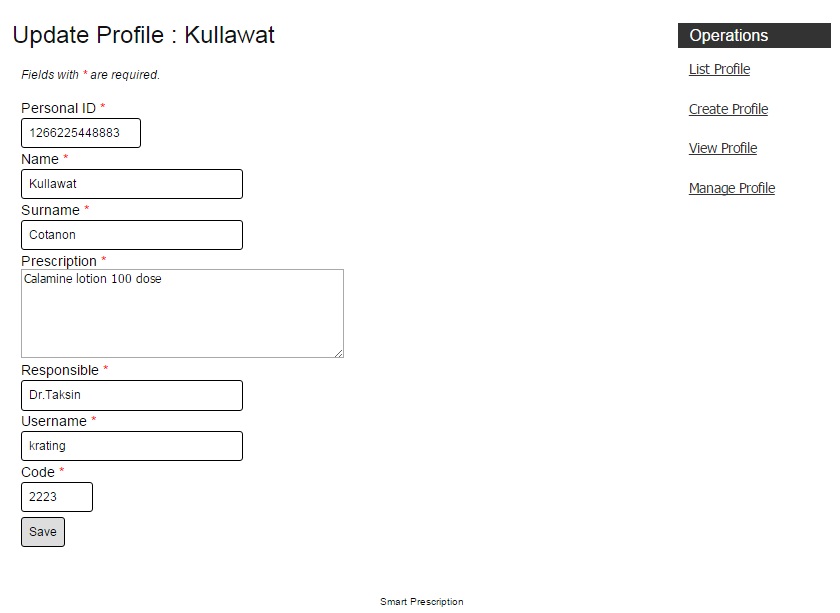
**UI-05: Create patient’s profile page**

****

*Figure 44: User Interface of “Create patient’s profile list page”*

URS-1: Doctors can create patient’s profiles on the web application.

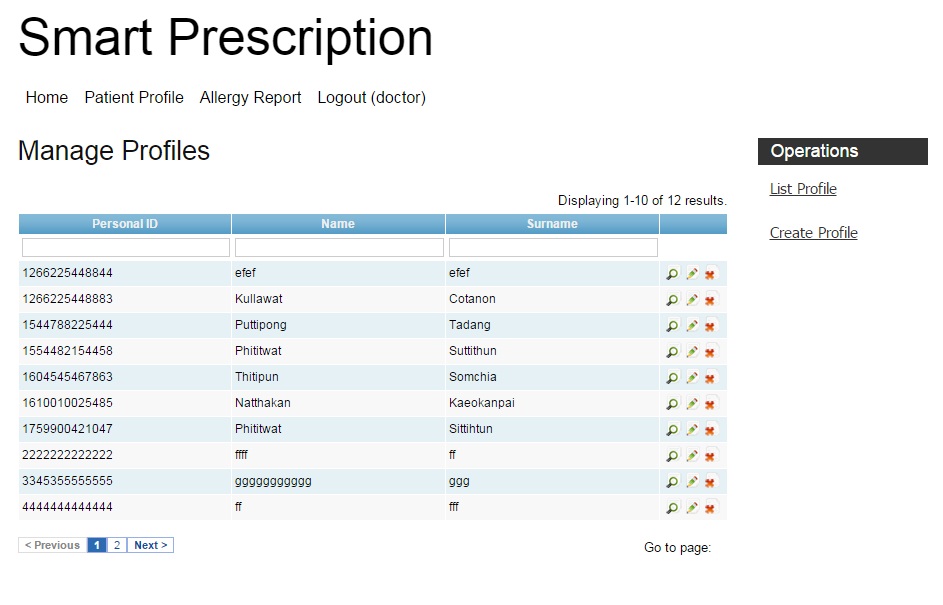
**UI-06: Update patient’s profile page**



*Figure 44: User Interface of “update patient’s profile page”*

URS-2: Doctors can update patient’s profiles on the web application.

**UI-07: Manage patient’s profile page**

****

*Figure 44: User Interface of “manage patient’s profile page”*

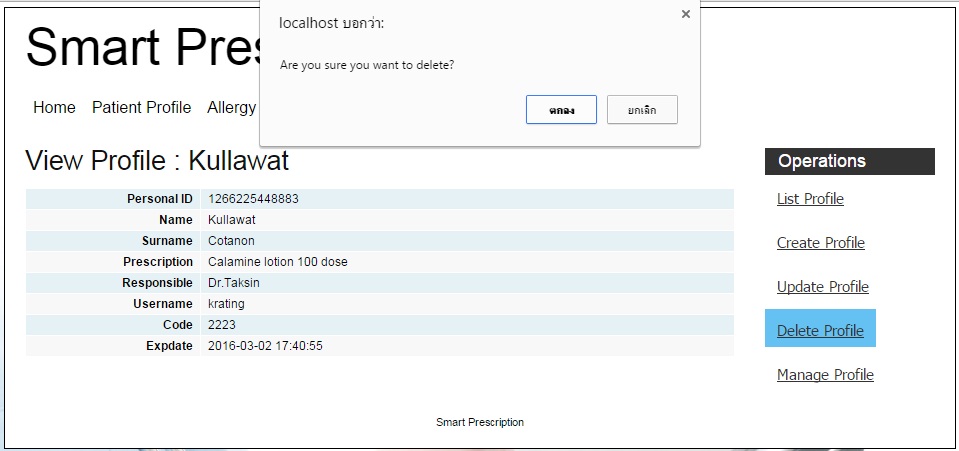
URS-2: Doctors can update patient’s profiles on the web application.

URS-3: Doctors can delete patient’s profiles on the web application.

URS-4: Doctors can search patient’s profiles on the web application.

URS-5: Doctors can view patient’s profiles on the web application.

**UI-08: Delete patient’s profile page**



*Figure 44: User Interface of “Delete patient’s profile page”*

URS-3: Doctors can delete patient’s profiles on the web application.

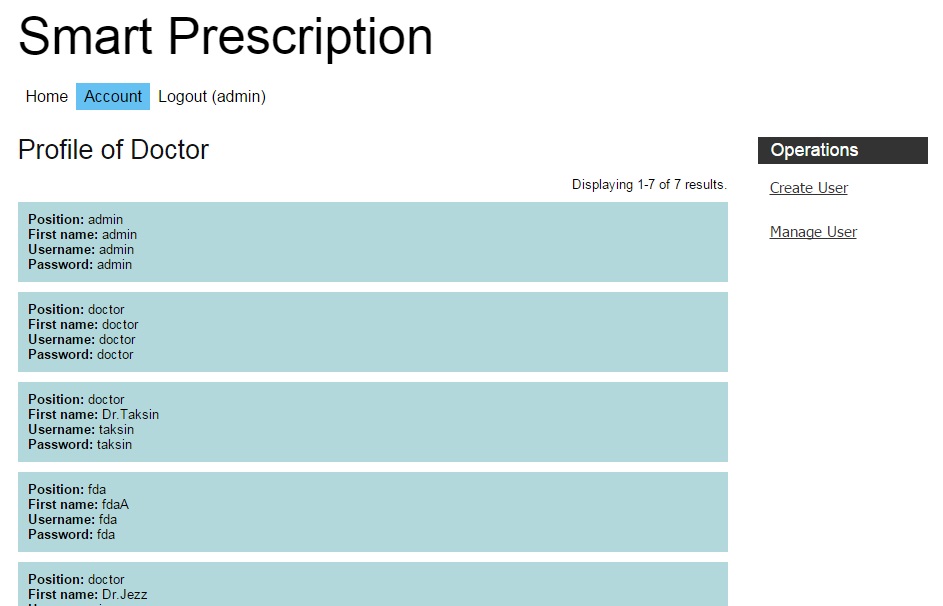
**UI-09: User’s profile page**



*Figure 44: User Interface of “user’s profile page”*

URS-11: Administrations can view user’s profiles on the web application.

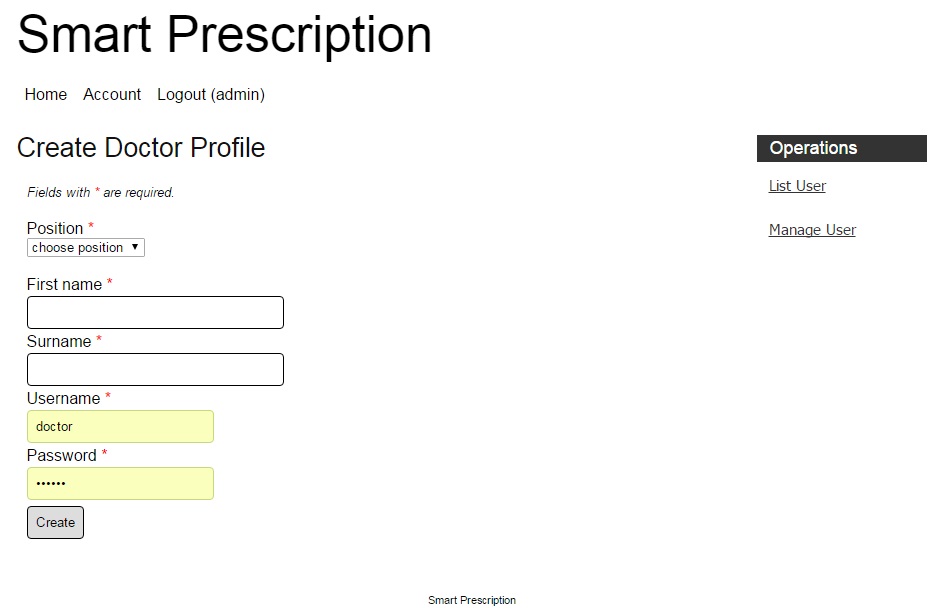
**UI-10: List of user’s profile page**



*Figure 44: User Interface of “user’s profile page”*

URS-12: Administrations can view a list of users on the web application.

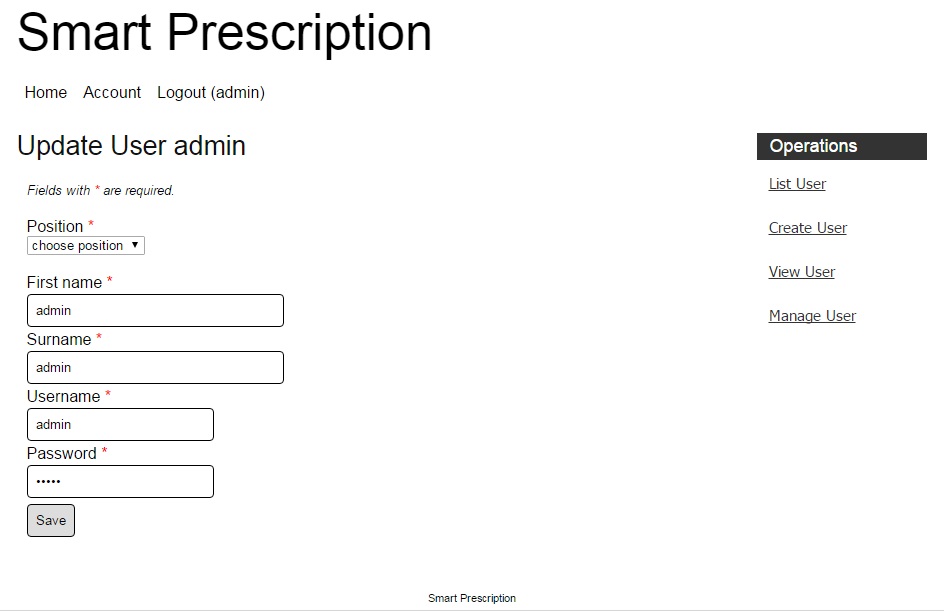
**UI-11: Create user’s profile page**



*Figure 44: User Interface of “create user’s profile page”*

URS-7: Administrations can create user’s profiles on the web application.

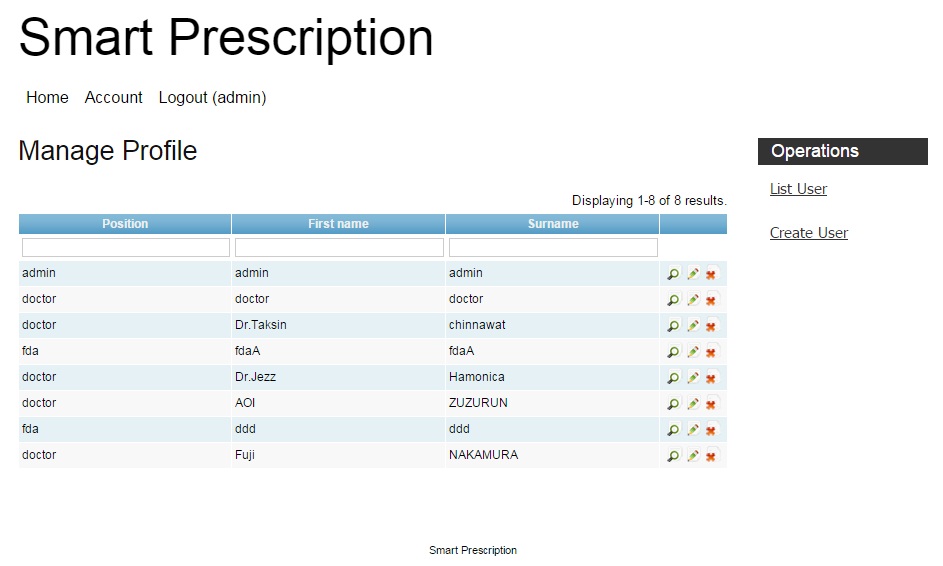
**UI-12: Update user’s profile page**



*Figure 44: User Interface of “create user’s profile page”*

URS-8: Administrations can update user’s profiles on the web application.

**UI-13: Manage user’s profile page**



*Figure 44: User Interface of “manage user’s profile page”*

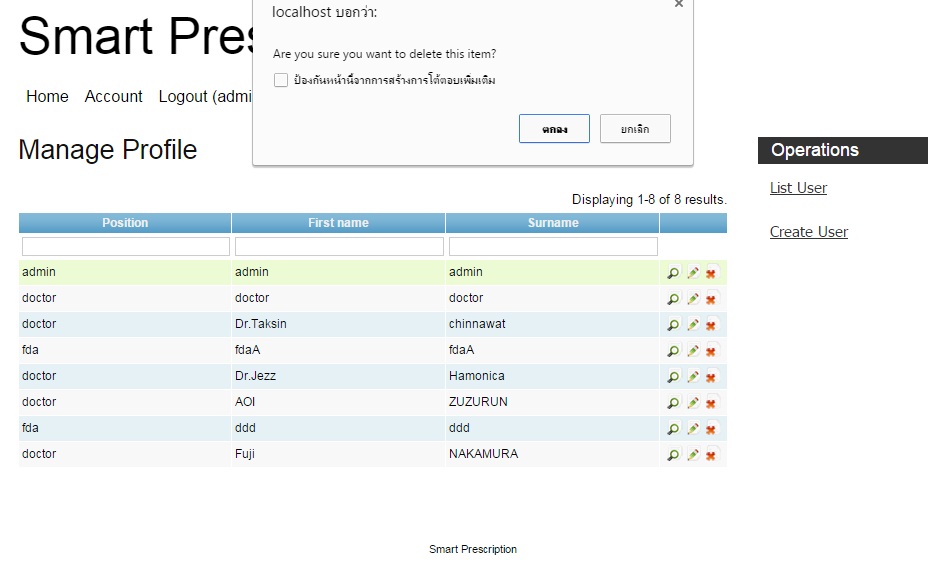
URS-8: Administrations can update user’s profiles on the web application.

URS-9: Administrations can delete user’s profiles on the web application.

URS-10: Administrations can search user’s profiles on the web application.

URS-11: Administrations can view user’s profiles on the web application

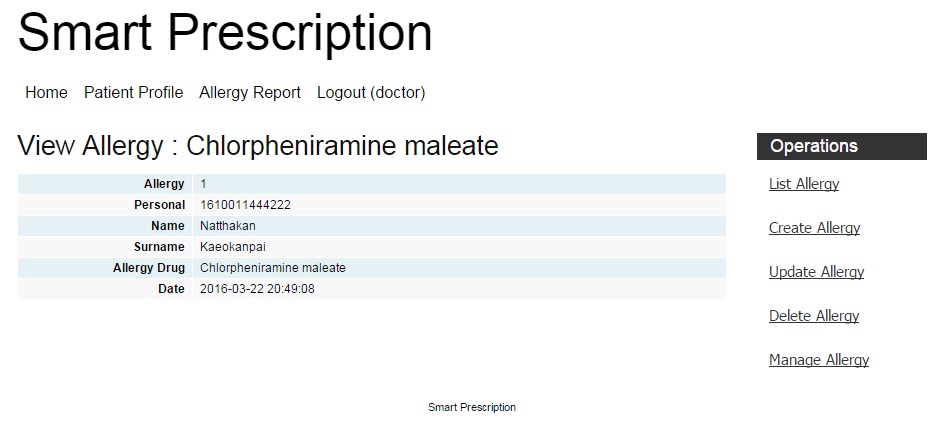
**UI-14: Delete user’s profile page**



*Figure 44: User Interface of “delete user’s profile page”*

URS-9: Administrations can delete user’s profiles on the web application.

**UI-15: Allergy report page**

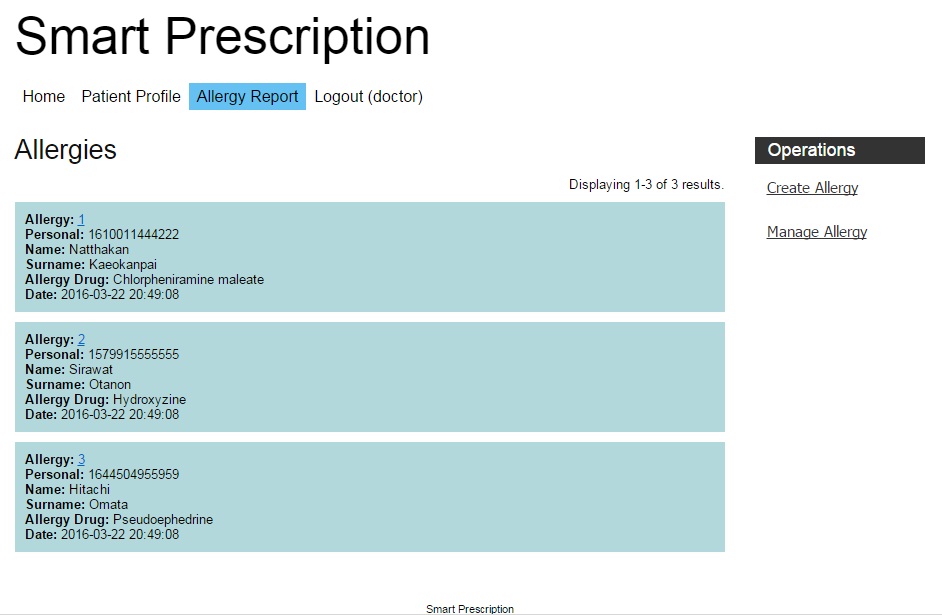


*Figure 44: User Interface of “allergy report page”*

URS-24: Doctors can view allergy drug reports on the web application.

URS-26: FDAs can view allergy reports on the web application.

**UI-16: List of allergy report page**

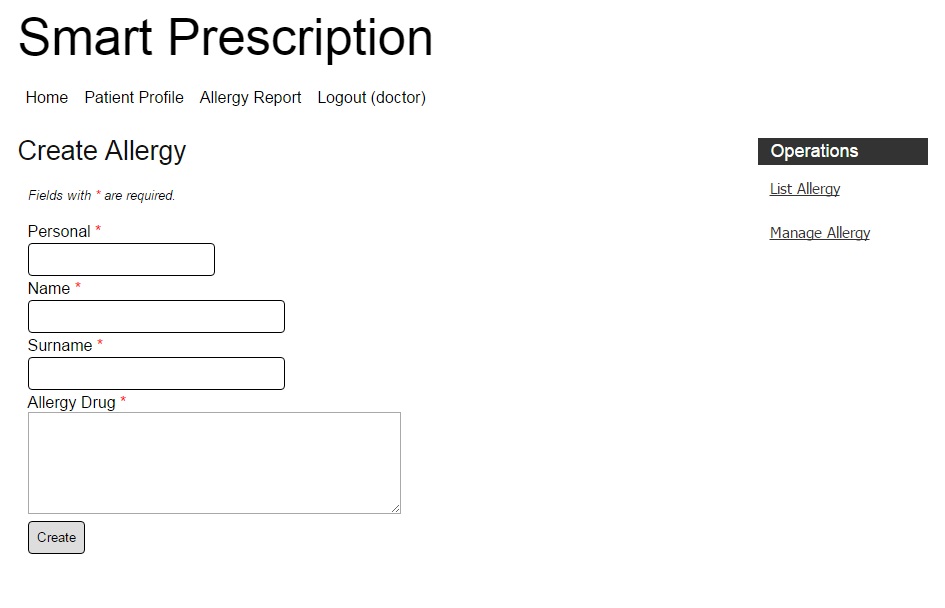


*Figure 44: User Interface of “list of allergy report page”*

URS-25: Doctors can view a list of allergy drug reports on the web application.

URS-27: FDAs can view a list of allergy reports on the web application.

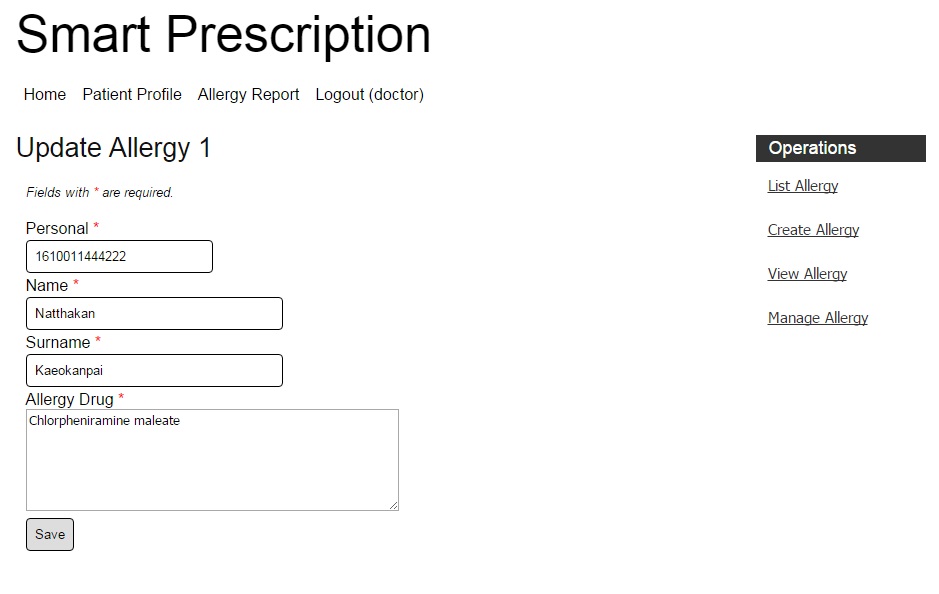
**UI-16: Create allergy report page**



*Figure 44: User Interface of “create allergy report page”*

URS-20: Doctors can create allergy reports on the web application.

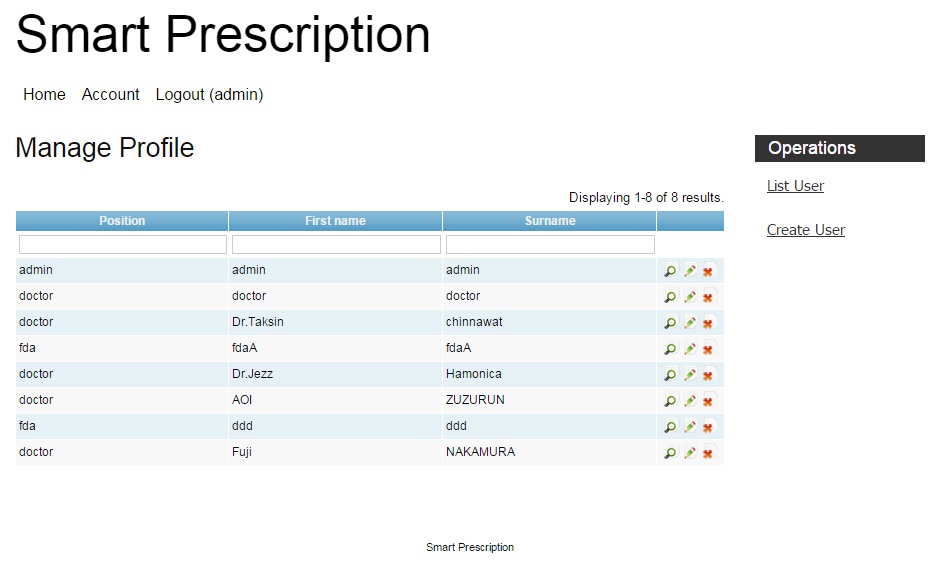
**UI-17: Update allergy report page**



*Figure 44: User Interface of “update allergy report page”*

URS-21: Doctors can update allergy reports on the web application.

**UI-18: Manage allergy report page**



*Figure 44: User Interface of “manage allergy report page”*

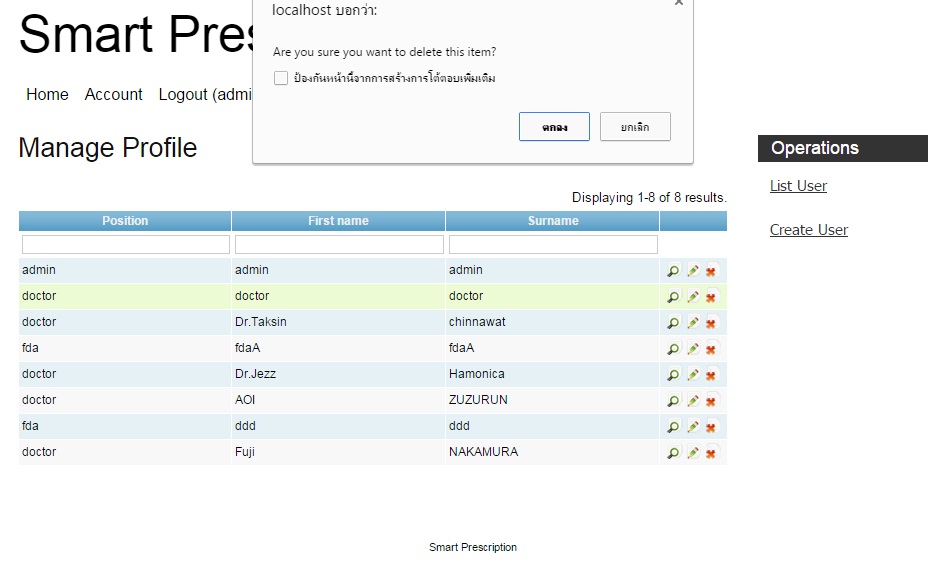
URS-21: Doctors can update allergy reports on the web application.

URS-22: Doctors can delete allergy reports on the web application.

URS-23: Doctors can search allergy reports on the web application.

URS-24: Doctors can view allergy drug reports on the web application.

**UI-19: Delete allergy report page**



*Figure 44: User Interface of “delete allergy report page”*

URS-22: Doctors can delete allergy reports on the web application.

**UI-20: Delete allergy report page**

Chapter Seven | Reference

[1] Class diagram

https://en.wikipedia.org/wiki/Class\_diagram

[2] UML

http://www.tutorialspoint.com/uml/uml\_basic\_notations.htm