AriCare: Hospital and Patients Sync Up System

Rel 1.0.0.0

Software Design Document for OO Systems

ARICENT:

Revision Number: 0.1.0.0

Contents

1. 1. Introduction 7

1.1 Purpose 7

1.2 Scope 7

1.3 Definitions and Acronyms 8

1.3.1 Definitions 8

1.3.2 Acronyms 8

1.4 References 8

1.5 Approvals and Authorizations 8

1.6 Distribution 8

1.7 Document Organization 9

1.8 Typographical Conventions 9

1. 2. Detailed Design or Lower Level Design 9

2.1 Class Detail Design 9

2.1.1 Class 1 Detail Design 9

2.1.1.1 AddDoctorServlet 9

2.1.1.2 Attributes List 9

2.1.1.3 Methods List 9

2.1.1.4 Methods 9

2.1.1.4.1 AddDoctorServlet:doPost() 10

2.1.2 Class 2 Detail Design 11

2.1.2.1 DeleteDoctorServlet 11

2.1.2.2 Attributes List 11

2.1.2.3 Methods List 11

2.1.2.3.1 DeleteDoctorServlet:doPost() 12

2.1.3 Class 3 Detail Design 20

2.1.3.1 DisplayDoctorServlet 20

2.1.3.2 Attributes List 20

2.1.3.3 Methods List 20

2.1.3.3.1 Methods 20

2.1.3.3.2 DisplayDoctorServlet:doGet() 21

2.1.4 Class 4 Detail Design 27

2.1.4.1 DoctorDetailsServlet 9

2.1.4.2 Attributes List 9

2.1.4.3 Methods List 9

2.1.4.4 Methods 9

2.1.4.4.1 DoctorDetailsServlet:doGet() 10

2.1.5 Class 5 Detail Design 27

2.1.5.1 DoctorsBySpecializationServlet 9

2.1.5.2 Attributes List 9

2.1.5.3 Methods List 9

2.1.5.4 Methods 9

2.1.5.4.1 DoctorsBySpecializationServlet:doGet() 10

2.1.6 Class 6 Detail Design 27

2.1.6.1 AppointmentServlet 9

2.1.6.2 Attributes List 9

2.1.6.3 Methods List 9

2.1.6.4 Methods 9

2.1.6.4.1 AppointmentServlet:doPost() 10

2.1.7 Class 7 Detail Design 27

2.1.7.1 LoginServlet 9

2.1.7.2 Attributes List 9

2.1.7.3 Methods List 9

2.1.7.4 Methods 9

2.1.7.4.1 LoginServlet:doPost() 10

2.1.8 Class 8 Detail Design 27

2.1.8.1 SignUpServlet 27

2.1.8.2 Attributes List 27

2.1.8.3 Methods List 27

2.1.8.4 Methods 9

2.1.8.4.1 SignUpServlet:doPost() 10

2.1.9 Class 9 Detail Design 27

2.1.9.1 PatientProfileServlet 27

2.1.9.2 Attributes List 27

2.1.9.3 Methods List 27

2.1.9.4 Methods 9

2.1.9.4.1 PatientProfileServlet:doGet() 10

2.1.10 Class 10 Detail Design 27

2.1.10.1 DisplayNewsServlet 27

2.1.10.2 Attributes List 27

2.1.10.3 Methods List 27

2.1.10.4 Methods 9

2.1.10.4.1 DisplayNewsServlet:doGet() 10

2.1.11 Class 11 Detail Design 27

2.1.11.1 PillDisplayServlet 27

2.1.11.2 Attributes List 27

2.1.11.3 Methods List 27

2.1.11.4 Methods 9

2.1.11.4.1 PillDisplayServlet:doGet() 10

2.1.12 Class 12 Detail Design 27

2.1.12.1 UpdatePillDetailsServlet 27

2.1.12.2 Attributes List 27

2.1.12.3 Methods List 27

2.1.12.4 Methods 9

2.1.12.4.1 UpdatePillDetailsServlet:doPost() 10

2.1.13 Class 13 Detail Design 27

2.1.13.1 MailTextCreator 27

2.1.13.2 Attributes List 27

2.1.13.3 Methods List 27

2.1.13.4 Methods 9

2.1.13.4.1 MailTextCreator:textGenerator() 10

2.1.14 Class 14 Detail Design 27

2.1.14.1 MailListCreator 27

2.1.14.2 Attributes List 27

2.1.14.3 Methods List 27

2.1.14.4 Methods 9

2.1.14.4.1 MailListCreator:listcreator() 10

2.1.15 Class 15 Detail Design 27

2.1.14.1 MailSimulator 27

2.1.14.2 Attributes List 27

2.1.14.3 Methods List 27

2.1.14.4 Methods 9

2.1.14.4.1 MailSimulator:mailCredentials() 10

2.1.16 Class 16 Detail Design 27

2.1.16.1 ConnectionManager 27

2.1.16.2 Attributes List 27

2.1.16.3 Methods List 27

2.1.16.4 Methods 9

2.1.16.4.1 ConnnectionManager:getConnection() 10

2.1.17 Class 17 Data Detail Design 27

2.1.17.1 Data Entity 1 Detail Design 28

2.1.17.2 Data Entity 2 Detail Design 28

2.1.17.3 Data Entity 3 Detail Design 29

2.1.17.4 Data Entity 4 Detail Design 29

2.1.17.5 Data Entity 5 Detail Design 30

2.1.18 ScreenShots 31

2.1.18.1 Home Page 31

2.1.18.2 About Page 32

2.1.18.3 Doctors Page 33

2.1.18.4 Specialization Page 34

2.1.18.5 Services Page 35

2.1.18.6 News Page 36

2.1.18.7 Login Page 37

2.1.18.8 Sign Page 38

2.1.18.9 Admin Page 39

2.1.18.10 Add Doctor Page(admin) 40

2.1.18.11 Delete Doctor Page(admin) 41

2.1.18.12 Update Pill Details Page(admin) 42

2.1.18.13 Add News Page(admin) 43

2.1.18.14 Patient Logged in Page 44

2.1.18.15 Patient Profile Page 45

2.1.18.16 Book Appointment Page 46

2.1.18.17 Medicine Details Page 47

2.1.18.18 Appointment History Page 48

2.1.18.19 Doctor Page 49

2.1.18.20 Doctor Profile Page 50

2.1.18.21 Doctor Today's Appointment Page 50

2.1.18.22 Logout Page 50

2.2 Configuration and Control 50

2.2.1 Overview 50

2.2.2 Startup 50

2.2.3 Shutdown 50

2.2.4 Creation of patientHome New Entries 50

2.2.5 Creation of patient New Entries 50

2.2.6 Creation of doctor New Entries 50

2.2.7 Creation of AppointmentDetails New Entries 51

2.2.8 Creation of a new Specialization 51

2.2.9 Deletion of Patient Entries 51

2.2.10 Deletion of Doctor Entries 51

2.2.11 Changes to Standard MIB 51

2.3 Logging: 51

2.4 Tracing 51

2.5 Error Handling 51

2.5.1 System Errors 51

2.5.2 Interface Errors 51

2.5.3 Protocol Errors 52

1. Design Entity Attributes 53

Identification: DFD 53

Identification: ER Diagram 53

Identification: Pseudocode 53

Identification: PDL 53

Template Revision History 53

Tables

[Table 1‑1: Definitions Used in this Document 2](#_Toc165179715)

[Table 1‑2: Acronyms Used in this Document 2](#_Toc165179716)

[Table 1‑3: Approvals and Authorizations 2](#_Toc165179717)

[Table 1‑4: Distribution 2](#_Toc165179718)

Chapter

1

# Introduction

## Purpose

* Purpose of the software is to provide Patient and Hospital Sync up System for a hospital. This system will enable to book appointments, view their pills and the dosage. This system will also notify patients 1 hour before scheduled pill time and 1 day before the appointment.
* Purpose of the document is to provide high- level design for the system.
* Intended audience of the software is various patients who will register themselves for the product.
* Intended audience of this document is the development team.

## Scope

Name of the software: AriCare

System will do the following things:

1. Enable new patients to register.
2. Enable patients to book appointment.
3. Enable patients to view their pill details;
4. Enable patients to view their appointment history.
5. Automatically send pills and appointment notification to patients.
6. Enable doctors to view their today’s patient appointment details.
7. Enable admin to add and remove doctors, add news and add pill details.

## Definitions and Acronyms

### Definitions

Table ‑: Definitions Used in this Document

|  |  |
| --- | --- |
| DFD | It is a diagram which illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored. |
| SSADM | It is a set of standards developed in the early 1980s for Systems Analysis and Application Design which is widely used for government computing projects in the United Kingdom |
|  |  |

### Acronyms

Table ‑: Acronyms Used in this Document

|  |  |
| --- | --- |
| Acronym | Explanation |
| DFD | Data Flow Diagram |
| SSADM | Structured Systems Analysis And Design Method |
|  |  |

## References

N/A

## Approvals and Authorizations

N/A

## Distribution

Table ‑3: Distribution

|  |  |  |
| --- | --- | --- |
| **Copy No.** | **Holder’s Designation** | **Issue Date** |
| 1 | Naveen Kumar(Mentor) | 28/04/2016 |
| 2 | Aayush Chawla | 28/04/2016 |
| 3 | Keshav Batra | 28/04/2016 |
| 4 | Mandeep Kaur | 28/04/2016 |
| 5 | Sakshi Kansal | 28/04/2016 |
| 6 | Sukanya Sehgal | 28/04/2016 |
|  |  |  |

## Document Organization

Software Design Document is organized in the following sections:

* Introduction: Purpose of this section is to introduce reader to the system. Explain various terms and acronyms used in the documents and also explain scope of the system.
* High level Design: Purpose of this section is to present high level design of the system via DFDs, Architectural design descriptions and class descriptions.
* Alternate Design: Purpose of this section is to present alternative design of the system to the reader using its scope, requirements and advantages and disadvantages.
* Low Level Design: Purpose of this section is to give the internal logical design of the actual program code.

## Typographical Conventions

* The convention used in this document is **SSADM** (for DFDs).

Chapter

2

# Low Level Design

## Class Detail Design

### Class 1 Detail Design

#### AddDoctorServlet

This class is used to add a doctor (doctor name, phone, address, qualification, timings, specialization and picture) to the doctor table. This functionality is performed by the admin.

#### Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doPost();

#### Methods

##### AddDoctorServlet::doPost()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to add a doctor to the database by making and using objects of the corresponding Bean and DAO classes- AddDoctorBean, AddDoctorDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| request | HttpServletRequest | Represents the request sent by client to the server. |
| response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| N/A |  |  |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the functionality to add a doctor to the databse.

### Class 2 Detail Design

#### DeleteDoctorServlet

This class is used to delete a doctor from the doctor table. This functionality is performed by the admin.

#### Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doPost();

#### Methods

##### DeleteDoctorServlet::doPost()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to delete a doctor from the doctor table by making and using objects of the corresponding Bean and DAO classes- DeleteDoctorBean, DeleteDoctorDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| request | HttpServletRequest | Represents the request sent by client to the server. |
| response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| N/A |  |  |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the functionality to delete a doctor from the databse.

### Class 3 Detail Design

2.1.3.1 DisplayDoctorServlet

This class is used to display doctor list from doctor\_profile table. This functionality is performed by a patient currently logged in.

#### 2.1.3.2 Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doGet();

#### Methods

##### DisplayDoctorServlet::doGet()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to display doctor list using objects of the corresponding Bean and DAO classes- AddDoctorBean, DisplayDoctorDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| Request | HttpServletRequest | Represents the request sent by client to the server. |
| Response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the functionality to display list of all doctors from the database.

### Class 4 Detail Design

#### 2.1.4.1 DoctorDetailsServlet

This class is used to display the details of a particular doctor. This functionality is performed by a patient currently logged in.

2.1.4.2 Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doGet();

#### Methods

##### DoctorDetailsServlet::doGet()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to to display the details of a particular doctor using objects of the corresponding Bean and DAO classes- AddDoctorBean AddDoctorDetailsDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| Request | HttpServletRequest | Represents the request sent by client to the server. |
| Response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| N/A |  |  |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the functionality to display dtails of particular doctor from the database.

### Class 5 Detail Design

#### 2.1.5.1 DoctorsBySpecializationServlet

This class is used to display list of doctors based on their specialization. This functionality is performed by a patient currently logged in.

2.1.5.2 Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doGet();

#### Methods

##### DoctorsBySpecializationServlet::doGet()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to display doctor list based on their specialization using objects of the corresponding Bean and DAO classes- AddDoctorBean, DoctorsBySpecializationDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| Request | HttpServletRequest | Represents the request sent by client to the server. |
| Response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the functionality to display list of all doctors based on their specialization from the database.

### Class 6 Detail Design

#### 2.1.6.1 AppointmentServlet

This class is used to add an appointment entry to the database. This functionality is performed by a patient currently logged in.

#### Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doPost();

#### Methods

##### AppointmentServlet::doPost()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to add an appointment entry to the database using objects of the corresponding Bean and DAO classes- AppointmentBean, AppointmentDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| request | HttpServletRequest | Represents the request sent by client to the server. |
| response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| N/A |  |  |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the functionality to add an appointment entry to the database.

### Class 7 Detail Design

#### LoginServlet

This class is used to make a doctor/patient/admin login the application. It authenticates the credentials entered by the person who is logging in.

#### Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doPost();

#### Methods

##### LoginServlet::doPost()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to make a doctor/patient/admin login the application. It authenticates the credentials entered by the person who is logging in. To achieve this functionality it uses objects of the corresponding Bean and DAO classes- LoginBean, LoginDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| request | HttpServletRequest | Represents the request sent by client to the server. |
| response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| N/A |  |  |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the login functionality for doctor, patient and admin.

### Class 8 Detail Design

#### SignupServlet

This class is used to make a new patient sign up the application. It enters the details of the person, entered for signing up for the application, to the database.

#### Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doPost();

#### Methods

##### LoginServlet::doPost()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to make a new patient sign up the application. It enters the details of the person, entered for signing up for the application, to the database. To achieve this functionality it uses objects of the corresponding Bean and DAO classes- SignupBean, SignupDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| request | HttpServletRequest | Represents the request sent by client to the server. |
| response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| N/A |  |  |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the login functionality for doctor, patient and admin.

### Class 9 Detail Design

#### PatientProfileServlet

This class is used to display details of a particular patient. This functionality is performed by a patient currently logged in.

Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doGet();

#### Methods

##### PatientProfileServlet::doGet()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to display the details of a particular patient who is already registered using objects of the corresponding Bean and DAO classes-PatientDetailsBean PatientProfileDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| Request | HttpServletRequest | Represents the request sent by client to the server. |
| Response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| N/A |  |  |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the functionality to display details of particular patient from the database.

### Class 10 Detail Design

#### DisplayNewsServlet

This class is used to display news to all the registered patients.

Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doGet();

#### Methods

##### DisplayNewsServlet::doGet()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to display news list to the registered patients using objects of the corresponding Bean and DAO classes- AddNewsBean, DisplayNewsDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| Request | HttpServletRequest | Represents the request sent by client to the server. |
| Response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the functionality to display the list of news from the database.

### Class 11 Detail Design

#### PillDisplayServlet

This class is used to display pill details (pill name, start date, end date and dosage time)to the patient.

Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doGet();

#### Methods

##### PillDisplayServlet::doGet()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to display pill details to the registered patients using objects of the corresponding Bean and DAO classes- PillDetailsBean, DisplayPillDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| Request | HttpServletRequest | Represents the request sent by client to the server. |
| Response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides the functionality to display the list of news from the database.

### Class 12 Detail Design

#### UpdatePillDetailsServlet

This class is used to update the database for pills prescribed to a patient. This functionality is used by the admin.

#### Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

doPost();

#### Methods

##### PillDetailsServlet::doPost()

**Method type:** void [return type]

HttpServletRequest [parameter 1]

HttpServletResponse [parameter 2]

**Description:**

This method is used to update the database for pills prescribed to a patient. This functionality is used by the admin.

To achieve this functionality this method uses objects of the corresponding Bean and DAO classes- UpdatePillDetailsBean, UpdatePillDetailsDAO.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| request | HttpServletRequest | Represents the request sent by client to the server. |
| response | HttpServletResponse | Represents the request sent by server to the client. |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| N/A |  |  |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| NA |  |  |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

This method is used to update the database for pills prescribed to a patient.

### Class 13 Detail Design

#### 2.1.13.1 MailTextCreator

This class is used to create content of mail to be sent to patients regarding their appointment meeting or pill intake timings or news.

#### Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

textGenerator();

#### Methods

##### MailTextCreator::textGenerator()

**Method type:** void [return type]

**Description:** This method adds the content to be sent to patient in an araylist.

The functionality of this method is used by the MailSimulator class.

### Class 14 Detail Design

#### 2.1.14.1 MailListCreator

This class is used to create list of email ids of patients to whom mail is to be sent.

#### 2.1.14.2 Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

listCreator();

#### Methods

##### MailListCreator::listCreator()

**Method type:** void [return type]

**Description:** This method adds the email ids of the patients in an araylist.

The functionality of this method is used by the MailSimulator class.

### Class Detail Design

#### 2.1.15.1 MailSimulator

This class uses the functionality of the MailTextCreator and MailListCreator class to send mail to the registered patients.

#### Attributes List

**private:**

serialVersionUID

#### Methods List

**public:**

mailCredentials();

#### Methods

##### MailSimulator::mailCredentials()

**Method type:** void [return type]

**Description:** This method is invoked by the MailInvoker class at fixed interval of time for the mail sending process.

### Class 16 Detail Design

#### ConnectionManager

This class is used to establish a connection with the database server.

#### Attributes List

**public:**

connection

url

#### Methods List

**public:**

getConnection();

#### Methods

#### ConnectionManager::getConnection()

**Method type:** Conection [return type]

**Description:**

This method is to establish a connection with the database server and return the object of Connection class.

**Input parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| N/A |  |  |

**Output parameters:**

Detail the Output parameters that are modified within the method.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Purpose |
| NA |  |  |

**Return Values:**

|  |  |  |
| --- | --- | --- |
| Return Values | Type | Purpose |
| connection | Connection | The object of Connection class, which establishes a connection with the database server. |

**Exceptions:**

SQLException

IOException

**Pseudo Code or PDL (Program Design Language)**

It provides a connection with the database server.

### Data Detail Design

Name of database: hospital\_management

Tables:

login\_details

pill\_details

appointment\_details

patient\_profile

doctor\_profile

#### Data Entity 1 Detail Design

**login\_details:**

It is used to provide login credentials in order to login

phone\_number: bigint (primary key)

This attribute stores the phone number of a user.

password: varchar

This attribute stores the password of a user.

role: varchar

This option stores the role of a user (patient/doctor/admin).

#### Data Entity 2 Detail Design

**pill\_details:**

It stores the details of pills prescribed to various patients by doctors.

pill\_id: int (primary key)

This is a unique id auto-generated to distinguish each pill.

doctor\_id: int (foreign key referencing primary key of doctor\_profile)

This attribute stores id of the doctor who prescribed this pill.

patient\_id: int (foreign key referencing primary key of patient\_profile)

This attribute stores id of the patient who has been prescribed this pill.

pill\_name: varchar

This attribute stores name of the pill.

start\_date: varchar

This attribute stores start date of the pill.

end\_date: varchar

This attribute stores end date of the pill.

dosage\_time: time

This attribute stores dosage time of the pill.

dosage: int

This attribute stores the dosage for the pill.

#### Data Entity 3 Detail Design

**appointment\_details:**

It is used to store details about appointments corresponding to different doctors and patients.

appointment\_id: int (primary key)

This is a unique id auto-generated to distinguish each appointment.

patient\_id: int (foreign key referencing primary key of patient\_profile)

This attribute stores id of the patient who has been prescribed this pill.

doctor\_id: int (foreign key referencing primary key of doctor\_profile)

This attribute stores id of the doctor who prescribed this pill.

patient\_name: varchar

This attribute stores name of the patient who took this appointment.

doctor\_name: varchar

This attribute stores name of the doctor for whom this appointment has been taken.

timing: time

This attribute stores the timing of this appointment.

date: date

This attribute stores the date for which this appointment has been booked.

#### Data Entity 4 Detail Design

**patient\_profile:**

It is used to store details of all the patients registered with this hospital application.

patient\_id: int (primary key)

This is a unique id auto-generated to distinguish each patient.

patient\_name: varchar

This attribute stores name of the patient to whom this profile belongs.

phone\_number: bigint (foreign key referencing primary key of login\_details)

This attribute stores phone number of the patient to whom this profile belongs.

address: varchar

This attribute stores address of the patient to whom this profile belongs.

email\_id: varchar

This attribute stores email id of the patient to whom this profile belongs.

#### Data Entity 5 Detail Design

**doctor\_profile:**

It is used to store details of all the doctors registered with this hospital application.

doctor\_id: int (primary key)

This is a unique id auto-generated to distinguish each doctor.

doctor\_name: varchar

This attribute stores name of the doctor to whom this profile belongs.

phone\_number: bigint (foreign key referencing primary key of login\_details)

This attribute stores phone number of the doctor to whom this profile belongs.

address: varchar

This attribute stores address of the doctor to whom this profile belongs.

qualification: varchar

This attribute stores qualification of the doctor to whom this profile belongs.

timings: varchar

This attribute stores timings of the doctor to whom this profile belongs.

specialization: varchar

This attribute stores specialization of the doctor to whom this profile belongs.

image: varchar

This attribute stores image of the doctor to whom this profile belongs.

### Screen Shots

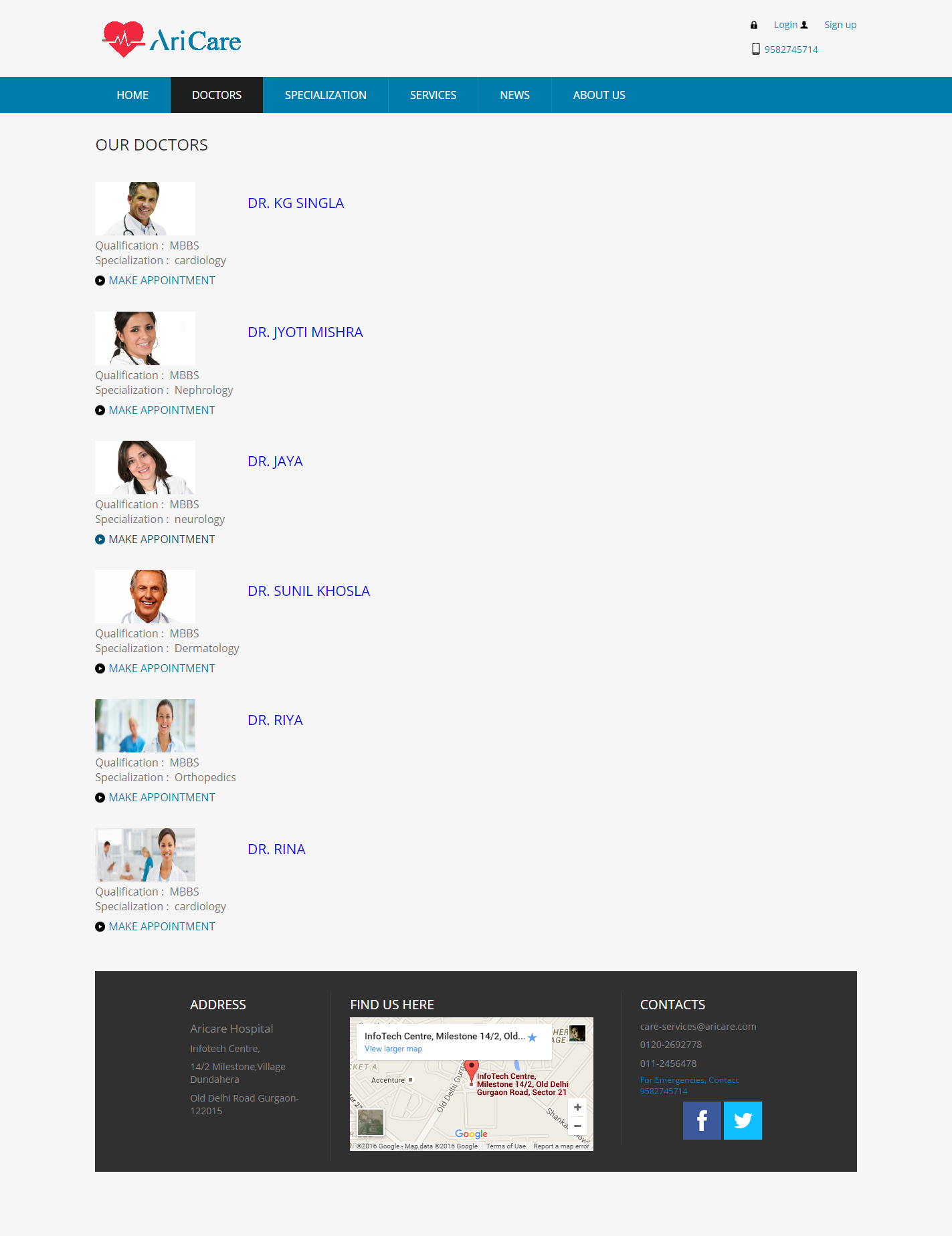
#### Home page



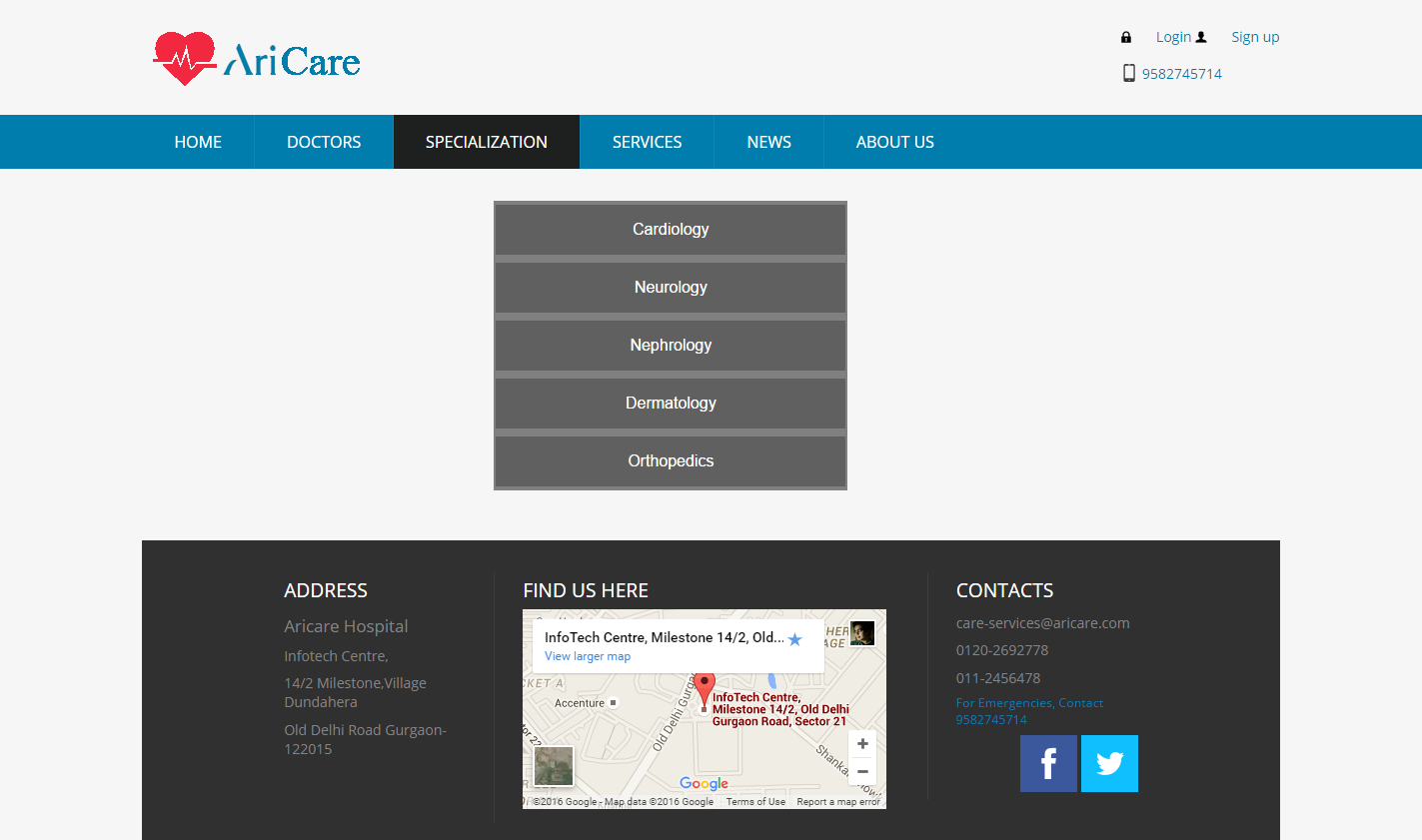
#### About Page

#### C:\Users\gur44629\Downloads\ScreenshotsPagesProject\ScreenshotsPagesProject\About.png

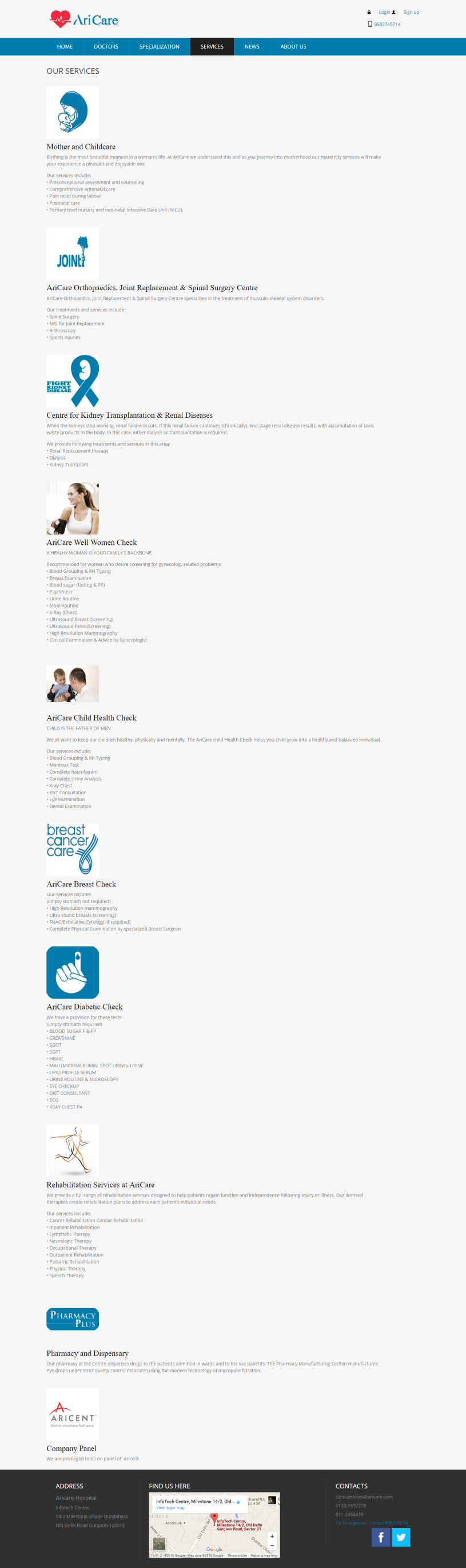
#### Doctors page



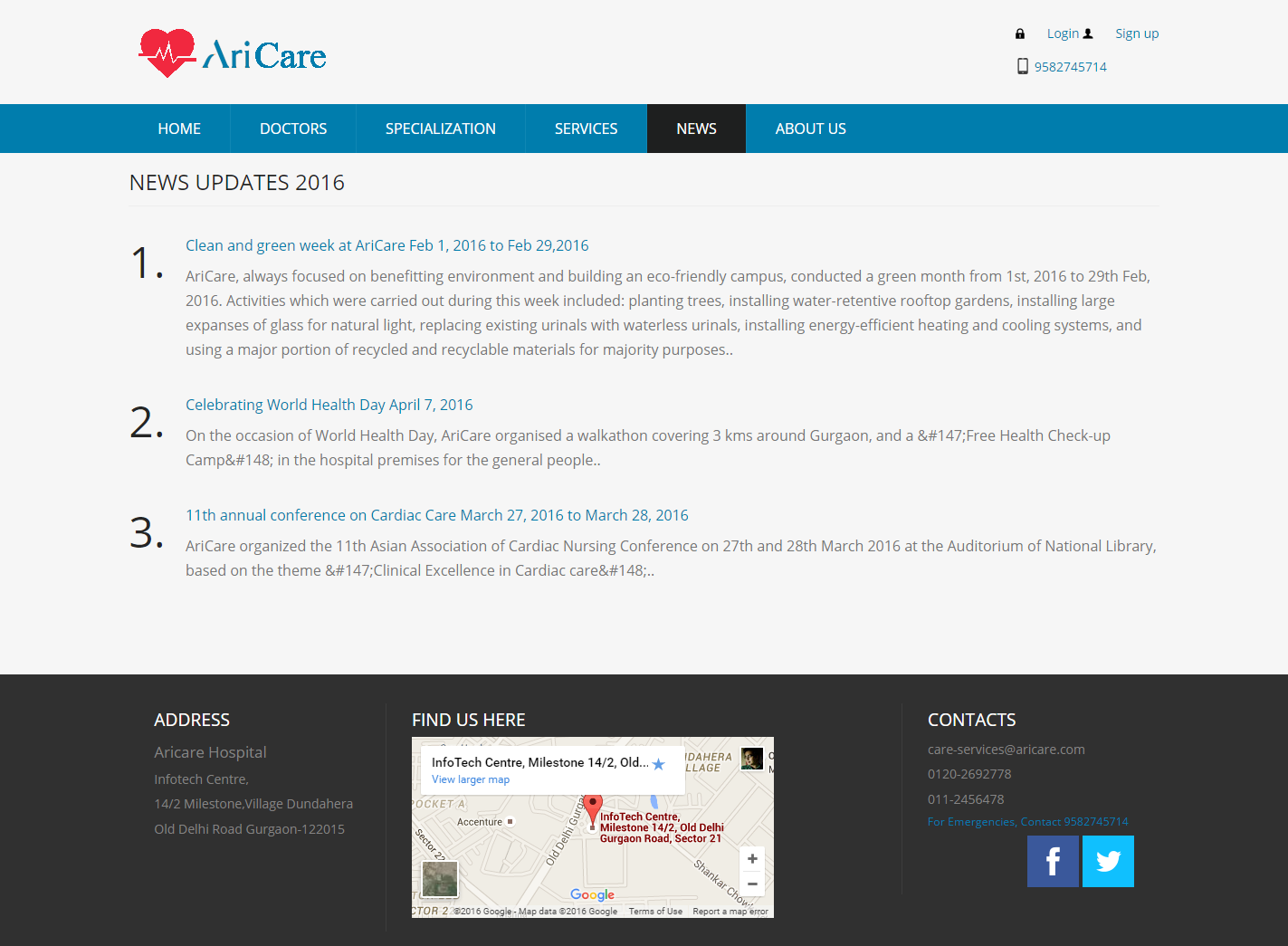
#### Specialization page



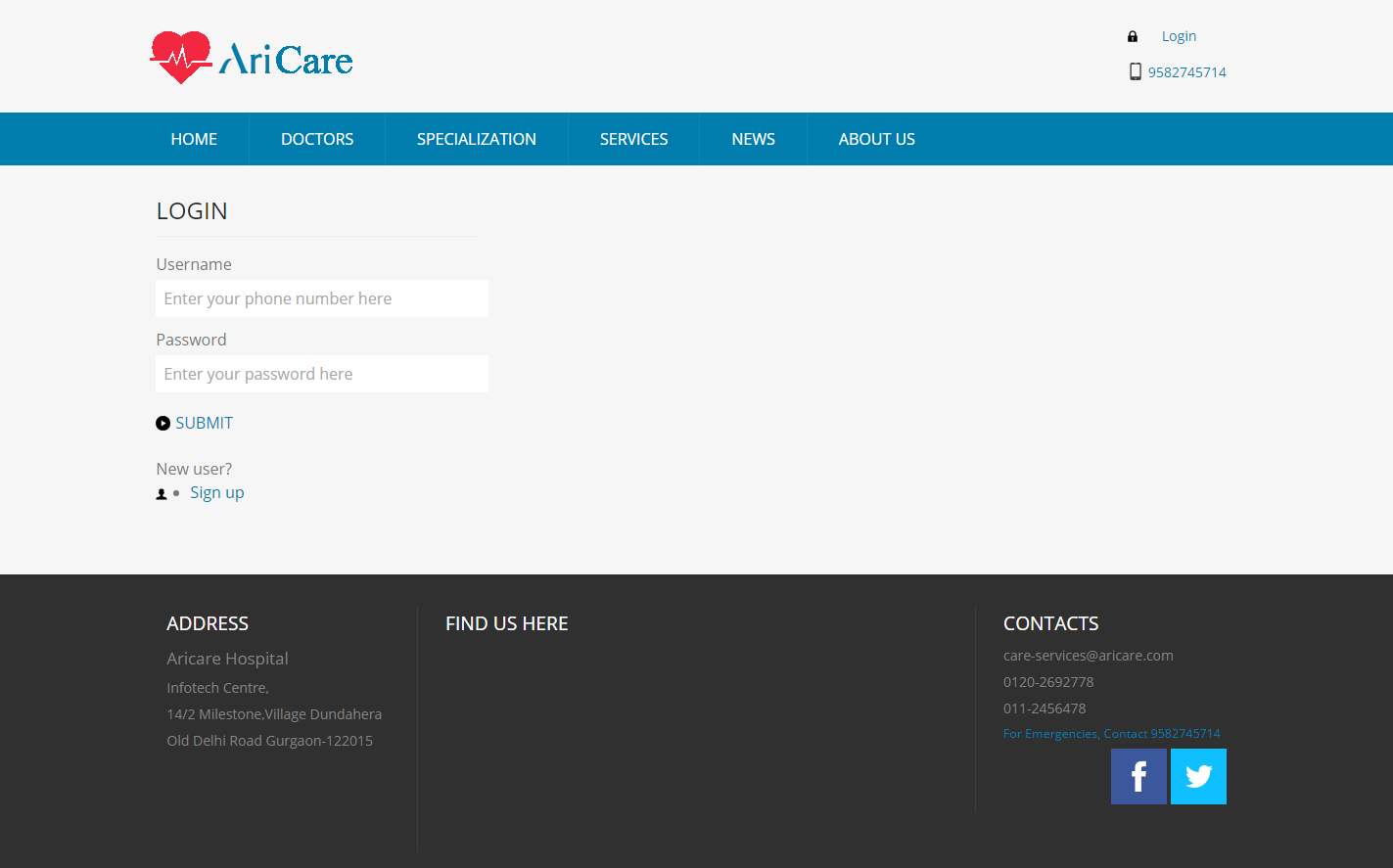
#### Services page

****

#### News Page

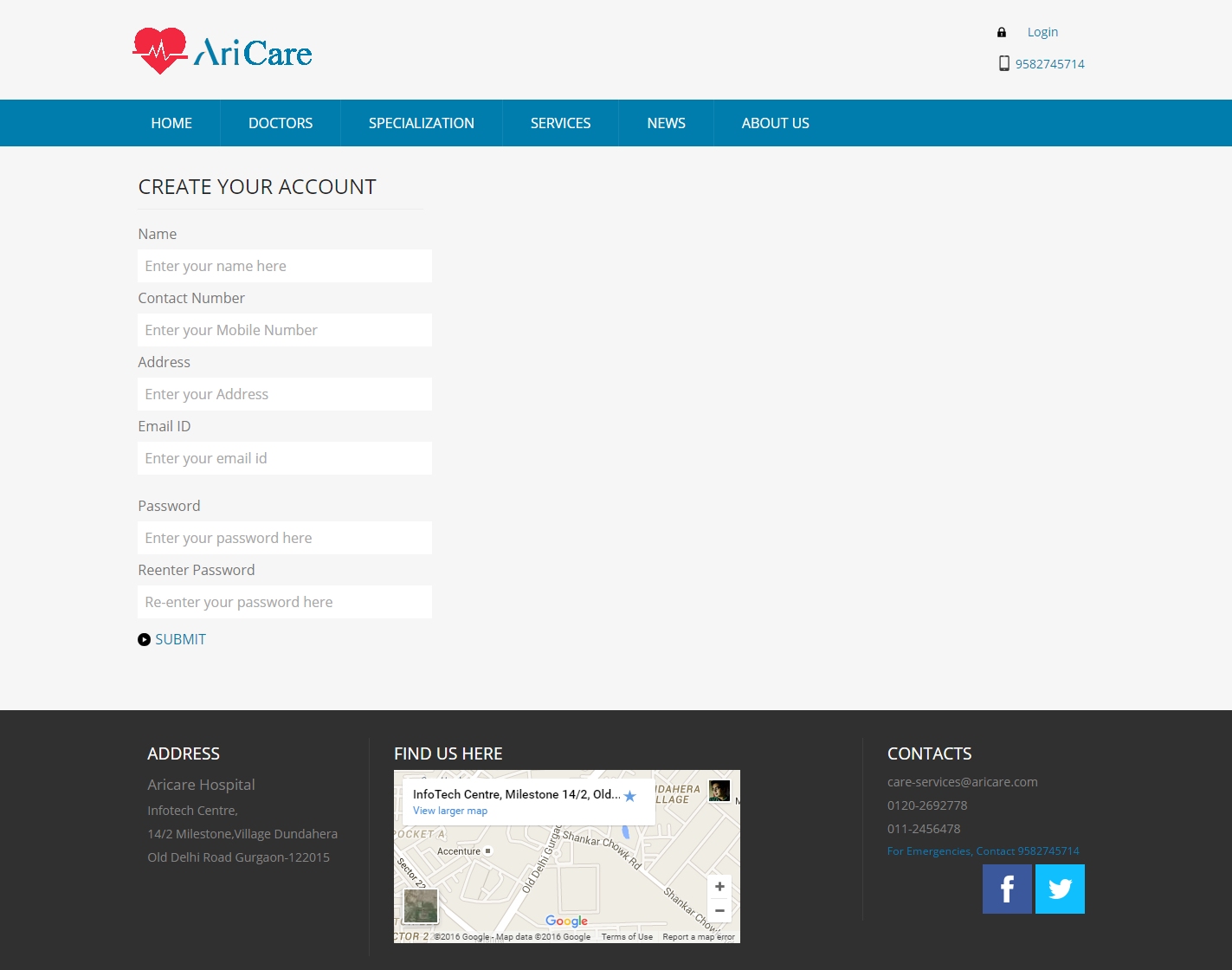


#### Login Page



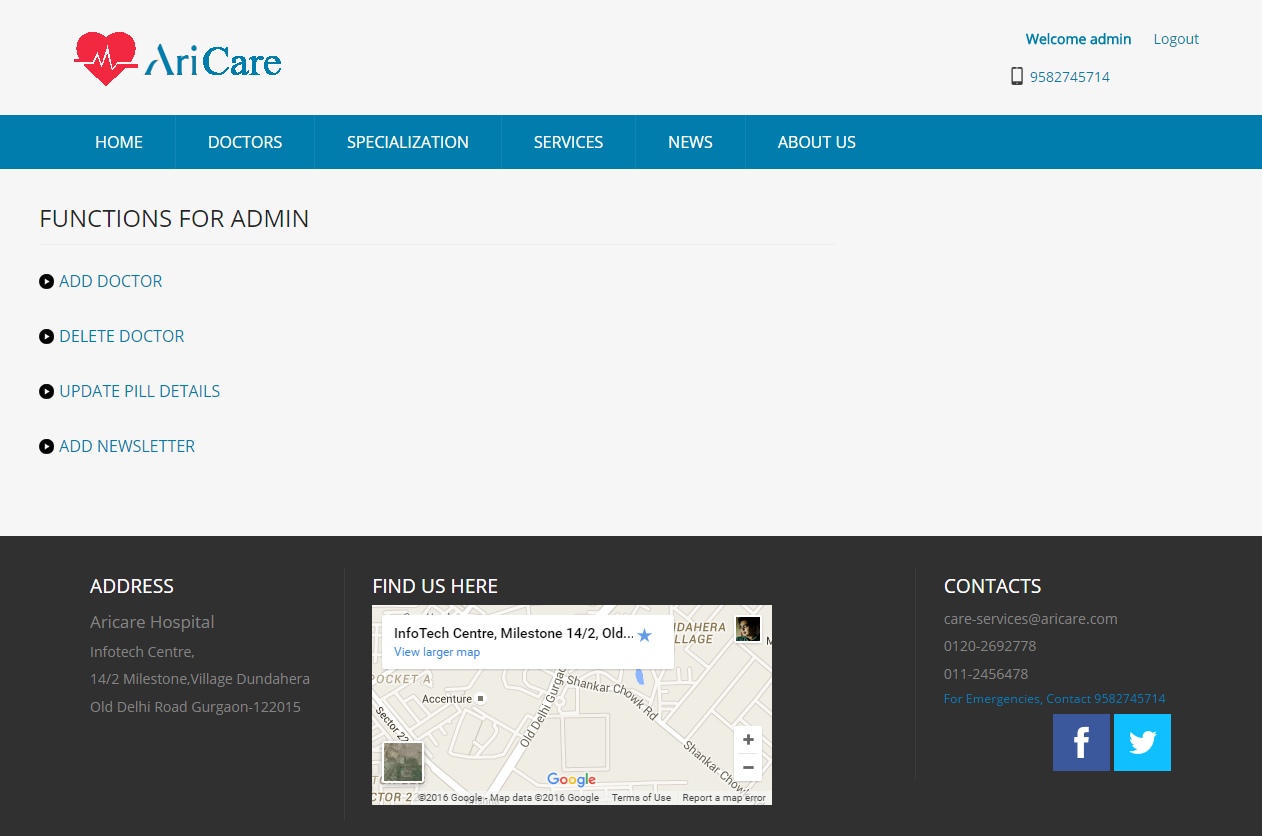
login module

#### Sign Up page



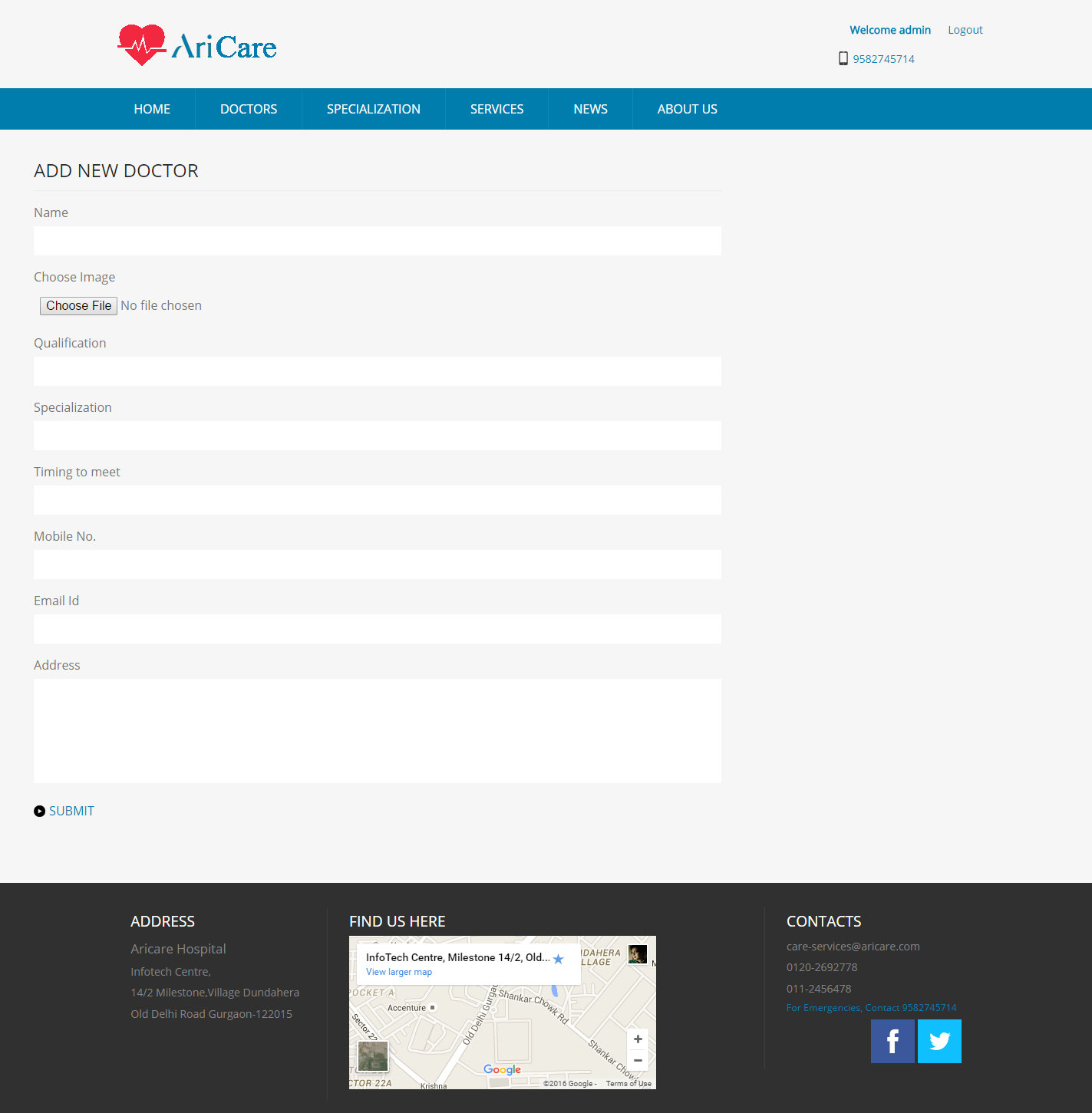
sign up module

#### Admin page



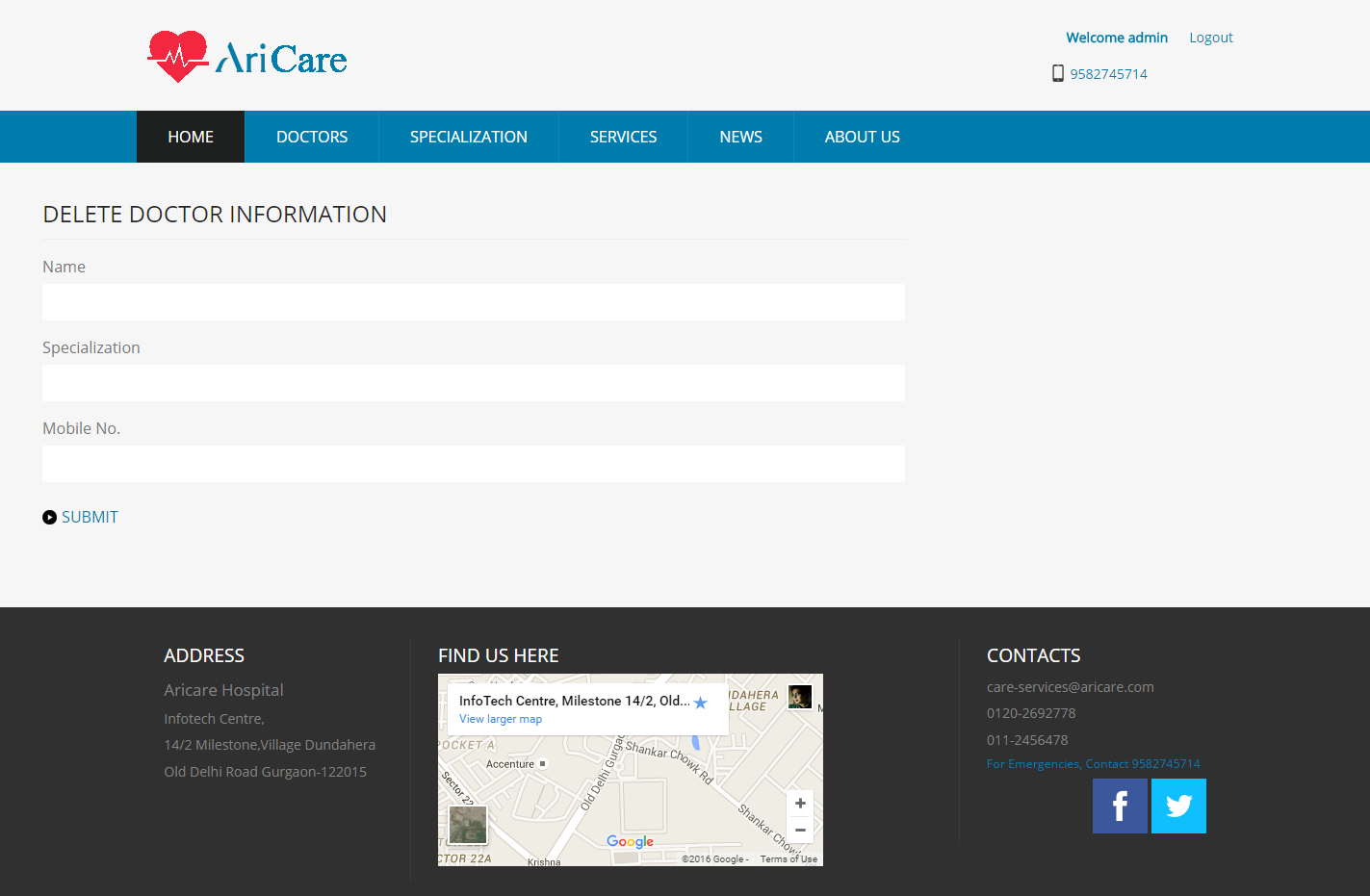
admin basic menu module

#### Add Doctor Page (admin)



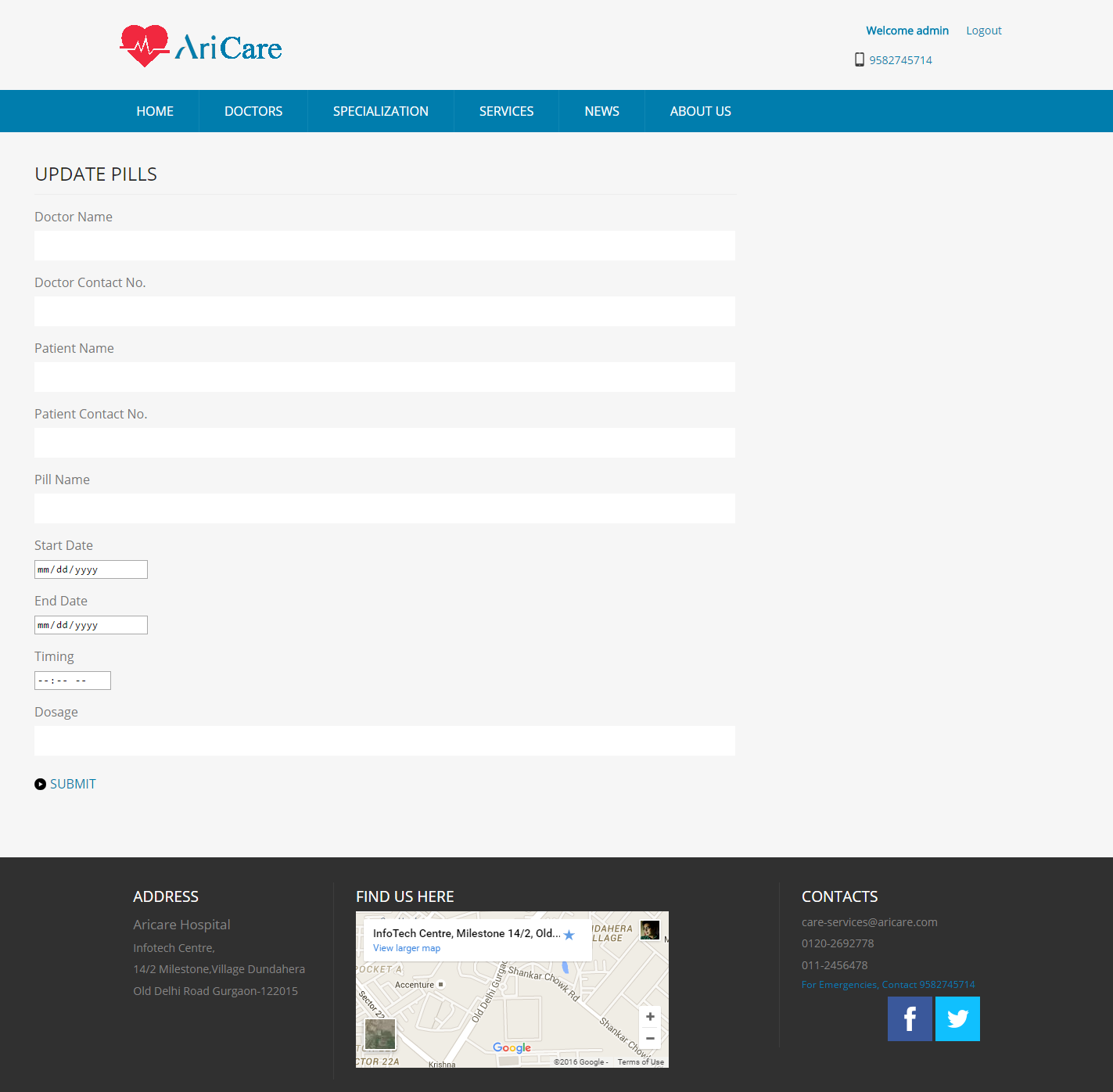
Adding a doctor(admin)

#### Delete Doctor page (admin)



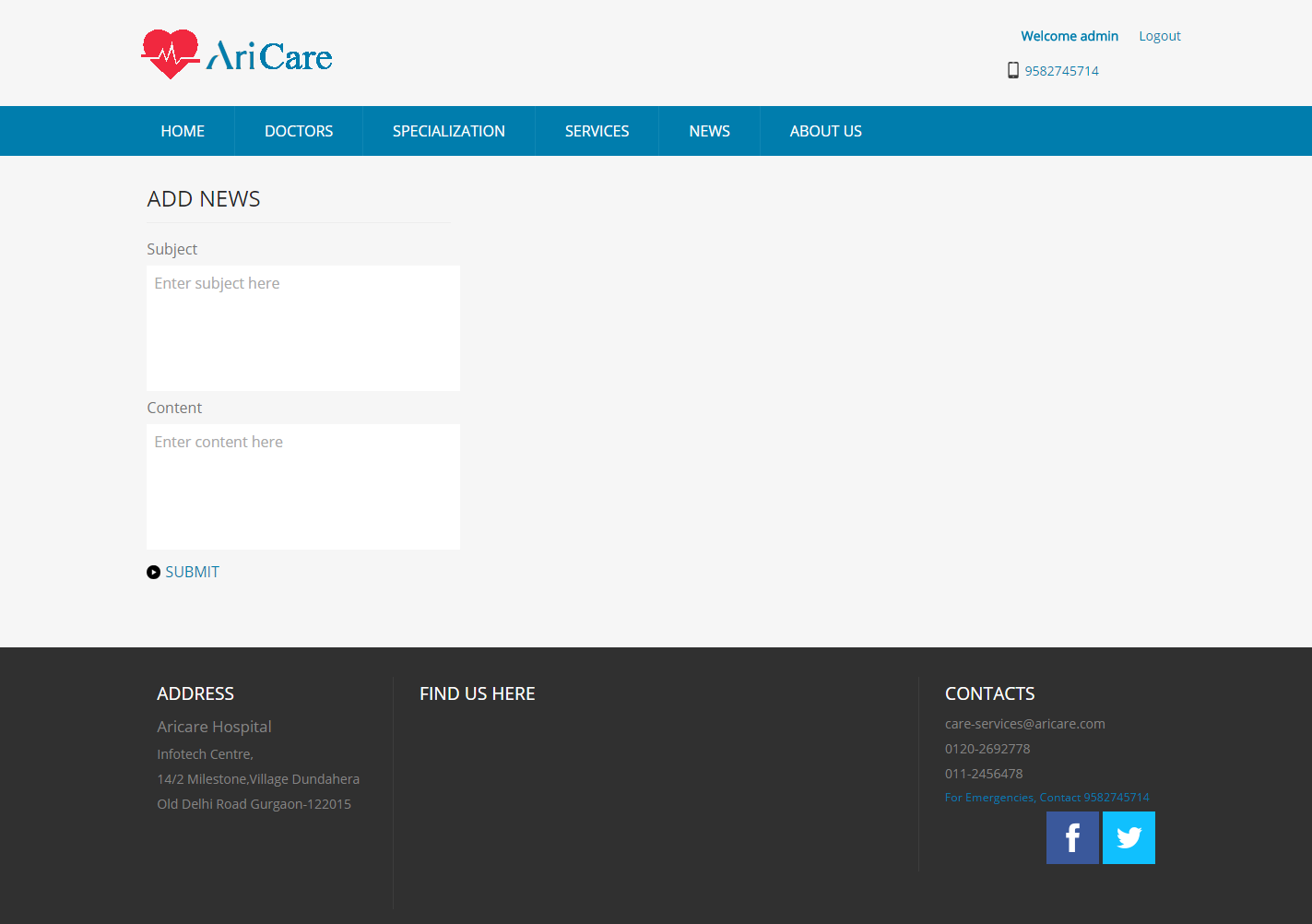
Deleting a doctor

#### Update Pill Details page (admin)

****

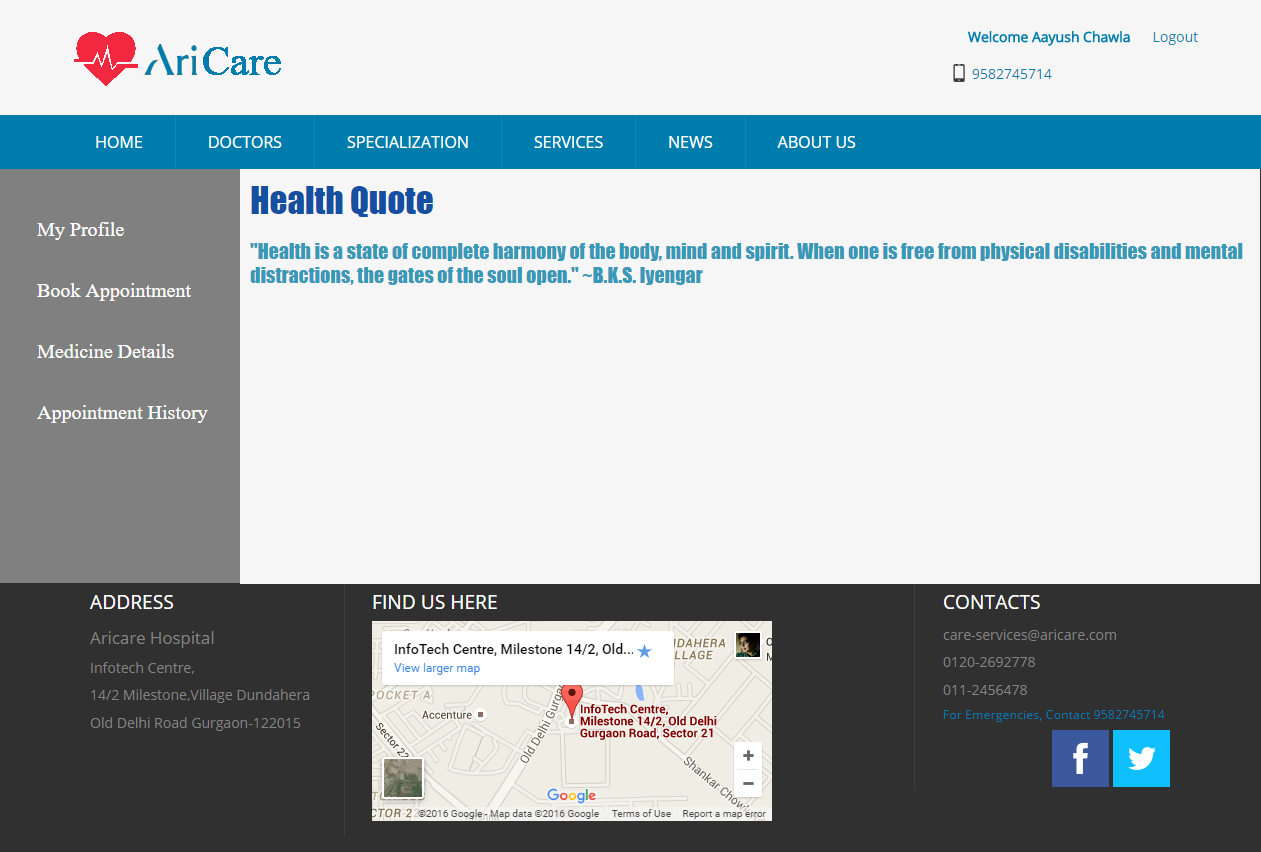
Updating pill details of patient(admin)

#### Add News page (admin)



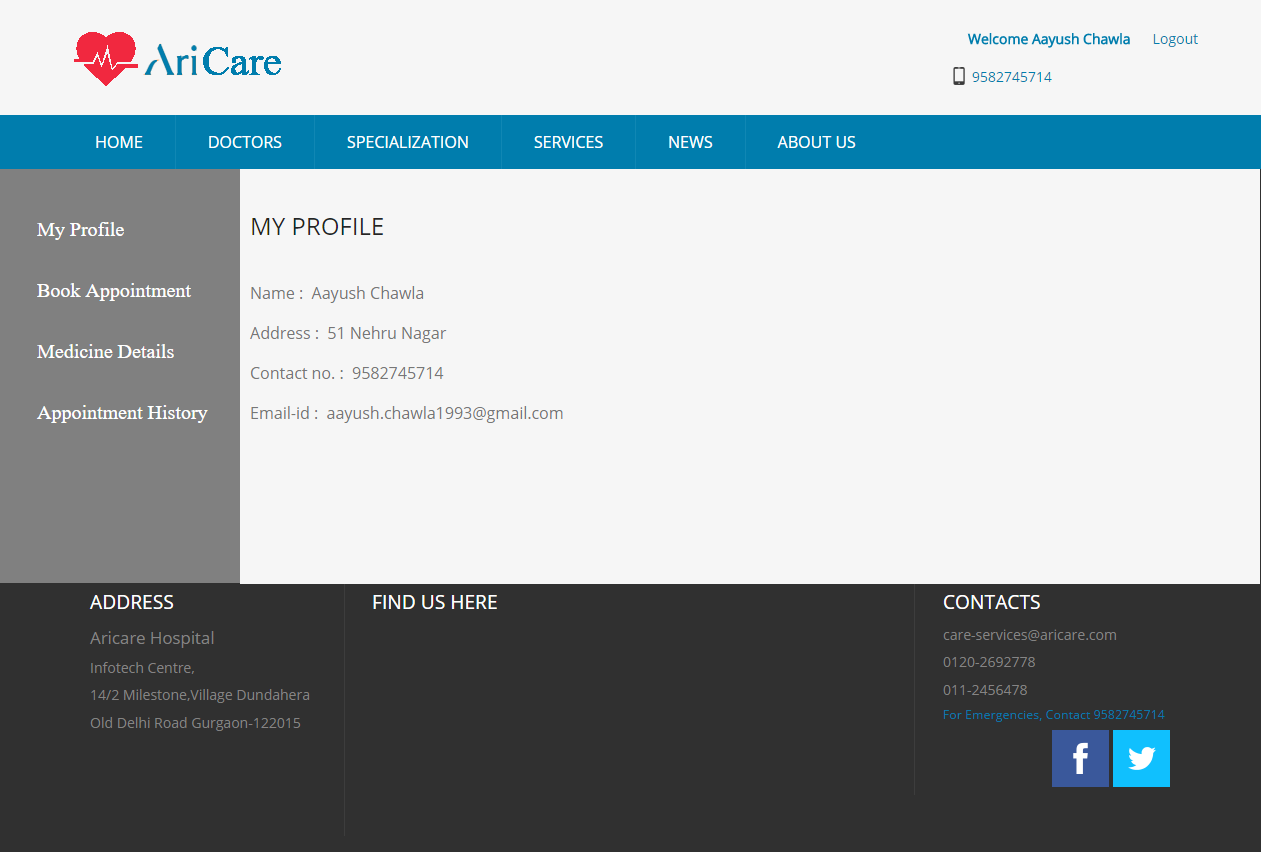
Adding news(admin)

#### Patient Logged in page



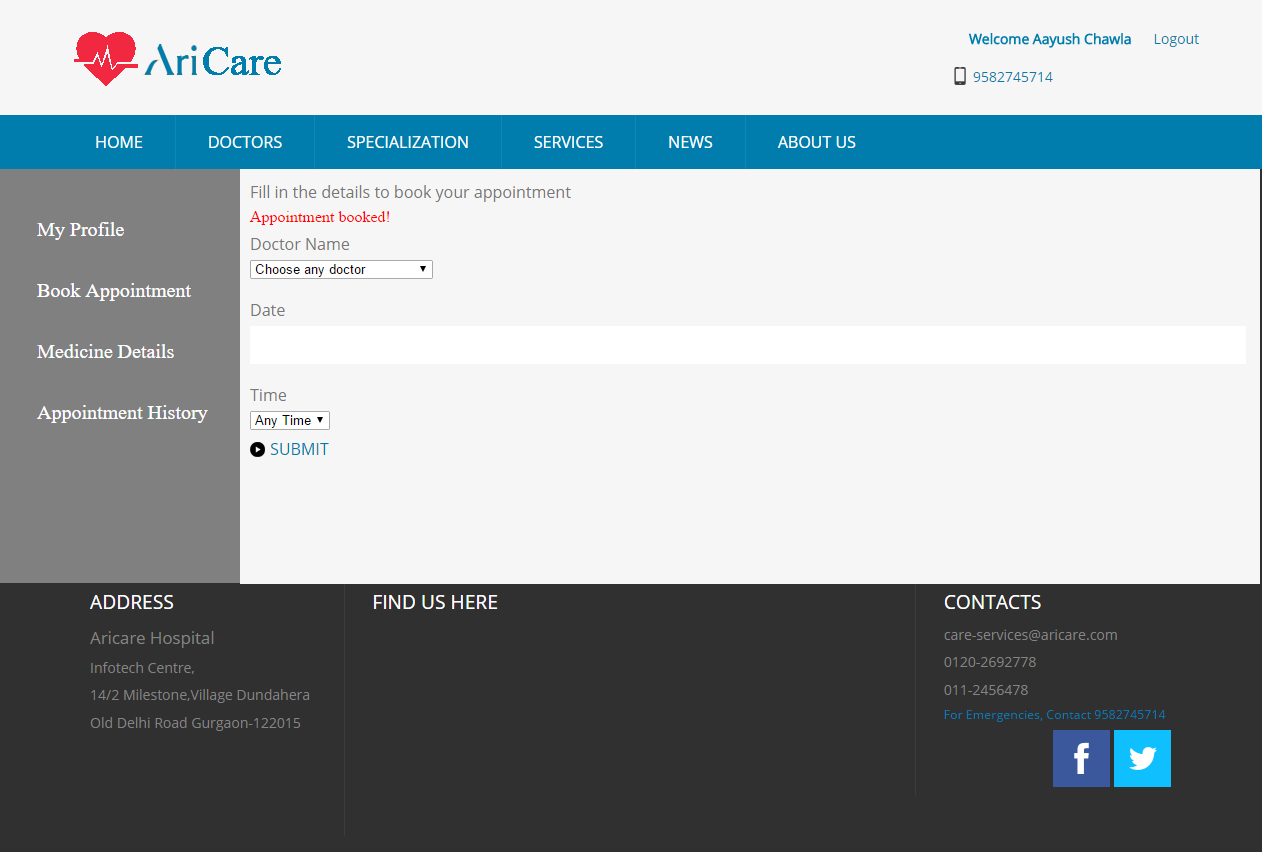
view patient’s page

#### Patient Profile Page



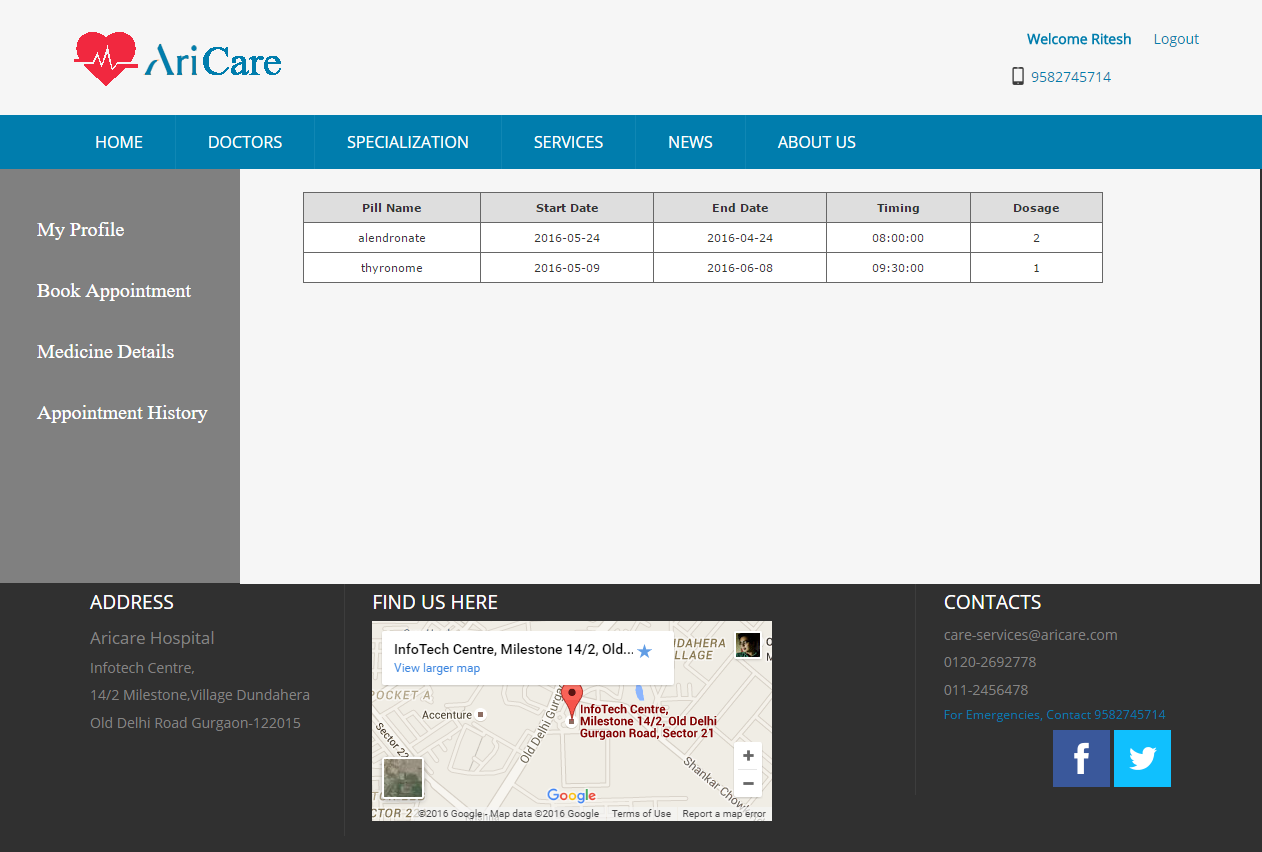
View Patient’s Profile

#### Book Appointment Page



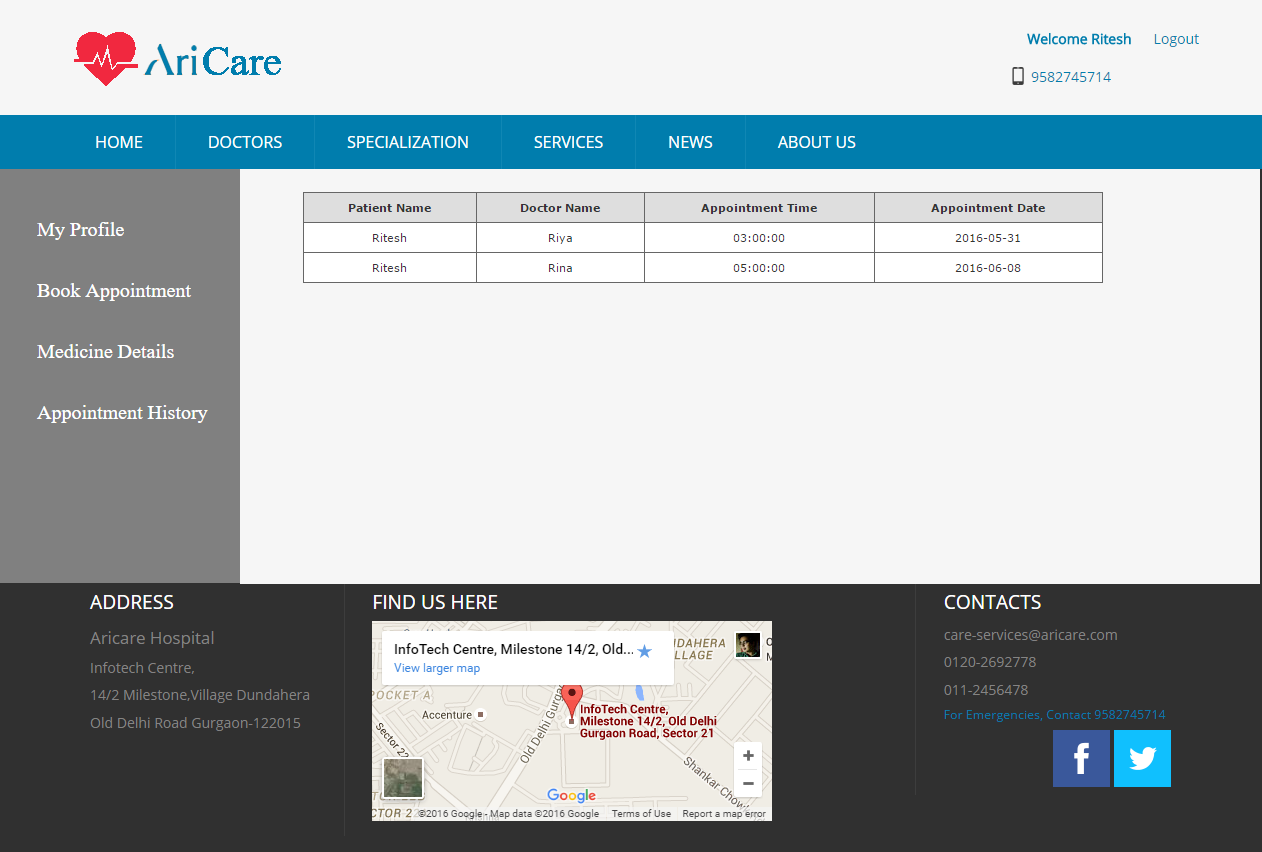
Book an appointment

#### Medicine Details Page



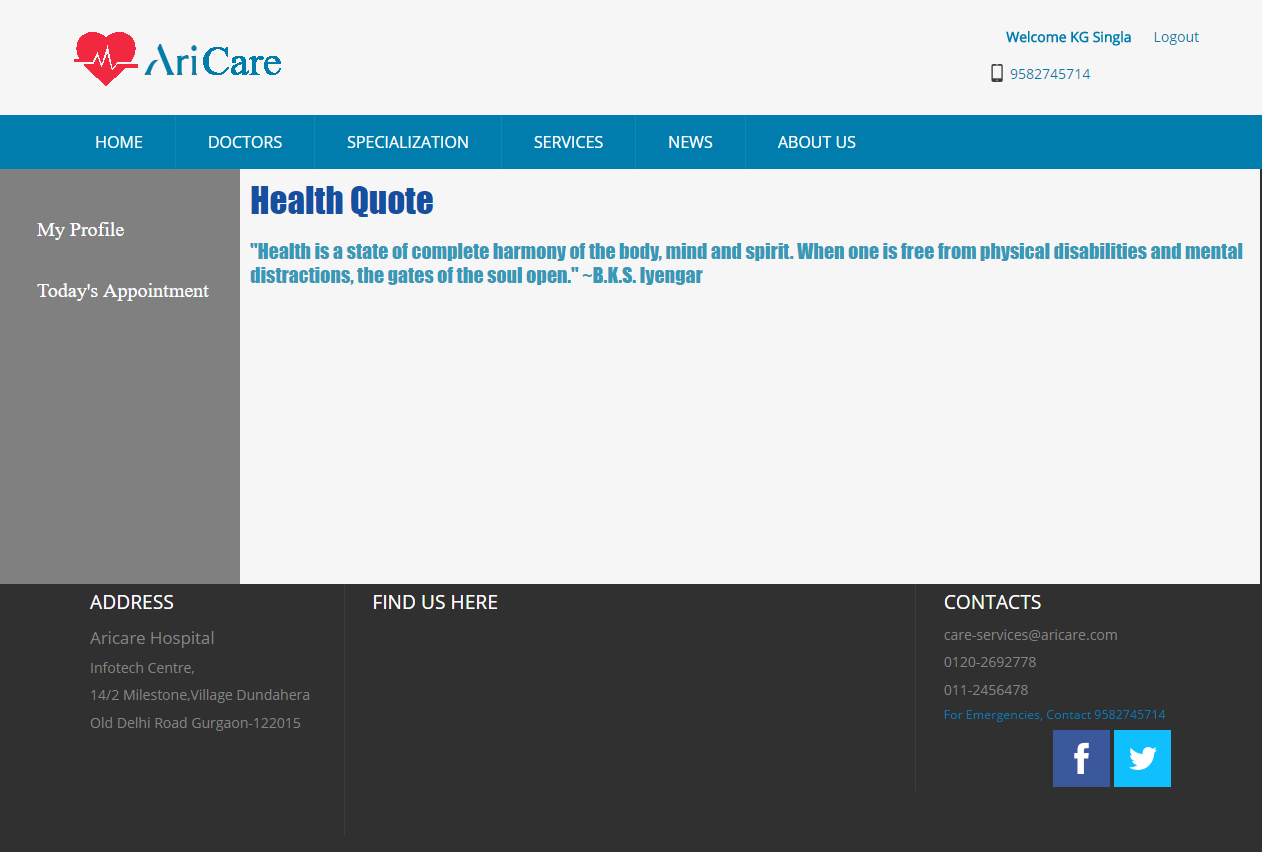
View Pill details

#### View Appointment History Page

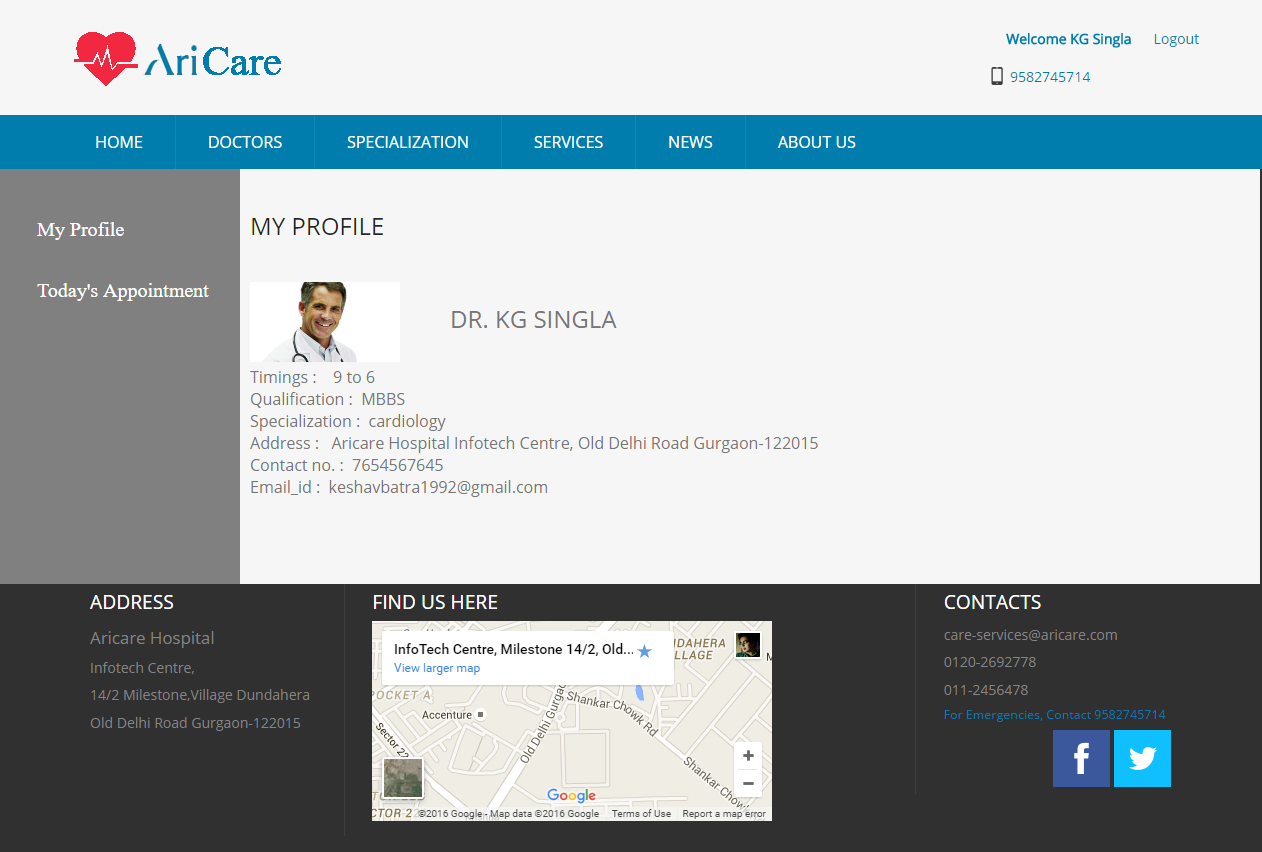


View patient’s appointment history

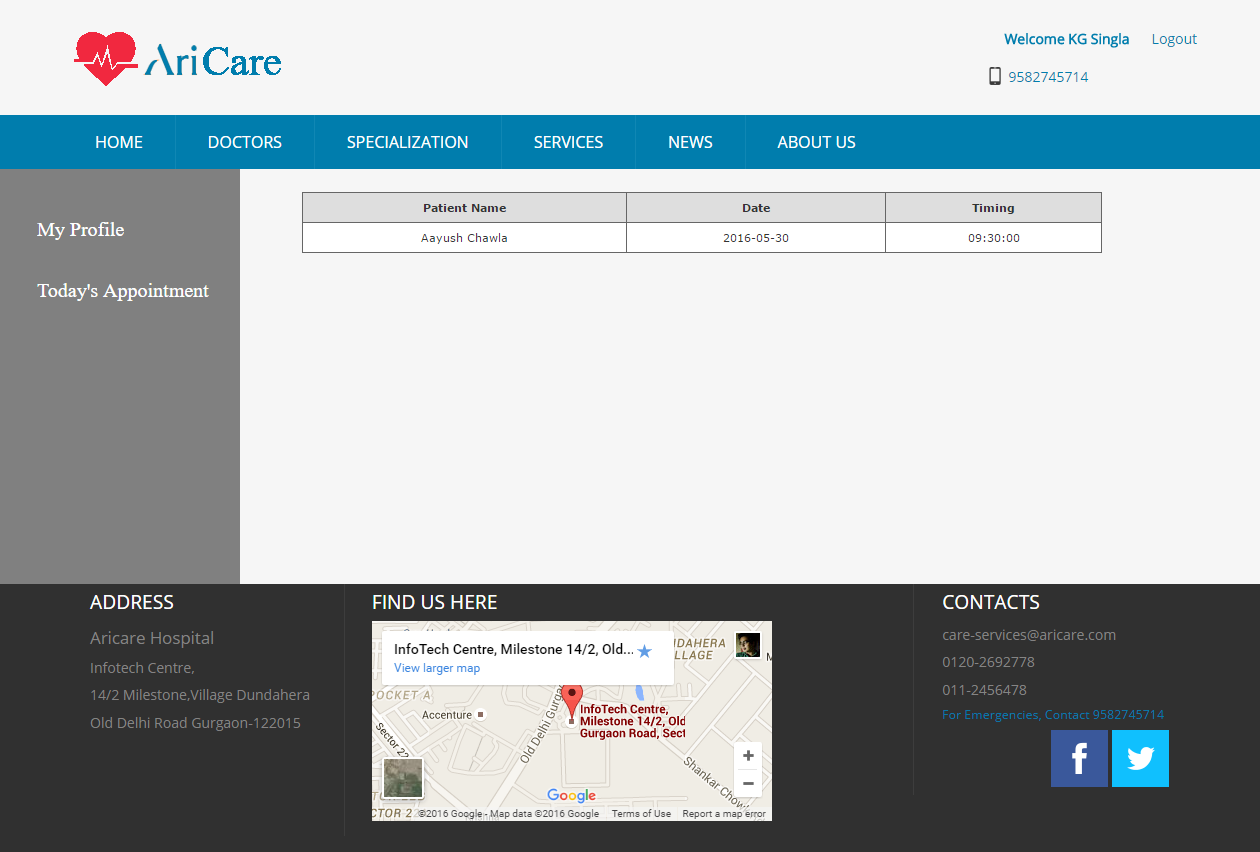
#### Doctor Page



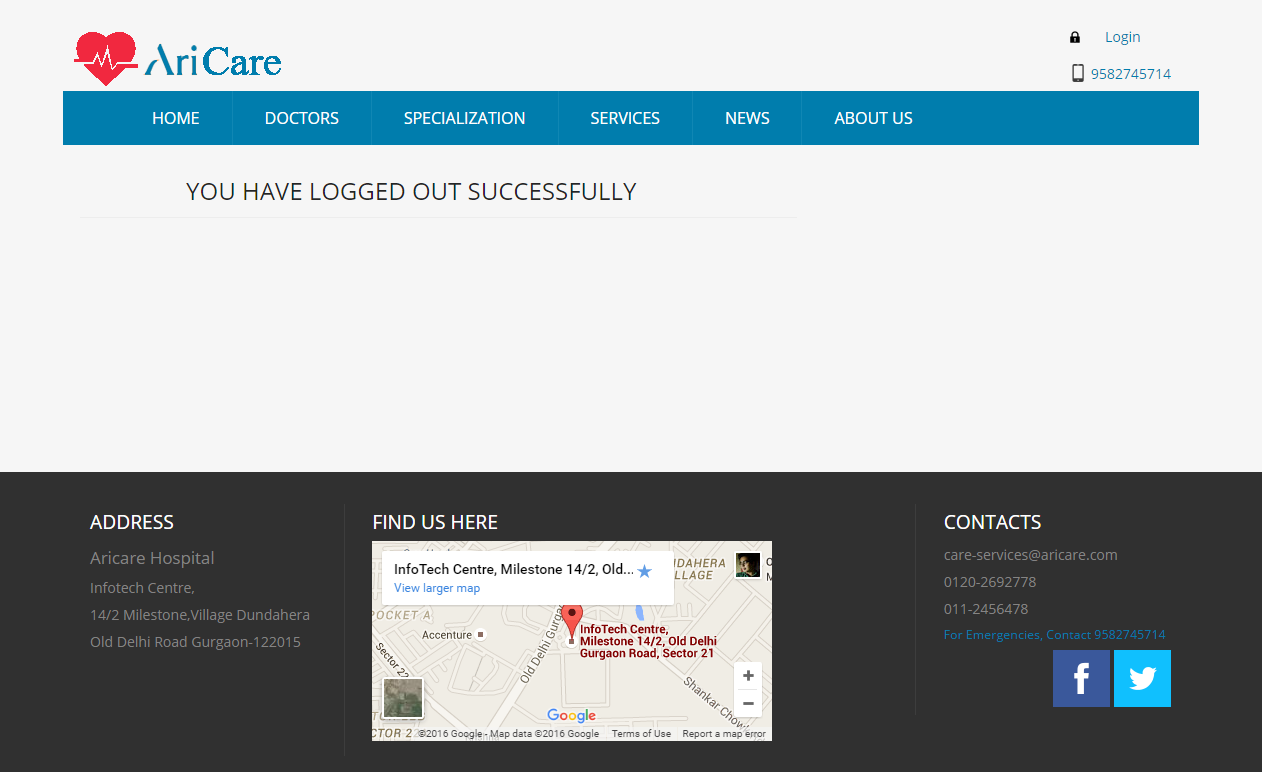
#### Doctor Profile Page



#### Doctor’s Today’s Appointment Page

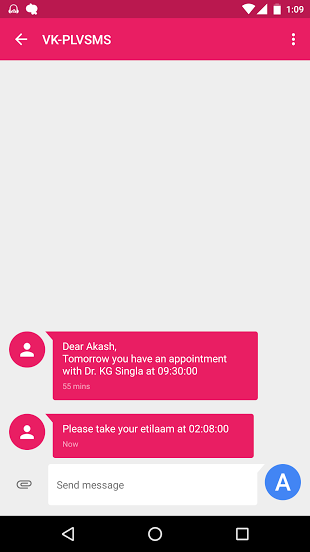


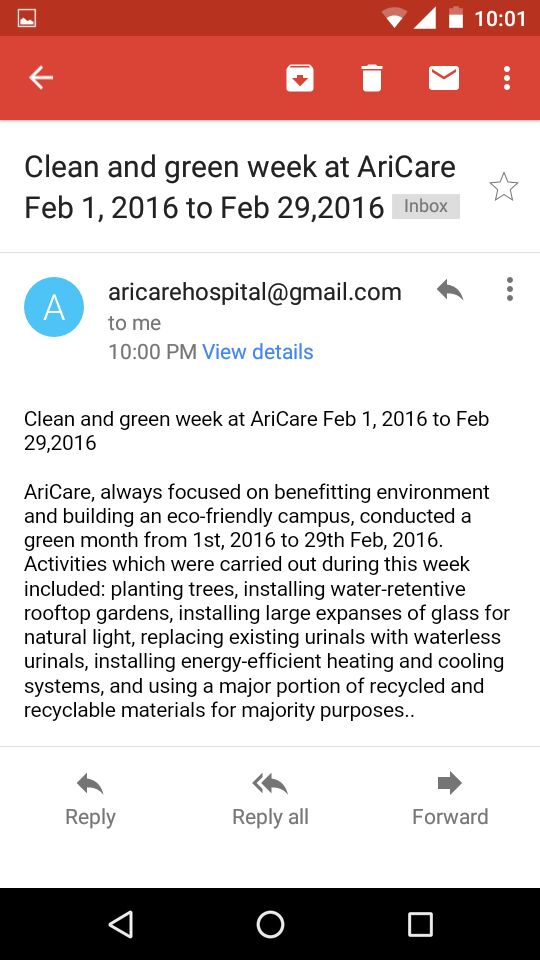
#### Logout Page



#### Mailing and SMS Notifications Page

#### Mac OS:Users:aayushchawla:Downloads:Screenshot_20160602-001438.png





## Configuration and Control

### Overview

Project ***AriCare*** runs on normal system configuration. The UI is compatible with Google Chrome browser. The system requires MySQL database and tomcat server to run the application. User will be provided with a login/registration page. After successful login, if user is a patient, he/she can schedule an appointment from the list of available slots, view his/her appointment history, and view details of pills prescribed to him/her. If the user is a doctor, then after successful login he/she can view his/her that day’s appointments. If the user is the admin, then he/she can delete a doctor from the database, add a new doctor to the database, and update pill details corresponding to a patient and a doctor. Also, a notification will be sent to each patient before the day of appointment and one hour before the scheduled pill time.

### Startup

* Import the database.
* Configure the tomcat server and start it to host the application.
* Open the login page.

### Shutdown

* Close the browser.
* Stop the server.

### Creation of patient New Entries

Insert into patient\_profile values (patient\_idpatient\_name,phone\_number,address,email)

values(patient\_idpatient\_name,phone\_number,address,email)

### Creation of doctor New Entries

insert into doctor\_details(doctor\_id,doctor\_name,phone\_number,address,qualification,timings,specialization,image,email\_id) values(doctor\_id,doctor\_name,phone\_number,address,qualification,timings,specialization,image,email\_id)

### Creation of appointment\_details New Entries

Insert into appointment\_details (appointment\_id,patient\_id,doctor\_id,patient\_name,doctor\_name,timing,

date)

values(appointment\_id,patient\_id,doctor\_id,patient\_name,doctor\_name,timing,date)

### Creation of pill\_details New Entries

Insert into pill\_details

(pill\_id,doctor\_id,patient\_id,pill\_name,start\_date,end\_date,dosage\_time,

dosage)

values(pill\_id,doctor\_id,patient\_id,pill\_name,start\_date,end\_date,

dosage\_time,dosage)

### Creation of News New Entries

Insert into newsletter(news\_id,subject,content)

values(news\_id,subject,content)

### Deletion of Doctor Entries

Delete from doctor\_profile where phone\_number=? And doctor\_name=?

And specialization=?

### Changes to Standard MIB

## Logging:

Introduce design information on logging object creation, destruction, and state of objects etc.

## Tracing

Introduce Tracing functionality may include tracing the functionality through which the protocol or application is transiting.

## Error Handling

### System Errors

N/A

### Interface Errors

N/A

### Protocol Errors

N/A

Appendix

###### Design Entity Attributes

#### Identification: DFD

Data Flow diagram

Describes the flow of data in the system.

#### Identification: ER Diagram

Entity Relationship diagram

An ER diagram is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them.

#### Identification: Pseudocode

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading. Pseudocode typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific code and some subroutines. The programming language is augmented with natural language description details, where convenient, or with compact mathematical notation. The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm.

#### Identification: PDL

Program Design Language

Program Design Language (or PDL, for short) is a method for designing and documenting methods and procedures in software. It is related to pseudocode, but unlike pseudocode, it is written in plain language without any terms that could suggest the use of any programming language or library.

Template Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rev | Date of Issue | Author | Approver | Scope |
| 1 | 28/04/2016 | TH3\_GR03 | Naveen Kumar |  |
|  |  |  |  |  |
|  |  |  |  |  |