Group 10

**<Master Clinic/10>**

**Project Master Clinic SDS 1.0CM_Id 04,01,2018**

**Master Clinic Statement of Work1.0_mc_04 01, 2018**

# Master Clinic

## Software Design Specification (SDS)
## Team 10 Master Clinic 1.1
## IdDateVersion: 1.0

### CM Identifier: Master Clinic1.0_mc_04 01, 2018

<The Software Design Specification (SDS) sections given in this outline are guidelines to the contents of your SDS. Include *at least* these sections. Your team may have good reasons for wanting to deviate from this proposed outline. In this case, you *must* motivate any deviations to me. If a section is not applicable in your case, do not delete it; instead, give the topic heading and write "Not applicable".

You will note that there is some overlap in the contents from document to document. This redundancy allows each document to stand on its own. >

# Revision History

| Sl. No. | Prepared/ Modified by | E-mail | Version | Date | Approved by | Descriptions/ Remarks |
|---|---|---|---|---|---|---|
| 1. | <Name(s) of the author(s)> | <e-mail> | <x.y> | <Mmm. dd, yyyy> | <Name of the approving authority> | <The action performed on the document. Include a link to the modified part> |

# Distribution list

| Name | E-mail | Notes |
|---|---|---|
| Dr: Ahmed Hamdy | | |
| TA: Ali El-Sedeek | alielseddeek@gmail.com | |
| TA: Dina El Reedy | dinaelreedy@gmail.com | |

# Table of Contents

# List of Tables

# List of Figures

# 1. Introduction

<This should introduce the document to the reader. It should define the purpose of writing this document and give a brief description for its contents>.

## 1.1 Purpose of this Document

<This should define the goals of writing this document. Full description of the main objectives of the SDS document. For example, *"The purpose of this document is to provide an overview of the proposed solution to the problems specified in the Software Requirements Specification Document"*, etc >

## 1.2 Scope

<This section defines the scope of the development framework. It describes which set of the projects modules lies within the development framework. Hence, the scope section may introduce agreed-upon limitations between the customer and the development organization. If the intention is to add a number of new features, or to enhance old features, it should be clearly stated. For example, *"The project is mainly concerned with building a centralized database system, and building modules that allows the users to reserve tickets through a web site or using their mobile phones*", "*The development of the reservation software is out of the scope of the development process. An old reservation program is already running on network-connected computers in the reservation office".* At this phase, you scope definition should be more detailed and more accurate than the one given in the Software Requirements Specification document >

## 1.3 Table of Acronyms and Definitions

<The purpose of this section is to define any technical/business acronyms/abbreviations in order to unify the vocabulary. If acronyms/abbreviations are stated below, it may be used later in the document, Be sure to alphabetize.>

| Term | 1.3.1 *Definition* |
|---|---|
| | |

## 1.4 References

<This section will include technical books and documents related to design issues. At a minimum, this section should reference the SRS. Give links to documents as appropriate. Be certain that the references you give are complete and in the appropriate format>

<This should state all materials (books/papers/presentations/emails/etc…) used as references to develop this document. If some references were placed earlier under the configuration management platform, the configuration management identifier (CMI), and the version number of the reference are required. All references MUST follow the following format:

[1] "Software Engineering", 7th Edition, Ian Sommerville, Addison-Wesley, 2004
[2] "The Unified Modeling Language User Guide", Grady Booch, James Rumbaugh, Ivar Jacobson, Addison-Wesley, 1998
[3] "Mastering UML with Rational Rose 2002", Wendy Boggs, Michael Boggs, SYBEX, 2002
[4] "Patterns of Enterprise Application Architecture", Martin Fowler, Addison Wesley, 2002
[5] Software Engineering Course Lecture Notes, Dr. El Sayed Hemayed.

The number associated with each reference [1] should appear in the parts of the document that refer to this reference>

### 1.5 Overview of Document

<A short description of how the rest of the SDS is organized and what can be found in the rest of the document. This is *not* simply a table of contents. Motivate and briefly describe the various parts!>

## 2. System Architecture

Now we combine all these diagrams together as subsystems to show the overall connection between them. Since we are using MVC design pattern views are connected to controllers and controllers are connected to models.We start with relation between controllers and models.

**Relation between controllers and models**

- Admin Controllers use all the models since admins are allowed to create, update or delete any other entity and themselves given the right permissions (only super admins can delete or update any other admins and they are the only ones who can create other super admins. other than that there is no other restriction for normal admins).
- Nurse Controllers use both Nurse model and Patient models since a nurse can update herself and create, update or delete a patient
- Patient Controllers use only Patient model as a patient can only update himself

### Relation between views and controllers

Since the navigation bar contains a link for logging out (If you are logged in if not it's replaced with a link to the login page) and a link for the home page this makes them every user (admin, nurse, patient or guest) views capable of accessing both the home controller and login controller. That's why no connections to these 2 controllers from their respective views would mentioned after in the next section.

**Admin**

- login page: can only access forgot password controller
- send reset email page: can only access forgot password controller
- reset password page: can only access reset password controller
- rest of the views: can access patient, nurse, admin and profile controllers due to the links that includes all these views in the side navigation bar. these links would available in every single view as soon admin is logged in to make it easier to perform different functionalities

**Nurse**

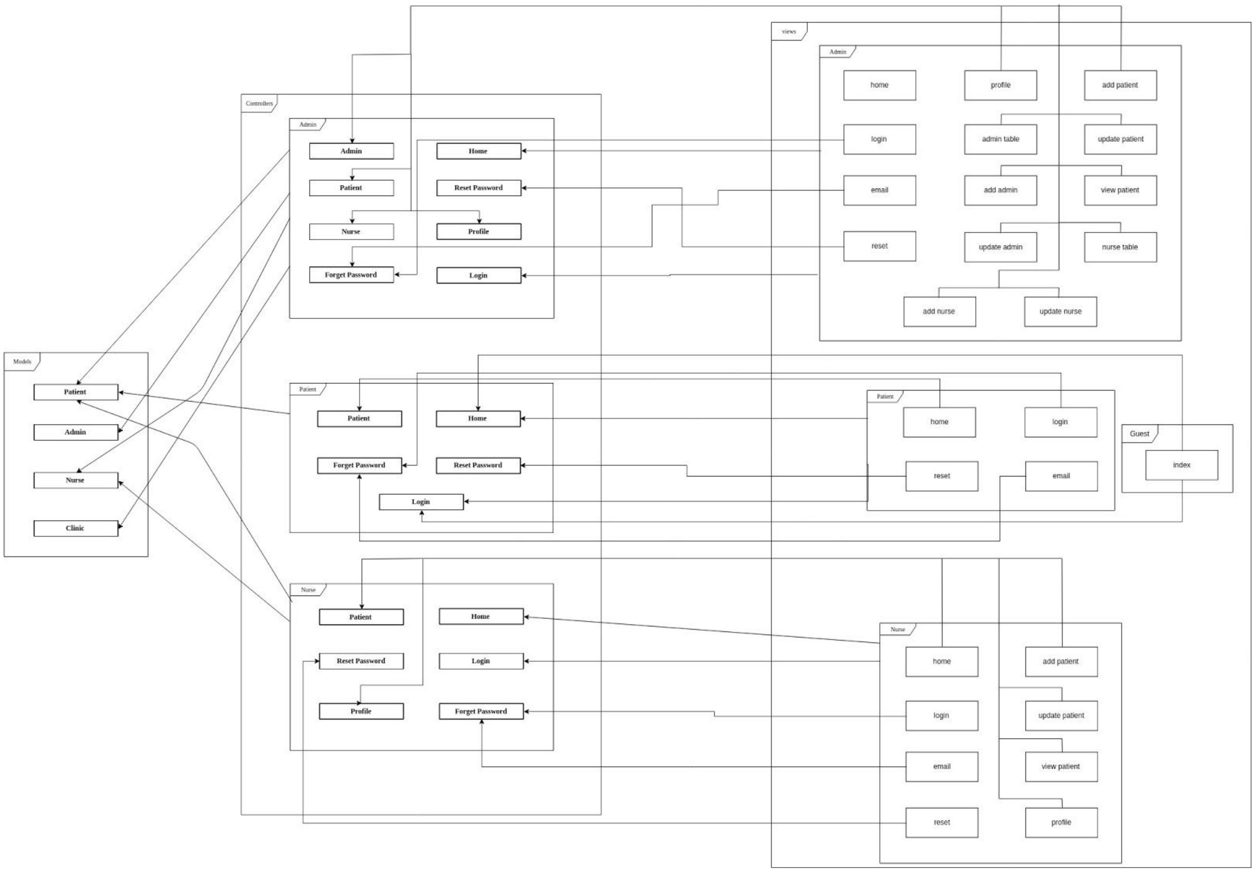
- login page: can only access forgot password controller
- send reset email page: can only access forgot password controller
- reset password page: can only access reset password controller
- rest of the views: can access patient and profile controllers due to the links that includes all these views in the side navigation bar. these links would available in every single view as soon nurse is logged in to make it easier to perform different functionalities

**Patient**

- login page: can only access forgot password controller
- send reset email page: can only access forgot password controller
- reset password page: can only access reset password controller
- home page: can access patient controller

**Guest**

- index page: accesses patient's home and login controllers as a normal guest is a logged out patient



# 3. Design Models

*<This should contains class, interaction, activity and state chart diagrams that models your system. Diagrams may contain classes from different subsystem/layers. Remember that your design should be complete and no ambiguities should appear. Developers will have to translate these diagrams into code. Assign each design element (class, sequence) an ID that can be used in the traceability matrix>*

## 3.1 Design Patterns Description

<You should illustrate which Design Patterns you've used, where and why? You've to use at least 2 Design Patterns>

### 3.1.1 *MVC Architectural design pattern*

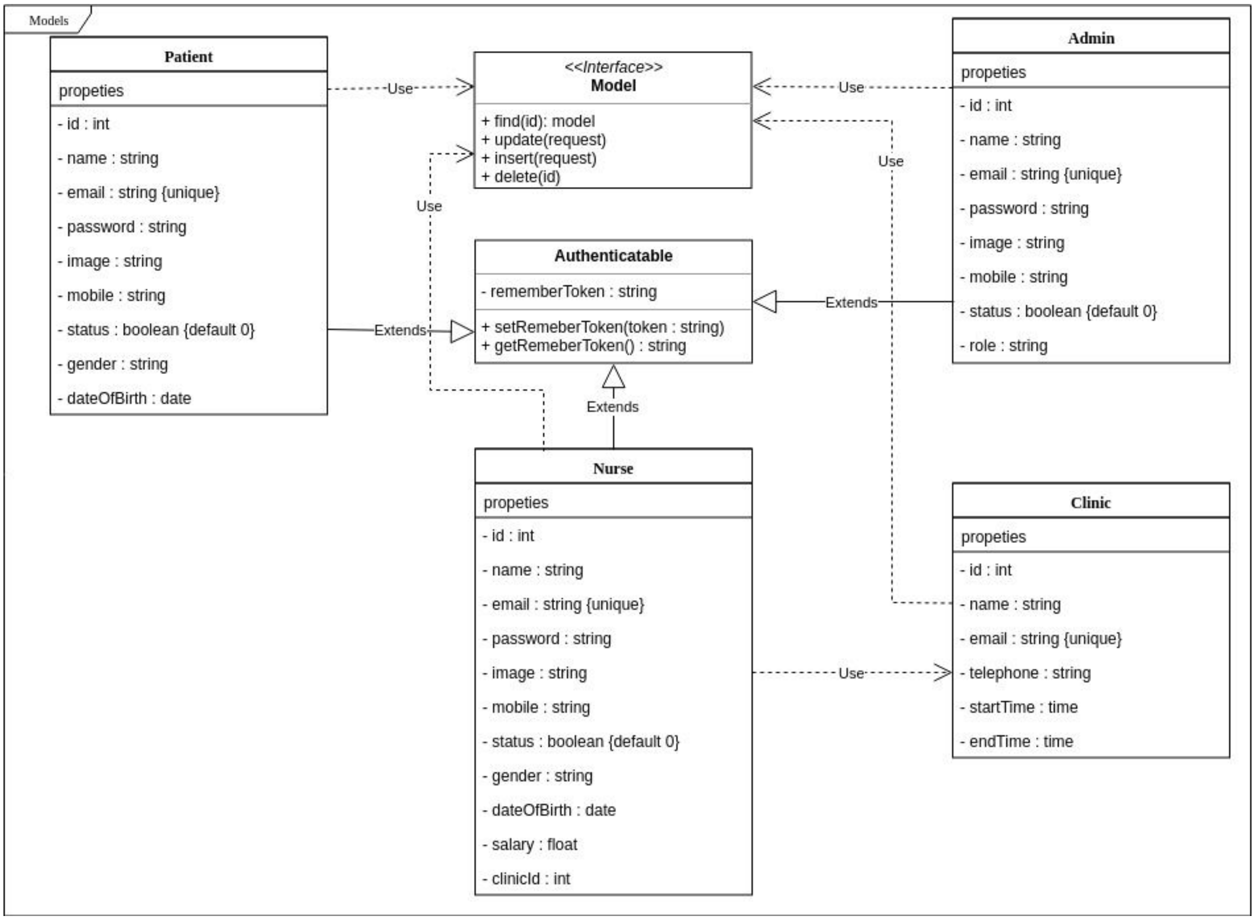
## 3.2 Class Diagrams

since MVC architecture is being used we present classes diagrams of three parts models, controllers and views then finally one diagram to show different interactions between them.

**Models**

We have four classes in the current phase Patient, Admin, Nurse and Clinic which are the basic classes required to initiate the project.

1. Patient Class: this class is used to handle patients and their data
2. Nurse Class: this class is used to handle nurses and their data
3. Admin Class: this class is used to handle admins (doctors) and their data
4. Clinic Class: this class is used to handle clinics and their data
5. Model (interface): interface for interacting with database that provides function to insert, update, delete, retrieve, etc any data from a certain table or through different relationships between tables
6. Authenticable: Base Class that provide authentication (login/out, session, guards, etc).



this diagram shows all six classes mentioned above. The actor classes (Patient, Nurse, Admin) inherit the basic class Authenticable to provide the authentication functionality. Also all classes use Model interface to interact with database tables in more secure and easier way other than direct queries.

## Controllers

Controllers are divided into three main blocks every block represents a user and his associated controllers.

**Admin**

1. admin controller: controller to handle functions on admins (create, update, delete, list, etc)
2. nurse controller: controller to handle functions on nurses (create, update, delete, list, setStatus, setSalary, etc)
3. patient controller: controller to handle functions on patients (create, update, delete, list, setStatus,

etc)

4. home controller: controller to handle functions of the home page
5. profile controller: controller to handle functions of the profile page (update profile photo, update profile data, change password)
6. login controller: controller to handle logging functions (login, logout, check guards, check credentials)
7. forgot password controller: controller to handle sending reset password emails
8. reset password controller: controller to handle resetting password from reset email

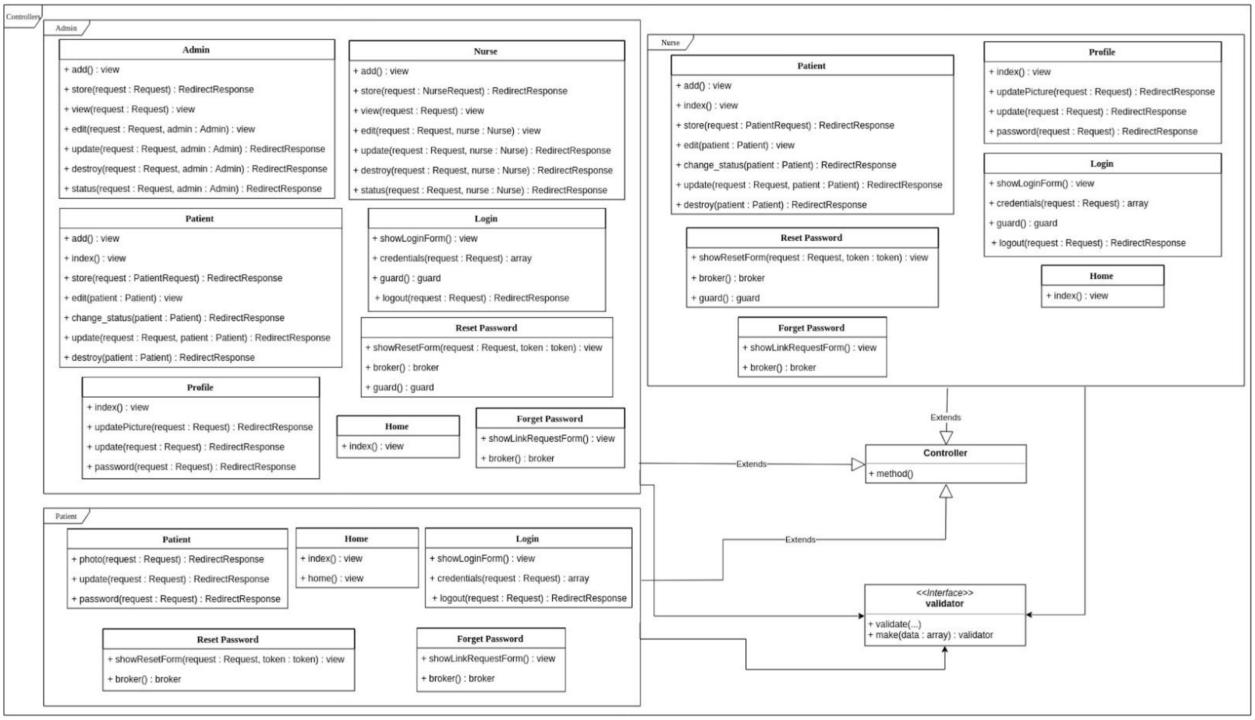
**Nurse**

1. patient controller: controller to handle functions on patients (create, update, delete, list, setStatus, etc)
2. home controller: controller to handle functions of the home page
3. profile controller: controller to handle functions of the profile page (update profile photo, update profile data, change password)
4. login controller: controller to handle logging functions (login, logout, check guards, check credentials)
5. forgot password controller: controller to handle sending reset password emails
6. reset password controller: controller to handle resetting password from reset email

**Patient**

1. patient controller: controller to handle functions on patients ( update photo, update data, change password)
2. home controller: controller to handle functions of the profile page
3. login controller: controller to handle logging functions (login, logout, check guards, check credentials)
4. forgot password controller: controller to handle sending reset password emails
5. reset password controller: controller to handle resetting password from reset email

All controllers inherit from base class controller that provide basic controller functions. They also use interface Validator that performs extra validation on the data to inform that it’s secure and makes sense.



Controllers
Admin
Admin
+ add() : view
+ store(request : Request) : RedirectResponse
+ view(request : Request) : view
+ edit(request : Request, admin : Admin) : view
+ update(request : Request, admin : Admin) : RedirectResponse
+ destroy(request : Request, admin : Admin) : RedirectResponse
+ status(request : Request, admin : Admin) : RedirectResponse
Nurse
+ add() : view
+ store(request : NurseRequest) : RedirectResponse
+ view(request : Request) : view
+ edit(request : Request, nurse : Nurse) : view
+ update(request : Request, nurse : Nurse) : RedirectResponse
+ destroy(request : Request, nurse : Nurse) : RedirectResponse
+ status(request : Request, nurse : Nurse) : RedirectResponse
Patient
+ add() : view
+ index() : view
+ store(request : PatientRequest) : RedirectResponse
+ edit(patient : Patient) : view
+ change_status(patient : Patient) : RedirectResponse
+ update(request : Request, patient : Patient) : RedirectResponse
+ destroy(patient : Patient) : RedirectResponse
Login
+ showLoginForm() : view
+ credentials(request : Request) : array
+ guard() : guard
+ logout(request : Request) : RedirectResponse
Reset Password
+ showResetForm(request : Request, token : token) : view
+ broker() : broker
+ guard() : guard
Profile
+ index() : view
+ updatePicture(request : Request) : RedirectResponse
+ update(request : Request) : RedirectResponse
+ password(request : Request) : RedirectResponse
Home
+ index() : view
Forget Password
+ showLinkRequestForm() : view
+ broker() : broker
Nurse
Patient
+ add() : view
+ index() : view
+ store(request : PatientRequest) : RedirectResponse
+ edit(patient : Patient) : view
+ change_status(patient : Patient) : RedirectResponse
+ update(request : Request, patient : Patient) : RedirectResponse
+ destroy(patient : Patient) : RedirectResponse
Profile
+ index() : view
+ updatePicture(request : Request) : RedirectResponse
+ update(request : Request) : RedirectResponse
+ password(request : Request) : RedirectResponse
Login
+ showLoginForm() : view
+ credentials(request : Request) : array
+ guard() : guard
+ logout(request : Request) : RedirectResponse
Reset Password
+ showResetForm(request : Request, token : token) : view
+ broker() : broker
+ guard() : guard
Home
+ index() : view
Forget Password
+ showLinkRequestForm() : view
+ broker() : broker
Extends
Extends
Controller
+ method()
Extends
Patient
Patient
+ photo(request : Request) : RedirectResponse
+ update(request : Request) : RedirectResponse
+ password(request : Request) : RedirectResponse
Home
+ index() : view
+ home() : view
Login
+ showLoginForm() : view
+ credentials(request : Request) : array
+ logout(request : Request) : RedirectResponse
<<Interface>>
validator
+ validate(...)
+ make(data : array) : validator
Reset Password
+ showResetForm(request : Request, token : token) : view
+ broker() : broker
Forget Password
+ showLinkRequestForm() : view
+ broker() : broker

## Views

Same as controllers views are also divided into three blocks that represents a user and his associated views and one extra block for Guest.

**Admin**

1. home page: one of the basic pages that is considered to as dashboard where you can access any other page and view any recent changes or updates
2. profile page: the second basic page that contains all admin's data and profile picture also contains a section or updating data, profile picture and changing password
3. add admin page: one of the admin pages that is used to add new admins
4. admin table: second admin page that displays a table of all existing admins and enable updating or deleting any other admin
5. admin update: the last admin page that contains a form for updating a specific admin chosen from admins table
6. add nurse: the first nurse page that is used to add new nurses
7. nurse table: second nurse page that same as admins table displays a table of all existing nurses with functionalities update and delete
8. nurse update: the last nurse page that is used to update a specific nurse chosen from nurse's table
9. add patient: the first patient page that is used to add new patients
10. patient table: second patient page that same as admin's and nurse's table displays a table of all existing patients with functionalities update and delete
11. patient update: the last patient page that is used to update a specific patient chosen from nurse's table
12. login page: a generic page used in all three blocks for logging in however in admin section it could only be access with the link for security issues
13. email page: also generic page for sending forgot password emails however it could be only

accessed from admin login page

14. reset page: the last generic page that is used to reset password of a specific admin and could only be accessed from the link emailed to the admin on his provided email account

**Nurse**

1. home page: one of the basic pages that is considered to as dashboard where you can access any other page and view any recent changes or updates
2. home controller: controller to handle functions of the home page
3. add patient: the first patient page that is used to add new patients
4. patient table: second patient page that displays a table of all existing patients with functionalities update and delete
5. patient update: the last patient page that is used to update a specific patient chosen from nurse's table
6. login page: a generic page used in all three blocks for logging in however in nurse section it could only be access with the link for security issues
7. email page: also generic page for sending forgot password emails however it could be only accessed from nurse login page
8. reset page: the last generic page that is used to reset password of a specific nurse and could only be accessed from the link emailed to the nurse on her provided email account

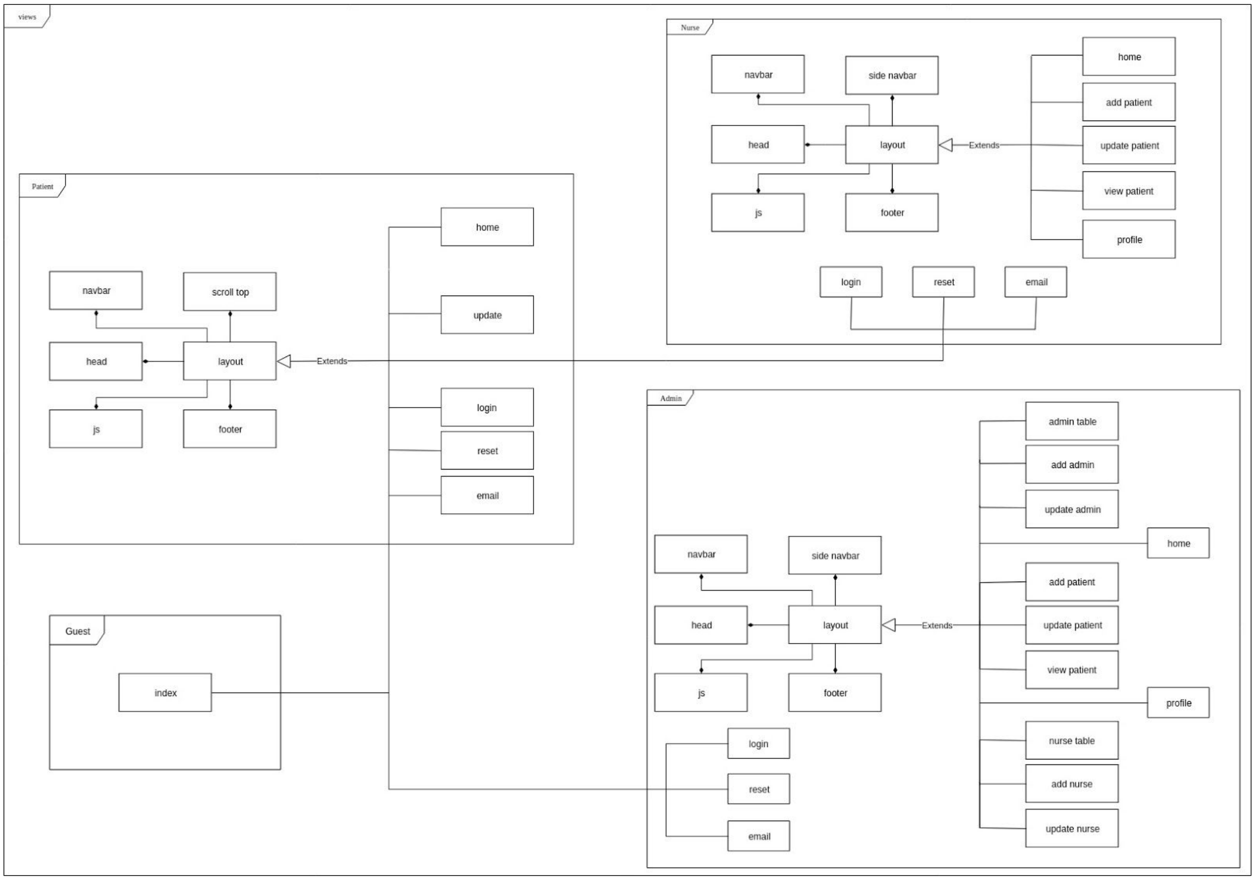
**Patient**

1. home page: patient profile page that contains his/er data and profile picture with two section one for updating data and profile picture, the other section is for changing password
2. login page: a generic page used in all three blocks for logging in however in nurse section it could only be access with the link for security issues
3. email page: also generic page for sending forgot password emails however it could be only accessed from nurse login page
4. reset page: the last generic page that is used to reset password of a specific nurse and could only be accessed from the link emailed to the nurse on her provided email account

**Guest**

1. index page: is the very first page of the website that could be accessed by anyone having the url of the website and contains three sections statistics sections then section that shows provided sections and finally section doctors sections

Every user block have its own layout that is used as a container for the entire page content however, all login, forgot password and reset password pages uses patient layout. such pages are accessed before signing in so they are all the same the only difference is nurse and admin pages are only accessible with their specific link.



Now we combine all these diagrams together as subsystems to show the overall connection between them. Since we are using MVC design pattern views are connected to controllers and controllers are connected to models.We start with relation between controllers and models.

### Relation between controllers and models

- Admin Controllers use all the models since admins are allowed to create, update or delete any other entity and themselves given the right permissions (only super admins can delete or update any other admins and they are the only ones who can create other super admins. other than that there is no other restriction for normal admins).
- Nurse Controllers use both Nurse model and Patient models since a nurse can update herself and create, update or delete a patient
- Patient Controllers use only Patient model as a patient can only update himself

### Relation between views and controllers

Since the navigation bar contains a link for logging out (If you are logged in if not it's replaced with a link to the login page) and a link for the home page this makes them every user (admin, nurse, patient or guest) views capable of accessing both the home controller and login controller. That's why no connections to these 2 controllers from their respective views would mentioned after in the next section.

**Admin**

- login page: can only access forgot password controller
- send reset email page: can only access forgot password controller
- reset password page: can only access reset password controller
- rest of the views: can access patient, nurse, admin and profile controllers due to the links that includes all these views in the side navigation bar. these links would available in every single view as soon admin is logged in to make it easier to perform different functionalities

**Nurse**

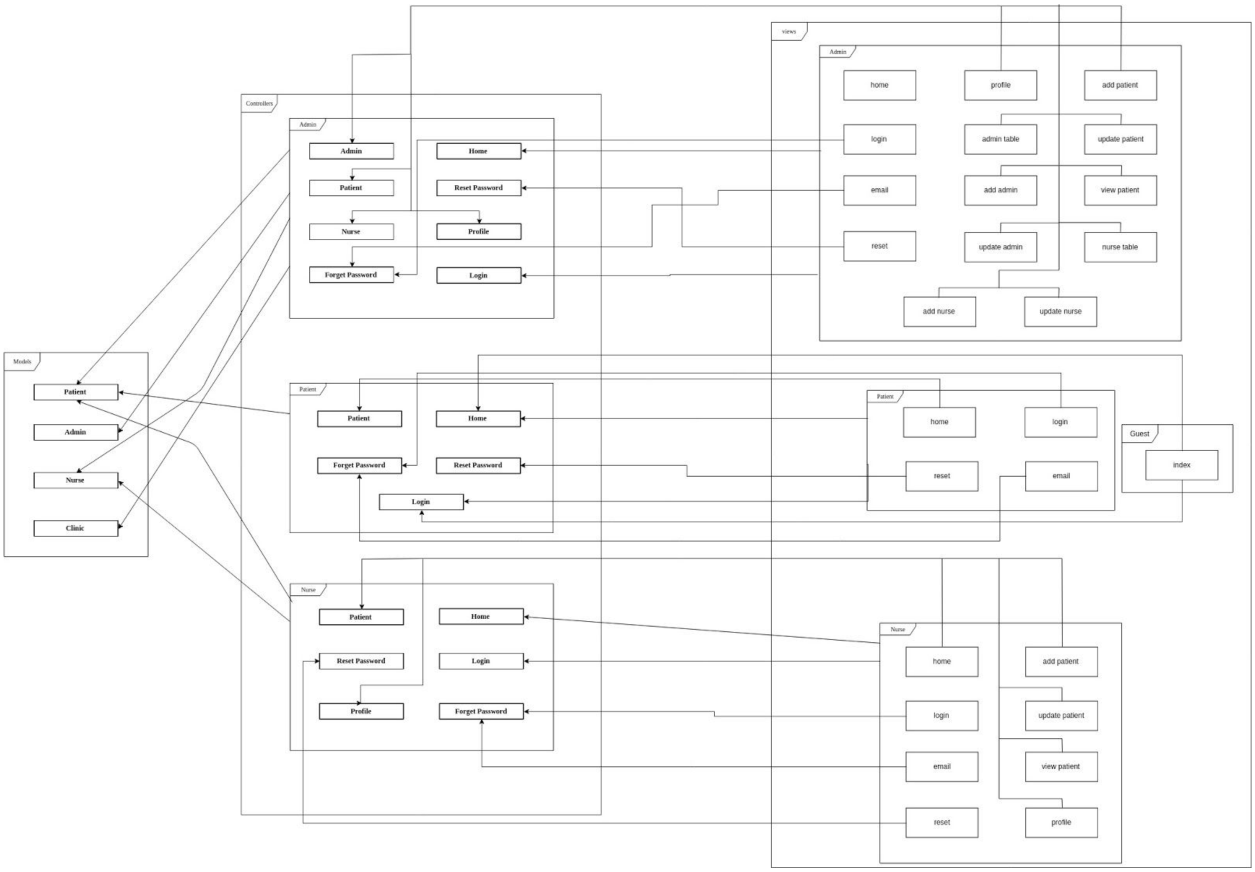
- login page: can only access forgot password controller
- send reset email page: can only access forgot password controller
- reset password page: can only access reset password controller
- rest of the views: can access patient and profile controllers due to the links that includes all these views in the side navigation bar. these links would available in every single view as soon nurse is logged in to make it easier to perform different functionalities

**Patient**

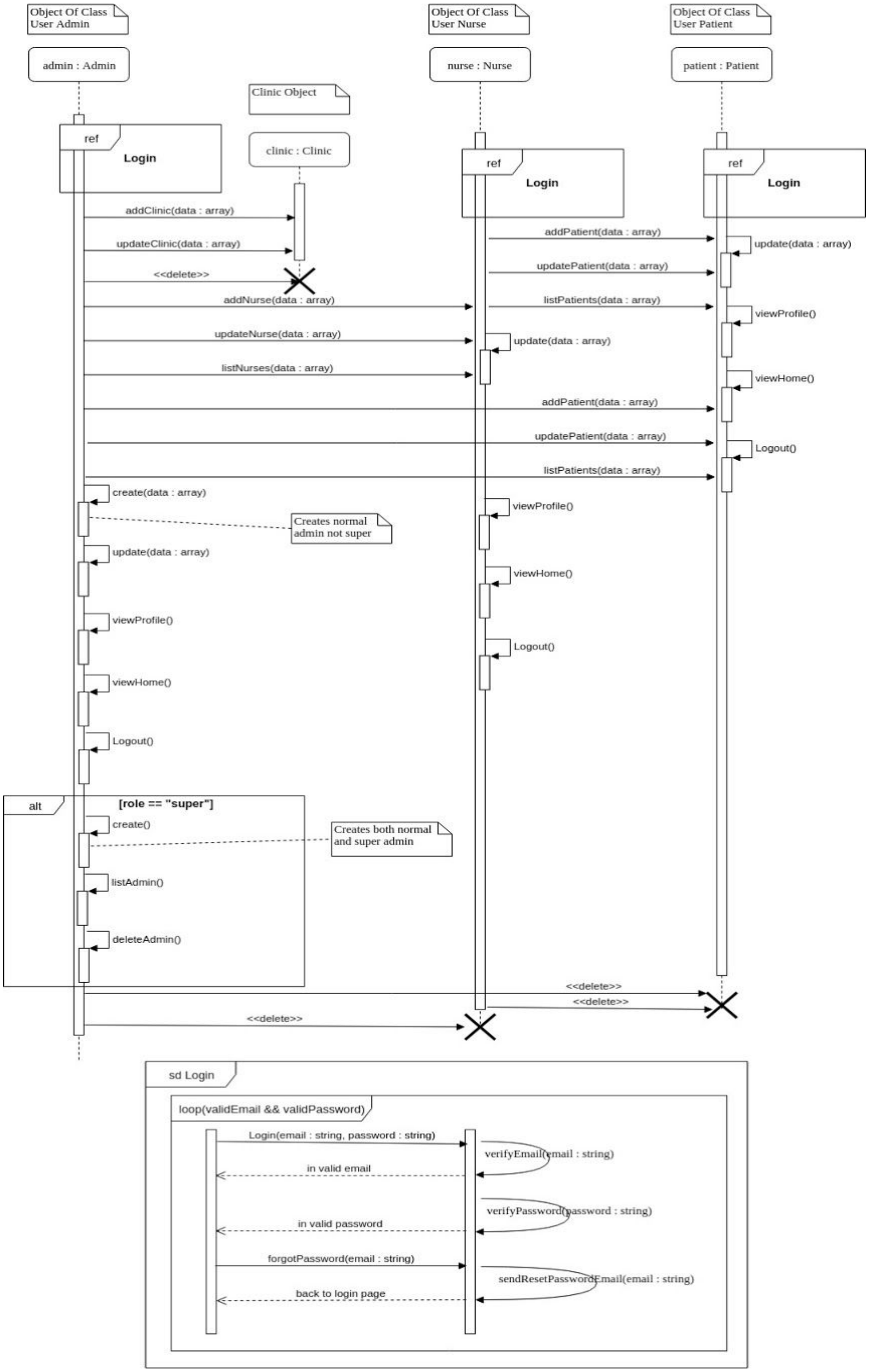
- login page: can only access forgot password controller
- send reset email page: can only access forgot password controller
- reset password page: can only access reset password controller
- home page: can access patient controller

**Guest**

- index page: accesses patient's home and login controllers as a normal guest is a logged out patient



## 3.3 Interaction Diagrams



The previous sequence diagram explains how different parts and subsystems interact with one another. Which could be broken down to four parts or tracks for the three different users (admin, nurse, patient), one entity (Clinic) and also a login subsequence.

## Login Subsequence

This sequence is a reference for the login sequence which follows the following steps.

- When user request to login both email and password would be required
- After obtaining email and password verify that email is valid and that user exist
- If email is valid and user exist check if email and password match
- If password matches user is logged in and sequence ends
- If password doesn't match return to login page with invalid password error
- If email was not valid then return to the login page with invalid email error
- If user request send reset password email validate email
- If email is valid send email then return to login page with email success
- If email is not valid return email not valid error

## Clinic

Clinic is not a user so it can't perform any actions on itself. That's why only admin is performing actions on it.

## Admin

One of the users and the only one of them who have access to all possible functionalities. Admin sequence could be divided into four parts. Functions applied on patients, functions applied on nurses, functions applied on clinics and functions applied on admins. All these parts could only be accessed after admin passes the login sub sequence.

### Clinic part

This section shows all functions that admins apply on clinic.

- If admin request to create a clinic then clinic data is to be provided
- Clinic data would be verified if it's valid clinic is created and success message is returned
- If clinic data is invalid then invalid data error would be returned
- If admin request to  update clinic then same as creating data must be provided
- Data would be verified if valid clinic is updated and success message is returned
- If update data is not valid invalid data error is returned
- If admin request to delete a clinic then clinic is deleted and success message is returned

### Patient part

This section shows all functions that admins apply on patient.

- If admin request to create a patient then patient data is to be provided
- Patient data would be verified if it's valid patient is created and success message is returned
- If patient data is invalid then invalid data error would be returned
- If admin request to  update patient then same as creating data must be provided
- Data would be verified if valid patient is updated and success message is returned
- If update data is not valid invalid data error is returned
- If admin request to list patients then list of all patients is returned

- If admin request to delete a patient then patient is deleted and success message is returned

**Nurse part**

This section shows all functions that admins apply on nurse.

- If admin request to create a nurse then nurse data is to be provided
- Nurse data would be verified if it's valid nurse is created and success message is returned
- If nurse data is invalid then invalid data error would be returned
- If admin request to  update nurse then same as creating data must be provided
- Data would be verified if valid nurse is updated and success message is returned
- If update data is not valid invalid data error is returned
- If admin request to list nurses then list of all nurses is returned
- If admin request to delete a nurse then nurse is deleted and success message is returned

**Admin part**

This section shows all functions that admins apply on other admins and himself.

- If normal admin request to create a normal admin then admin data is to be provided
- Admin data would be verified if it's valid admin is created and success message is returned
- If admin data is invalid then invalid data error would be returned
- If normal/super admin request to  update himself then data must be provided
- Data would be verified if valid admin is updated and success message is returned
- If update data is not valid invalid data error is returned
- If normal admin request to create a super admin then access denied error is returned
- If normal admin request to list admins then access denied error is returned
- If normal admin request to delete another admin then access denied error is returned
- If normal amin request to update another admin then access denied error is returned
- If super admin request to create super/normal admin then data is to be provided
- Admin data would be verified if it's valid admin is created and success message is returned
- If admin data is invalid then invalid data error would be returned
- If super admin request to list admins then a list of admins is returned
- If super admin request to update an admin then data is to be provided
- If admin data is valid then admin is updated and success message is returned
- If admin data is invalid then invalid data error is returned
- If super admin request to delete super/normal admin then admin is deleted and success message is returned
- If normal/super admin request viewing home page then admin is redirected to home page
- If normal/super admin request viewing profile page then admin is redirected to profile page
- If normal/super admin request logging out then admin is logged out and would need to go through login subsequence again to be able to perform any of the previous functions

## Nurse

One of the users who have similar functionality to the admin. However, it unlike admin it only can perform actions on patients and itself. All functions could only be accessed after passing the login sub sequence.

**Patient part**

This section shows all functions that nurses apply on patient.

- If nurse request to create a patient then patient data is to be provided
- Patient data would be verified if it's valid patient is created and success message is returned
- If patient data is invalid then invalid data error would be returned
- If nurse request to update patient then same as creating data must be provided
- Data would be verified if valid patient is updated and success message is returned
- If update data is not valid invalid data error is returned
- If nurse request to list patients then list of all patients is returned
- If nurse request to delete a patient then patient is deleted and success message is returned

**Nurse part**

This section shows all functions that nurses apply on nurse.

- If nurse request to update herself then data must be provided
- Data would be verified if valid nurse is updated and success message is returned
- If update data is not valid invalid data error is returned
- If nurse request viewing home page then nurse is redirected to home page
- If nurse request viewing profile page then nurse is redirected to profile page
- If nurse request logging out then nurse is logged out and would need to go through login subsequence again to be able to perform any of the previous functions

### Patient

The last user who can only perform functions on himself that could only be accessed after passing the login sub sequence.

- If patient request to update himself then data must be provided
- Data would be verified if valid patient is updated and success message is returned
- If update data is not valid invalid data error is returned
- If patient request viewing home page then patient is redirected to home page
- If patient request viewing profile page then patient is redirected to profile page
- If patient request logging out then patient is logged out and would need to go through login subsequence again to be able to perform any of the previous functions

## 3.4 Component Diagrams

This diagram explains different parts and subsystems and what functionalities they provide or require. We have four components Admin, Nurse, Patient and Clinic.

### Admin

we start with admin component which represents admin entity. Admin doesn't require anything as it's the provider for all other components. That's because admin is responsible for creating, updating or deleting other components. Admin also can provide functions for himself. We could divide these relations to three parts Admin provide patient, Admin provide nurse, Admin provide clinic and Admin provide admin.

**Admin provide patient**

admin provide patient with three required functions

1. create patient: since patient as entity can’t create itself so it needs other entities to create it
2. update patient: admin can provide update for patient
3. delete patient: admin can delete patient because patient can’t delete himself

**Admin provide nurse**

admin provide nurse with three required functions

1. create nurse: same as patient nurse can’t create itself so it needs admin to create it
2. update nurse: admin can provide data update for nurse
3. delete nurse: admin provides deleting for nurse because nurse also can’t delete herself

**Admin provide clinic**

admin provide clinic with three required functions

1. create clinic: clinic is not a user so it can’t create itself thus admin provides creation for it
2. update clinic: admin provides update for clinic
3. delete clinic: same as nurse and patient admin also provide deletion for clinic

**Admin provide admin**

Apart from all that admin also provides some extra functions that works specifically on himself.

1. create admin: admin can create other admins
2. update admin: admin can update himself and other admins
3. delete admin: admin can delete other admins

## Nurse

Second we have nurse component which represents nurse entity. Nurse requires some admin functionality that is mentioned in the admin section above and provides patient functionalities and functionalities for herself. So we have two parts Nurse provide patient and Nurse provide nurse.

**Nurse provide patient**

nurse provide patient with three required functions

1. create patient: since patient as entity can’t create itself so nurse provide creation for it
2. update patient: nurse can provide update for patient
3. delete patient: nurse can delete patient because patient can’t delete himself

**Nurse provide nurse**

same as admin nurse also can provide herself with some functionality. However, unlike admin nurse can only do one function on herself

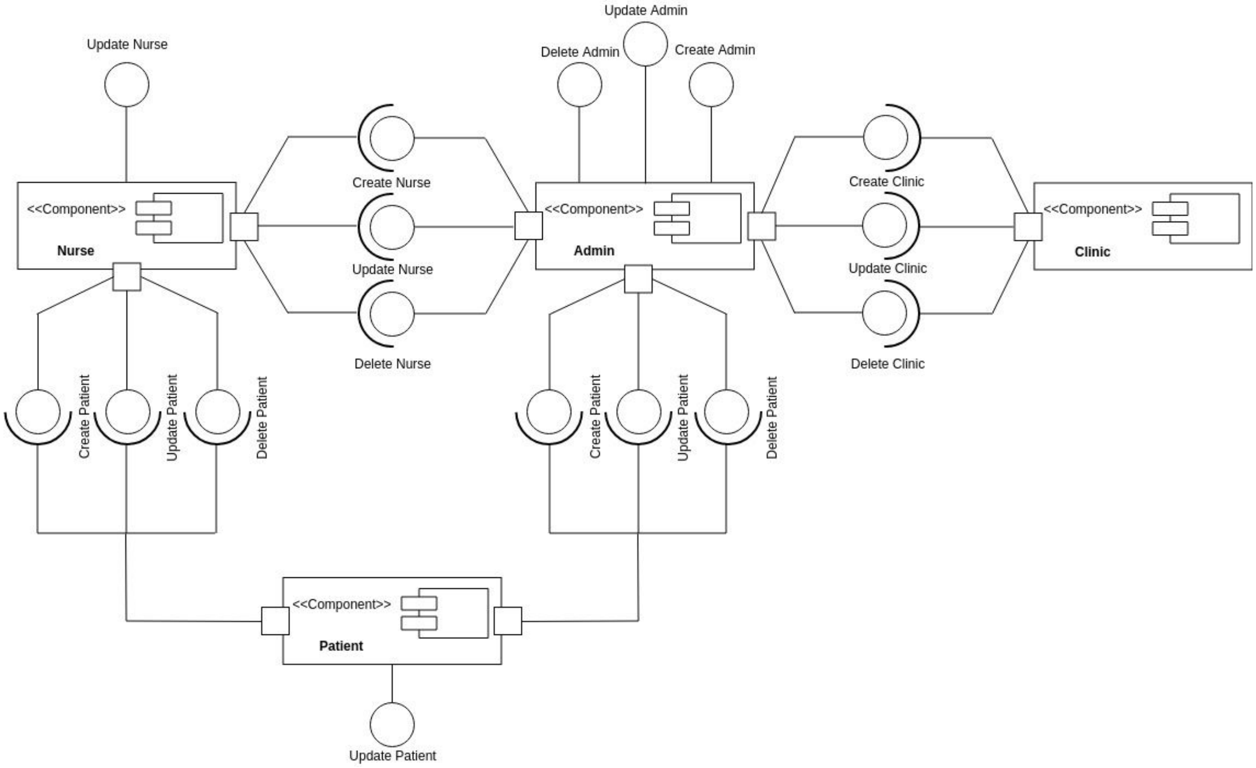
1. update nurse: nurse can update only herself

## Patient

The third component is patient component which represents patient entity. Patient requires some admin functionality and equivalent nurse functionalities which are mentioned in the admin and nurse sections above and provides a single update functionality for himself to update his data.

### Clinic

The last component that represents clinic entity. This component doesn’t provide any functionalities however, requires the three functionalities provided by the admin component in the admin section above.



# 4. Data Models

| | Nurse |
|---|---|
| PK | id |
| | name |
| | email |
| | mobile |
| | password |
| | image |
| | status |
| | date_of_birth |
| | salary |
| FK | clinic_id |

| | Admin |
|---|---|
| PK | id |
| | name |
| | email |
| | mobile |
| | password |
| | image |
| | role |
| | date_of_birth |

| | Patient |
|---|---|
| PK | id |
| | name |
| | email |
| | mobile |
| | password |
| | image |
| | status |
| | date_of_birth |
| | gender |

| | Worker |
|---|---|
| PK | id |
| | name |
| | mobile |
| | position |
| | date_of_start |
| | date_of_birth |
| | salary |
| FK | clinic_id |

| | Clinic |
|---|---|
| PK | id |
| | name |
| | email |
| | address |
| | telephone |
| | start_time |
| | end_time |

| | Comment |
|---|---|
| PK | id |
| | content |
| FK | patient_id |
| FK | admin_id |

| | Image |
|---|---|
| PK | id |
| | image |
| | caption |
| FK | patient_id |
| FK | admin_id |

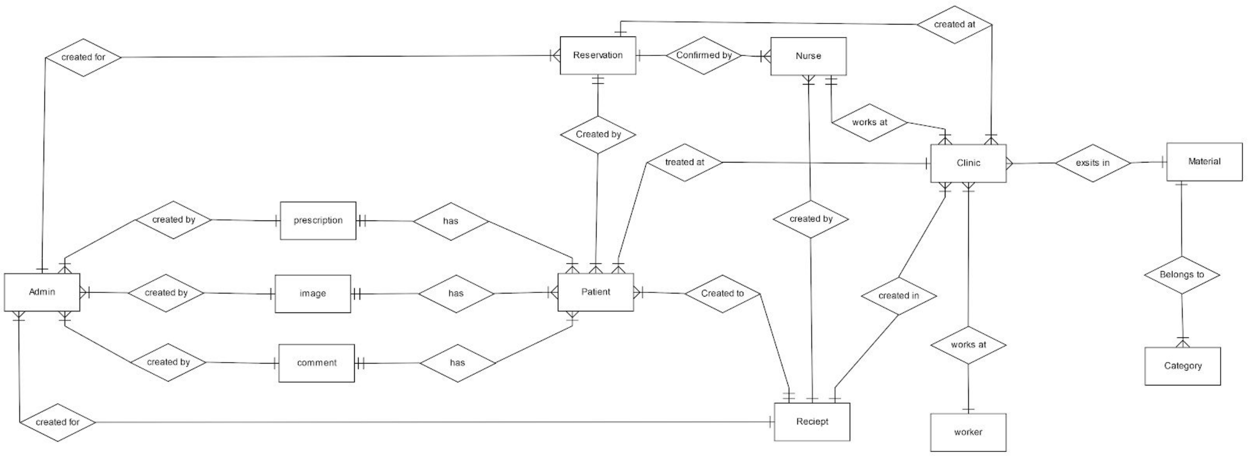
| | Material |
|---|---|
| PK | id |
| | name |
| FK | clinic_id |
| FK | category_id |
| | num |
| | cost |

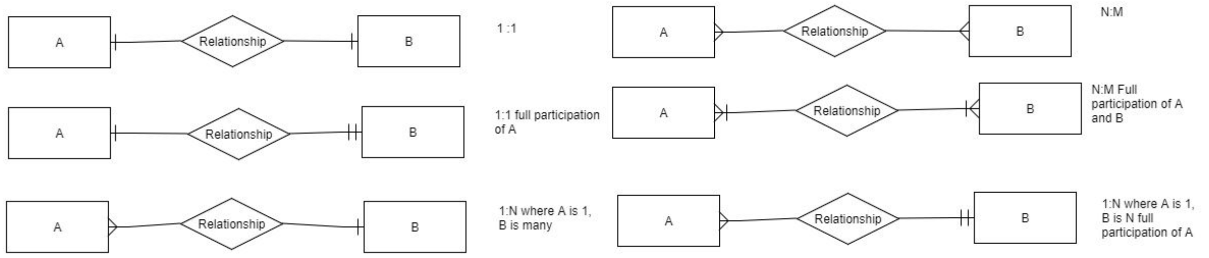
| | Receipt |
|---|---|
| PK | id |
| FK | patient_id |
| FK | admin_id |
| FK | nurse_id |
| FK | clinic_id |
| | day |
| | total_price |

| | Prescription |
|---|---|
| PK | id |
| | name |
| FK | patient_id |
| FK | admin_id |

| | Category |
|---|---|
| PK | id |
| | name |

| | Reservation |
|---|---|
| PK | id |
| FK | patient_id |
| FK | admin_id |
| FK | nurse_id |
| FK | clinic_id |
| | time |
| | attend |




A
Relationship
B
1 :1
A
Relationship
B
N:M
A
Relationship
B
1:1 full participation of A
A
Relationship
B
N:M Full participation of A and B
A
Relationship
B
1:N where A is 1, B is many
A
Relationship
B
1:N where A is 1, B is N full participation of A

# 5. System Deployment

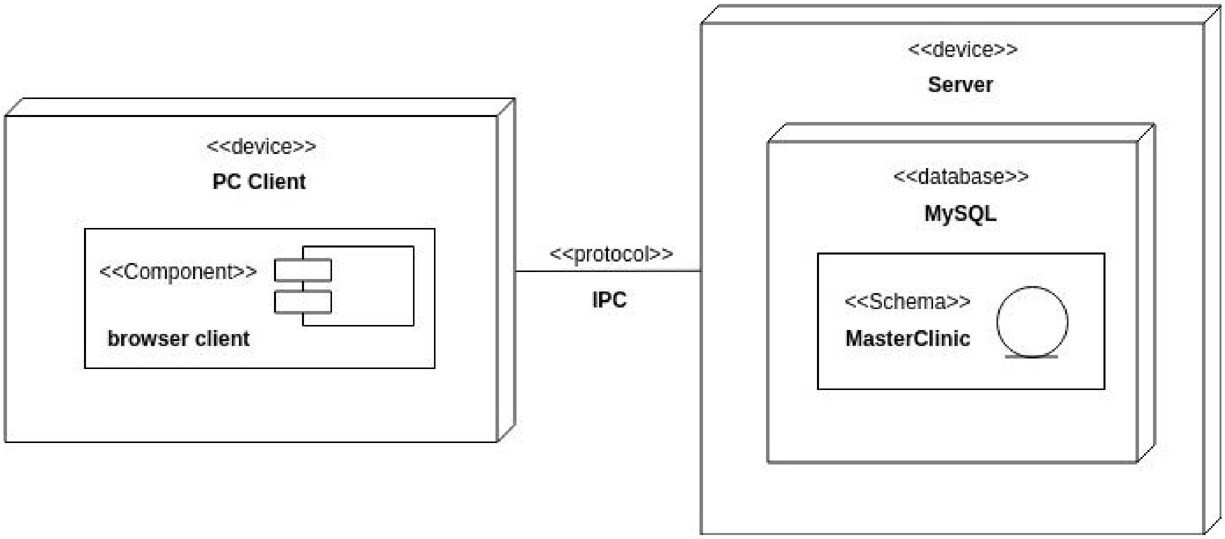
Component diagram (*see section 3.4*) shows the system software integrity as different components of software interact with one another. The following diagram is the deployment diagram which is shows the hardware deployment and integrity of the system. Following the two tier client server model the system would be divided into two tiers client tier and server tier.

## Client tier

This tier is deployed in the users side. It contains views that interact with controllers on server through a browser. This tier sends data and different requests to controllers on the server side.

## Server tier

This tier is deployed in the server side. It contains controllers, models and database. This tier receives data and requests from the client side. The connection protocol between the two tiers is a normal interprocess communication (**IPC**).



# 6. Traceability to Requirements

| RID | SD1 | SD2 | SD3 | SD4 | SD5 | SD6 | SD7 |
|---|---|---|---|---|---|---|---|
| US1 | | | | | | | |
| US3 | | | | | | | |
| US4 | | | | | | | |
| US11 | | | | | | | |
| US13 | | | | | | | |

*<This should maintain a matrix similar to the Traceability Matrix in the Software Requirements Specification document. The purpose of this matrix is to map requirements to design elements. This will immediately point out requirements that have not been considered. It will also allow you to trace into the design elements that require to be changed if requirements change. Use requirement IDs as row headers and design element IDs as column headers>*