We Care

# The Disease Tracker System
# Software Architecture Document

Version 1.1

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 09/04/16 | 1.0 | Initial software Architecture Document | K.N.A. Jayatissa |
| 02/05/16 | 1.1 | Updated software Architecture Document | K.N.A. Jayatissa |
| | | | |
| | | | |

# Table of Contents

# Software Architecture Document

## 1. Introduction

This document will give a higher level overview of the architecture of the Disease Tracker System and different perspectives of the system design will be illustrated

### 1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

### 1.2 Scope

This is a representation of the architecture of the Disease Tracker system, which allows the developer to document the system design that would be based for system implementation. Furthermore, external parties like clients and other stakeholders can get a clear idea about how the system is designed and implemented by viewing this document. Therefore they can understand the project as a whole with fewer efforts.

### 1.3 Definitions, Acronyms, and Abbreviations

| Term | Description |
|---|---|
| User | Someone who interacts with the web application |
| Patient | Someone who uses the system to diagnose a disease, to find doctors. |
| Admin/ Administrator | System administrator who has given specific permission to manage and control the system |
| MVC | Model View Controller |

### 1.4 References

[1] "Software Requirement Specification".

[2] "Project Proposal".

### 1.5 Overview

The structure of this document is organized as follows, in order to fully document all the aspects of the architecture.

- Architectural Representation
- Architectural Goals and Constraints
- Use-case view
- Logical view
- Process view
- Deployment view
- Implementation view
- Performance
- Quality

Each of the above areas will give a properly detailed view of the Disease Tracker System.

## 2. Architectural Representation

This document takes a detailed look at the architecture of the Disease Tracker System using several views. The following views have been used to illustrate the system architecture

- Logical View:

This will give an overview of how the system will be structured and this model will be used as a design model to better understand the system

- Process View:

This will illustrate how the process will flow within the system. This will look in depth to how the internal process happens and how each task is related with each other.

- Implementation View:

This will show the necessary components to make the system work and how they are bundled together. This will be very useful at the implementation stage of the system.

- Deployment View:

This will look at the sort of environment the finished system will have to be deployed in. This will show the environment specifics that needs attention.

- Use-case View:

This view is important for all the stakeholders of the system including the end-users to understand the system. This view describes the set of scenarios and use cases that represent some significant, central functionality of the system.

- Data view (optional):

Data view is important for data handling used by the system. This describes the architecturally significant persistent elements in the data model.

## 3. Architectural Goals and Constraints

The following are the key requirements and system architecture that have a significant impact on the architecture.

- System usage is controlled by user identification and password control. All the user details must be protected from unauthorized access.
- System should give the most accurate disease for the symptoms the user provides considering other user details like, age, pre- diagnosed diseases, etc.
- All performance and loading requirements must be taken into consideration as the architecture is being developed.
- Availability of the system should at least be 98%.

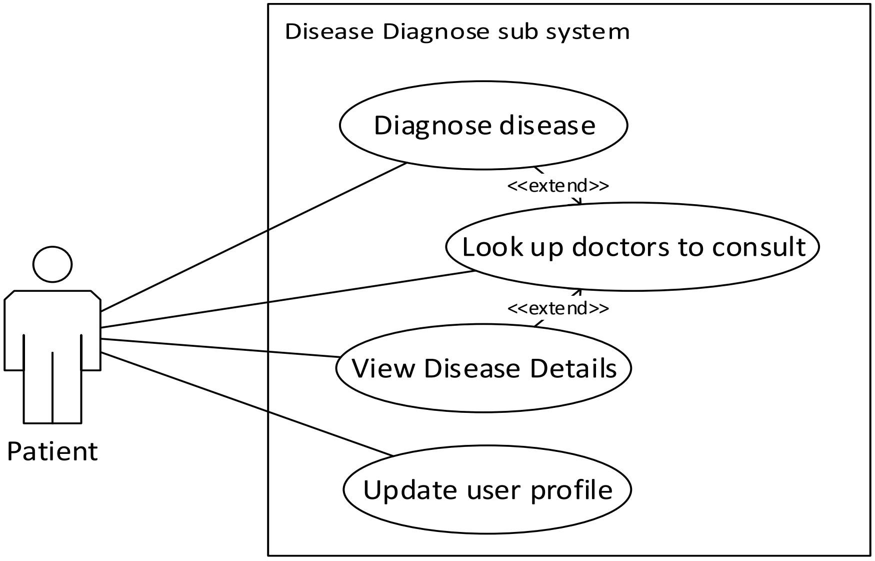
## 4. Use-Case View

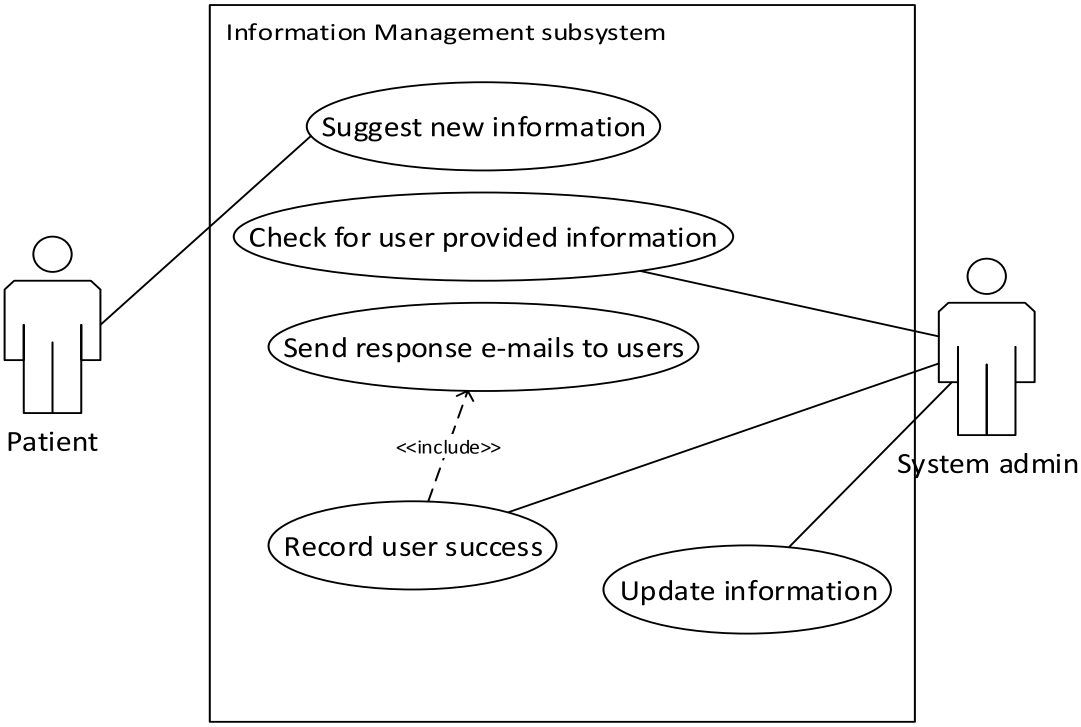
Use case view is to define main drivers of the system, which are the system requirements that stress or illustrate a specific, delicate point of the architecture or that have a substantial architectural coverage.

### 4.1 Use-Case Realizations

#### *4.1.1 Use Case Diagram*

A use case diagram was constructed in order to graphically represent the flow of events in the system. The two sub systems, disease diagnosis sub system and the information management subsystem have their specific use cases which show their specific functionalities and how the actors interact to the system.




Information Management subsystem
Suggest new information
Check for user provided information
Send response e-mails to users
<<include>>
Record user success
Update information
Patient
System admin

### 4.1.2 Diagnose Disease

| | |
|---|---|
| **Use case name** | Diagnose Diseases |
| **Actor** | Patient (user) |
| **Description** | Patient input the symptoms to diagnose the disease through the system. |
| **Preconditions** | Patient should have logged- into the system.<br>Patient profile should be up-to-date. |
| **Main flow** | Patient input the symptoms he/she has got. Then the system analyzes the symptoms along with the user details and medical history and give the most accurately guessed disease he/she may have got. |
| **Successful end/post condition** | The most accurate guess for the disease is given as the output |
| **Fail end/post condition** | No result found for the provided symptoms |
| **Extensions** | Look up doctors to consult |

### 4.1.3 Look up doctors to consult

| | |
|---|---|
| **Use case name** | Look up for doctors to consult |
| **Actor** | Patient (user) |
| **Description** | Patient search for doctors to consult for a particular disease. |
| **Preconditions** | Patient should have logged- into the system. |
| **Main flow** | Patient selects the disease from the drop down menu and click on the “Find Doctors” button. Then the system give the details of the doctors specialized for that particular disease as the output. |
| **Successful end/post condition** | System will give a list of appropriate doctors and their details. |
| **Fail end/post condition** | System database have no information for the disease the user wants to find about. |
| **Extensions** | N/A |

### 4.1.4 Suggest Information

| | |
|---|---|
| **Use case name** | Suggest Information |
| **Actor** | Patient (user) |
| **Description** | User provides the information related to a disease which does not available in the system. |
| **Preconditions** | User should have logged- into the system. |
| **Main flow** | User input the details in the provided format. |
| **Successful end/post condition** | The user provided information is accurate and valuable and is added to the system database. |
| **Fail end/post condition** | Provided details are false. |
| **Extensions** | N/A |

### 4.1.5 View user provided information

| | |
|---|---|
| **Use case name** | View user provided information |
| **Actor** | System Admin |
| **Description** | System admin check view the latest information provided by the users. |
| **Preconditions** | Admin should have logged into the system.<br>Users should have provided information |
| **Main flow** | System admin goes to "User Suggestions" page and view the information provided by the users.<br>Then carries on the accuracy check for all those information and decides whether to add to the system database. |
| **Successful end/post condition** | Information will be added to the system. |
| **Fail end/post condition** | Provided details are false |
| **Extensions** | N/A |

### 4.1.6 Record user contribution success

| | |
|---|---|
| **Use case name** | Record user contribution success |
| **Actor** | System admin |
| **Description** | System admin records the success of the provided info with the corresponding user identification. |

| | |
|---|---|
| | System sends response emails to corresponding users stating the success of their contribution. |
| **Preconditions** | System admin should have logged into the system.<br>Users have suggested information and the accuracy check have been done for those information. |
| **Main flow** | System Admin do the accuracy check for all information provided by the users.<br>Admin sends response emails to corresponding users who have provided information stating the accuracy of their information. |
| **Successful end/post condition** | The email will be sent to the corresponding user. |
| **Fail end/post condition** | The email will not be sent to the corresponding user. |
| **Extensions** | N/A |

### *4.1.7 Manage Information*

| | |
|---|---|
| **Use case name** | Manage Information |
| **Actor** | System Admin |
| **Description** | System admin adds, deletes or updates information in the system database based on the latest findings. |
| **Preconditions** | System admin should have logged into the system.<br>New information to be added to the system is available or a need to update or delete some information has aroused. |
| **Main flow** | Admin selects the needed action to be done and the information to be reviewed.<br>Does the necessary changes to the information and saves the changes. |
| **Successful end/post condition** | Database update is successful. |
| **Fail end/post condition** | Database update is not successful |
| **Extensions** | N/A |

# 5. Logical View

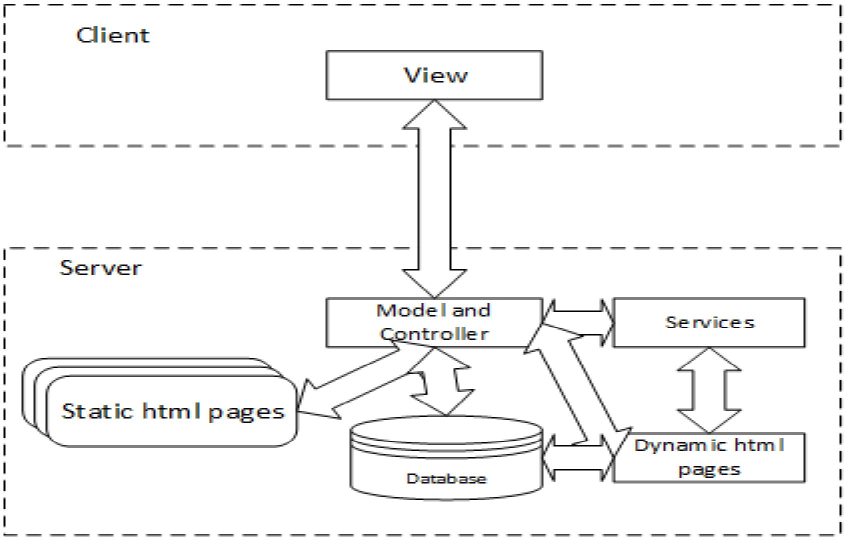
Logical view discloses the architecturally significant parts of the design model including the decomposition into subsystems and packages. It describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers.

The logical view of the disease tracker system is comprised of the 3 main packages: View, Model and Controller working in client server architecture.

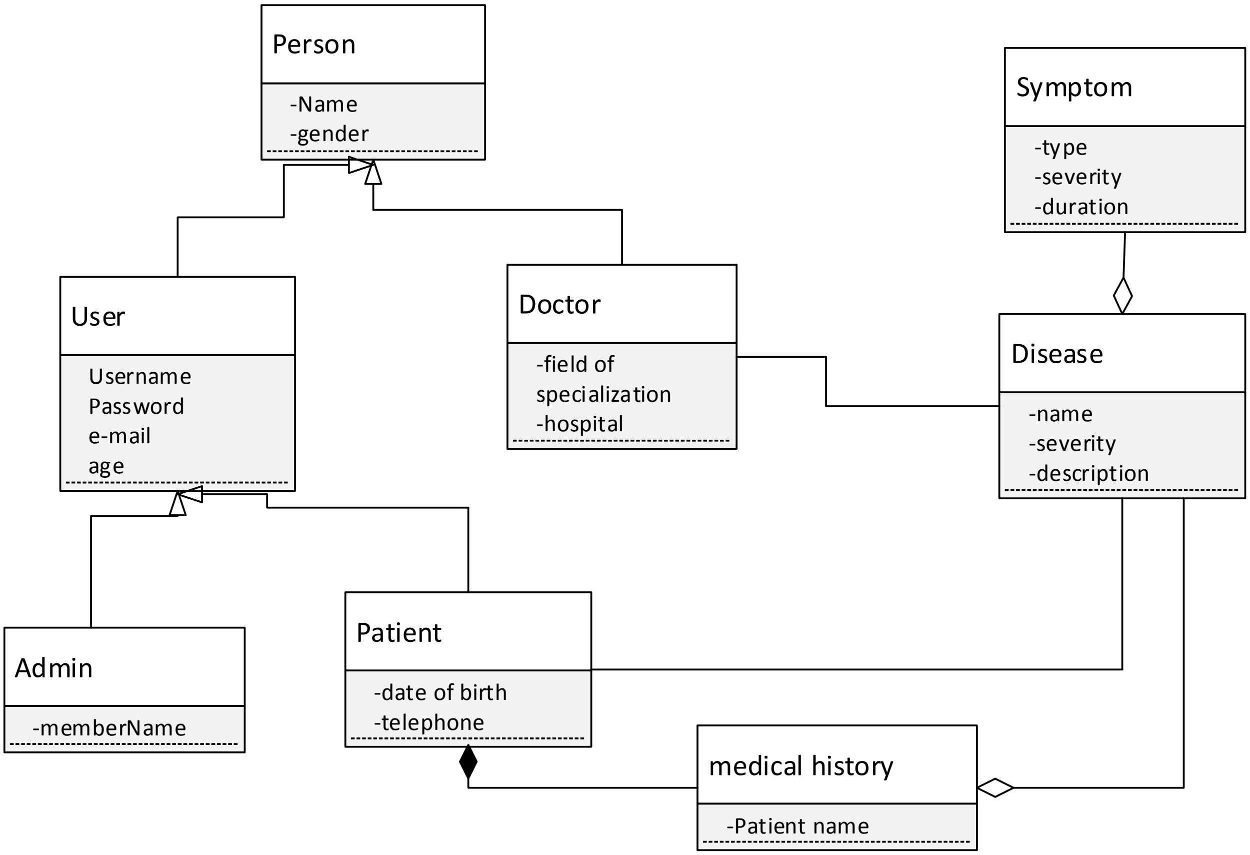
The View Package contains classes for each of the web pages that the actors use to communicate with the System. The Controller Package contains control classes for interfacing with the view package and the model package. The Model Package includes data access classes for the data bases.

## 5.1 Overview

The overall decomposition of the design model in terms of its package hierarchy and layers can be seen as follows;



### 5.2 Architecturally Significant Design Packages



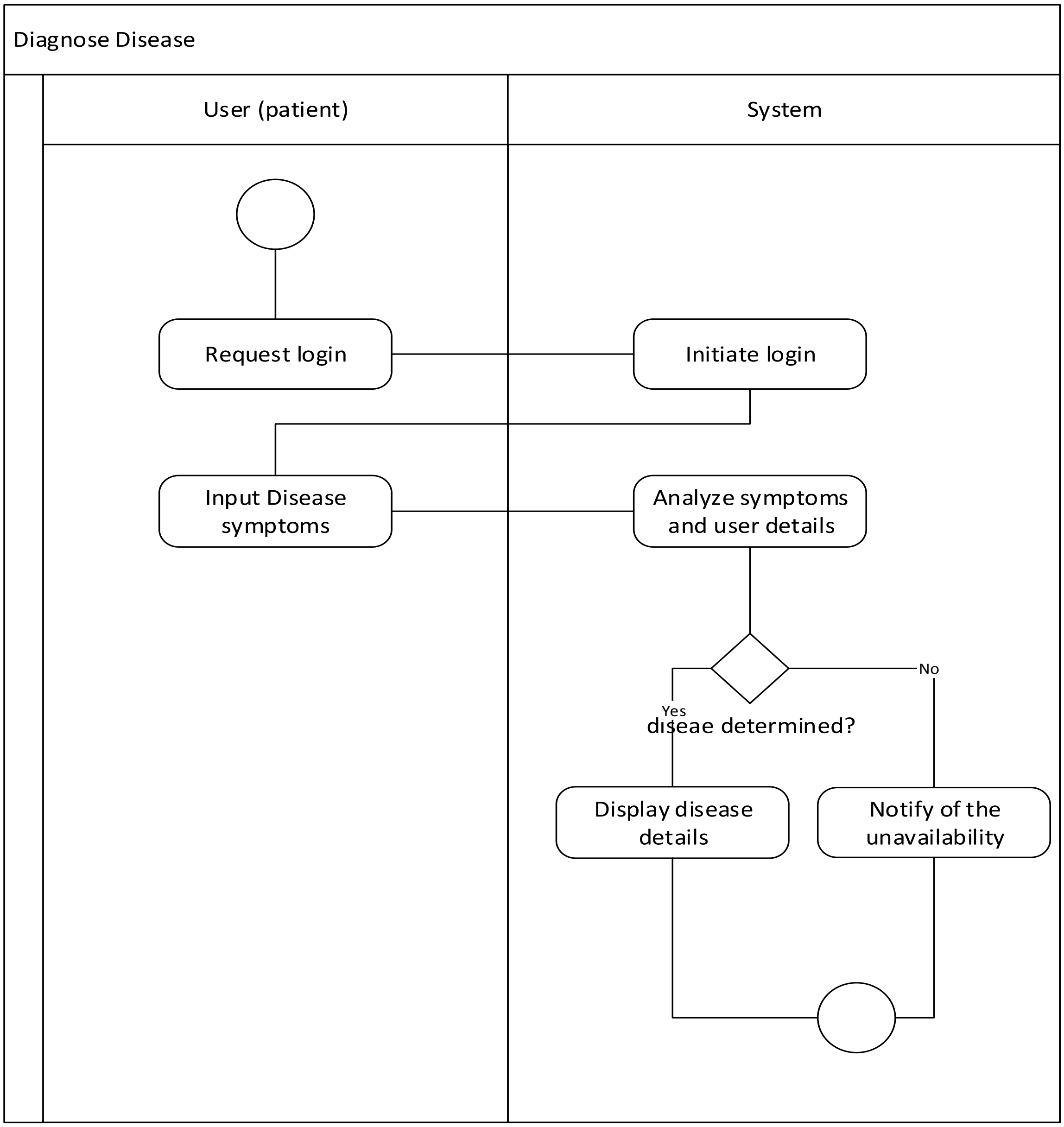
The structure of the system is also defined in terms of classes and their associations and relationships. Class diagrams with the association and relationship diagrams are made in order to generate a logical design for the system. Through the diagrams, connections between classes and their attributes are identified. This enables the thorough analysis and scrutiny of the system.

## 6. Process View

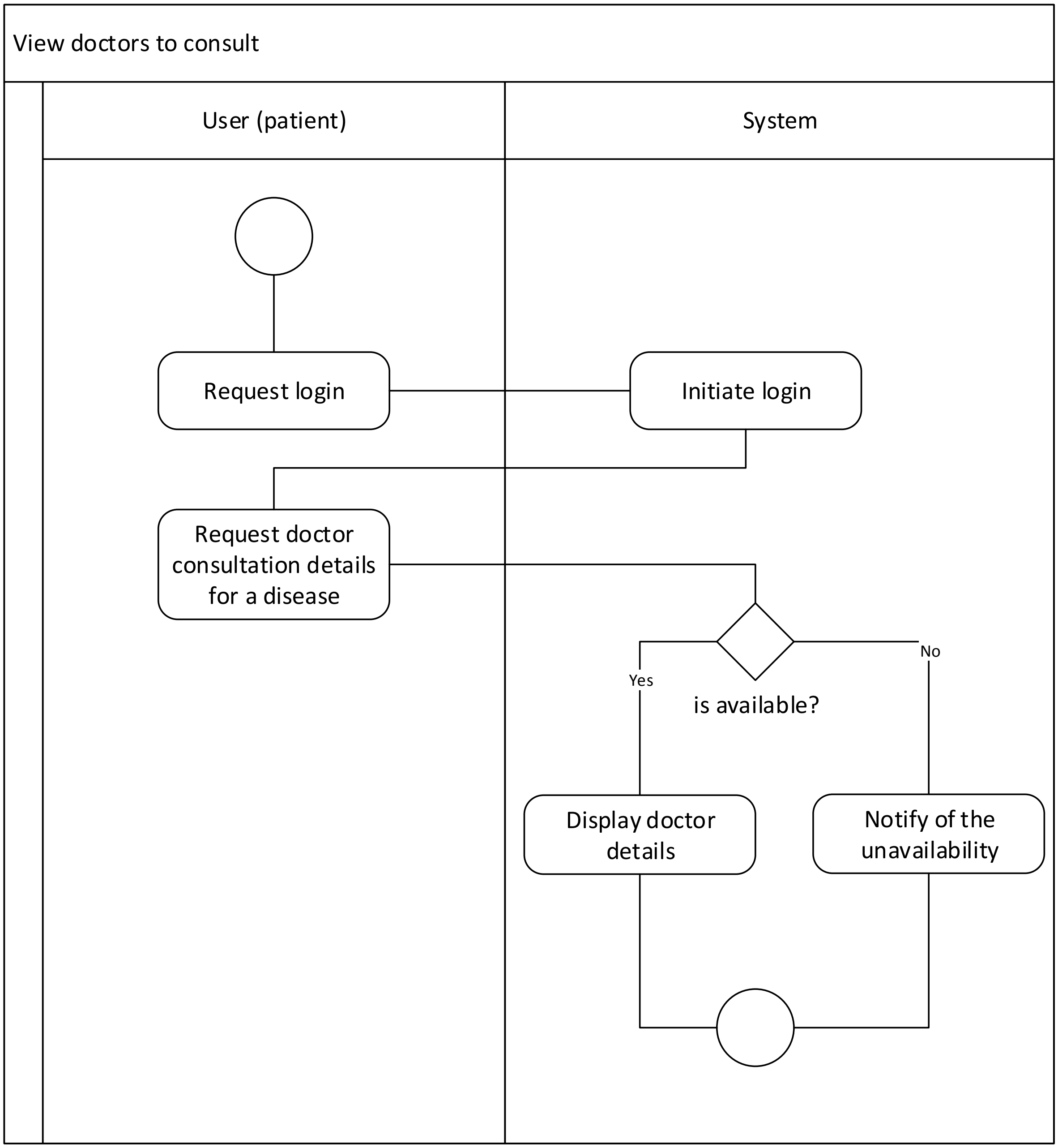
The process view will describe the system's decomposition into light weight processes and heavy weight processes as well as the forms of communication between processes, like message passing, activity between components, and message sequencing.

## 6.1 Activity Diagrams

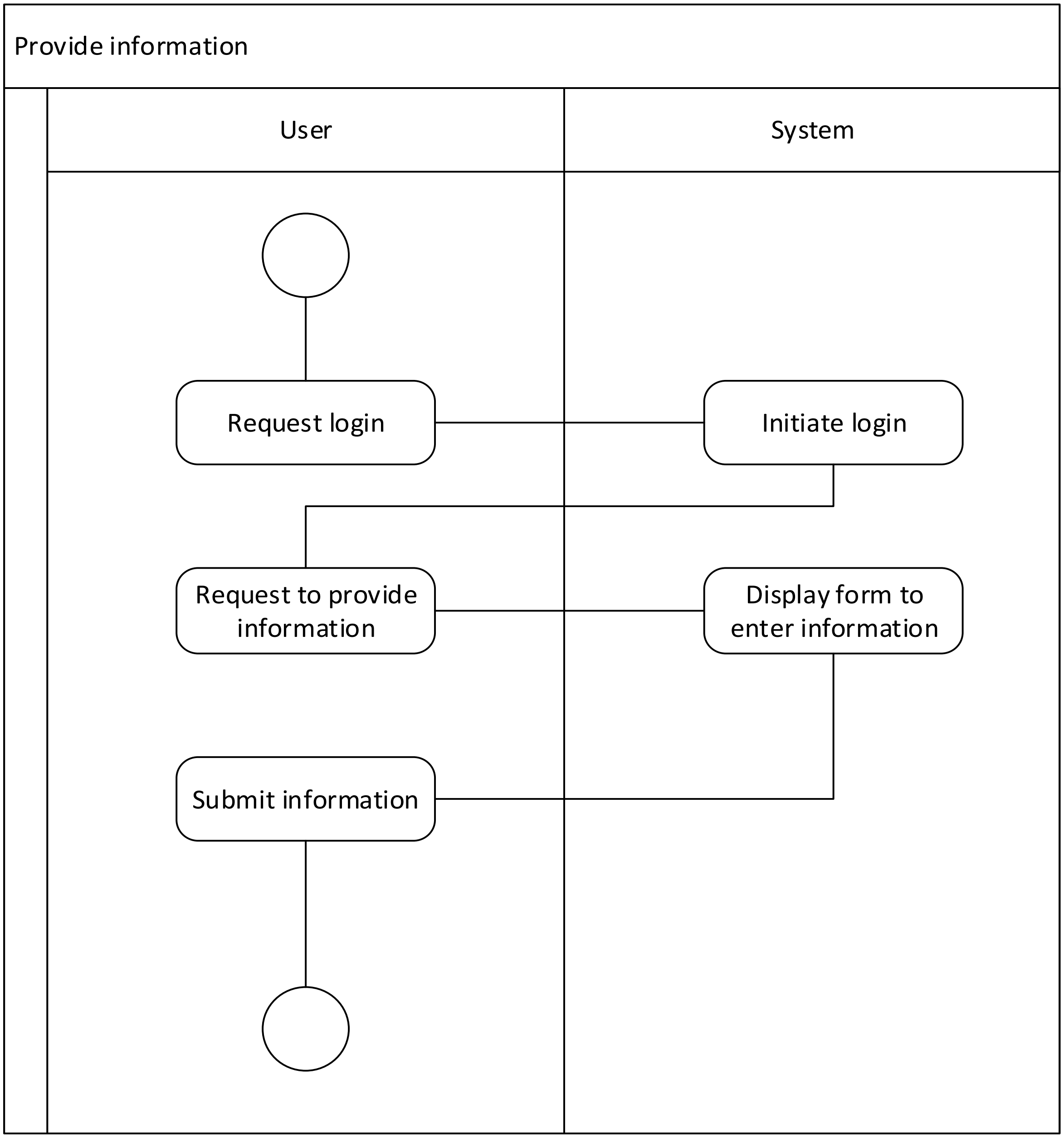
Activity diagram for the symptom checking functionality



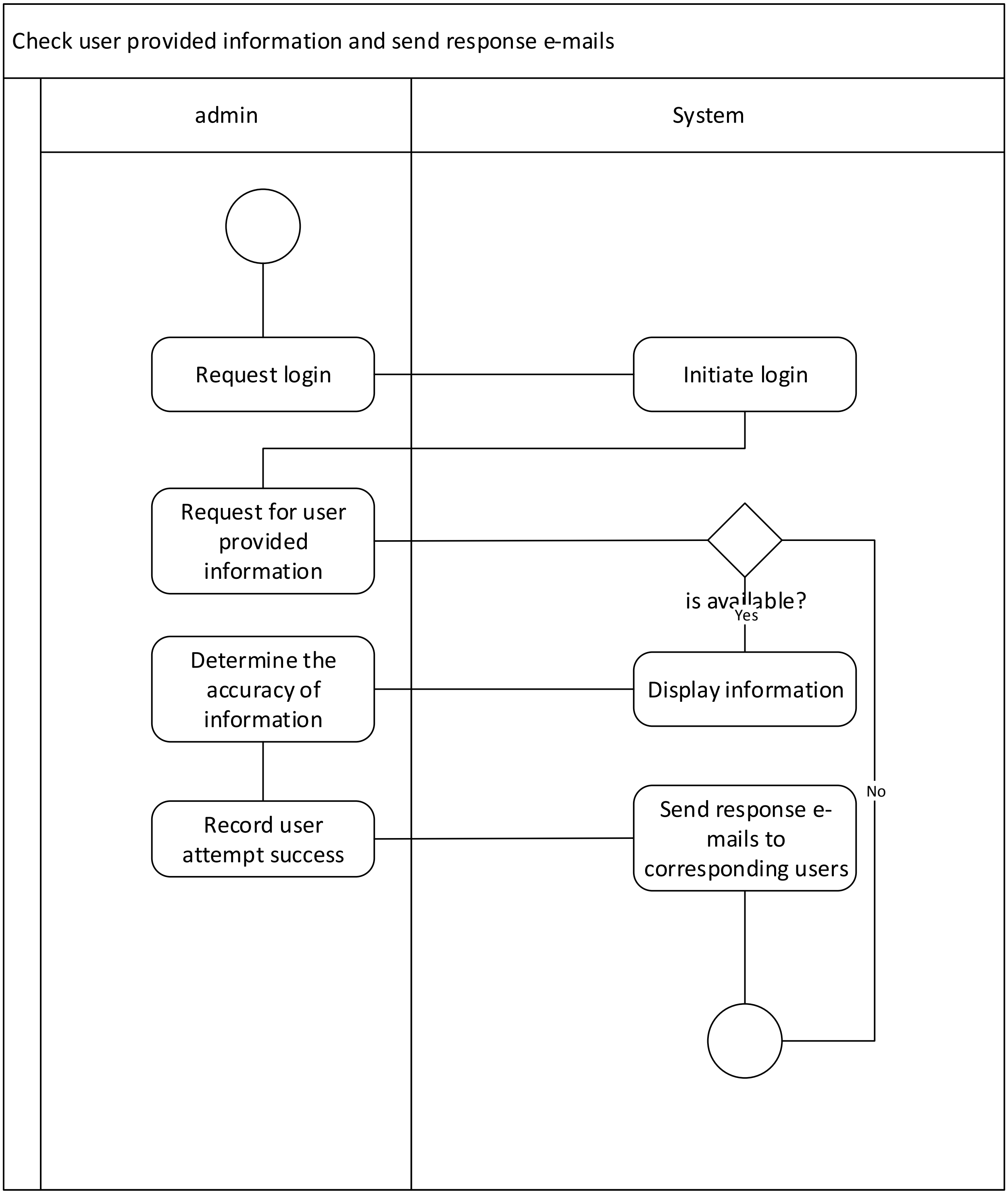
Activity diagram for viewing doctor details functionality



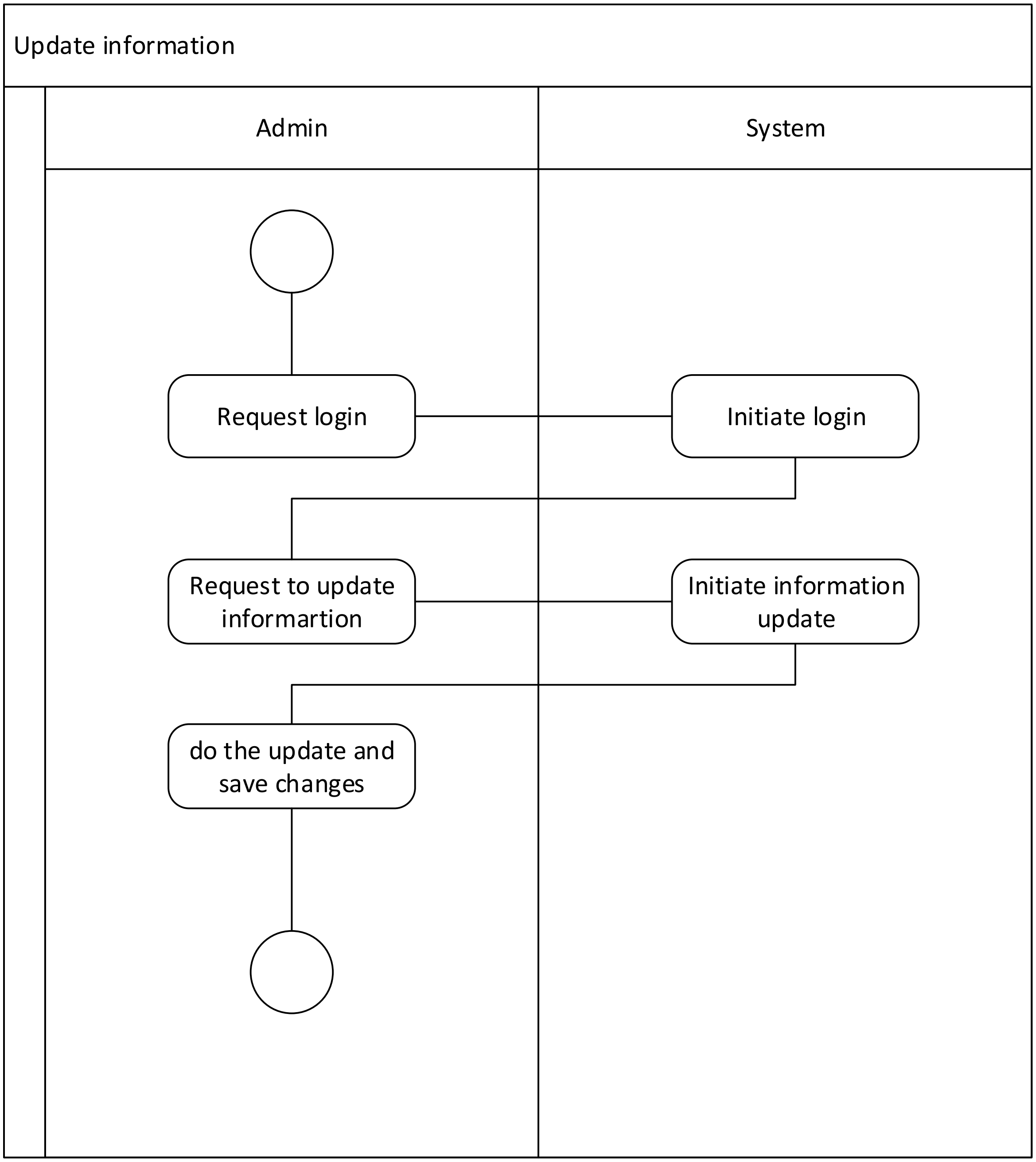
Activity diagram for user providing information functionality



Activity diagram for admin checking user provided information and sending response emails to corresponding users

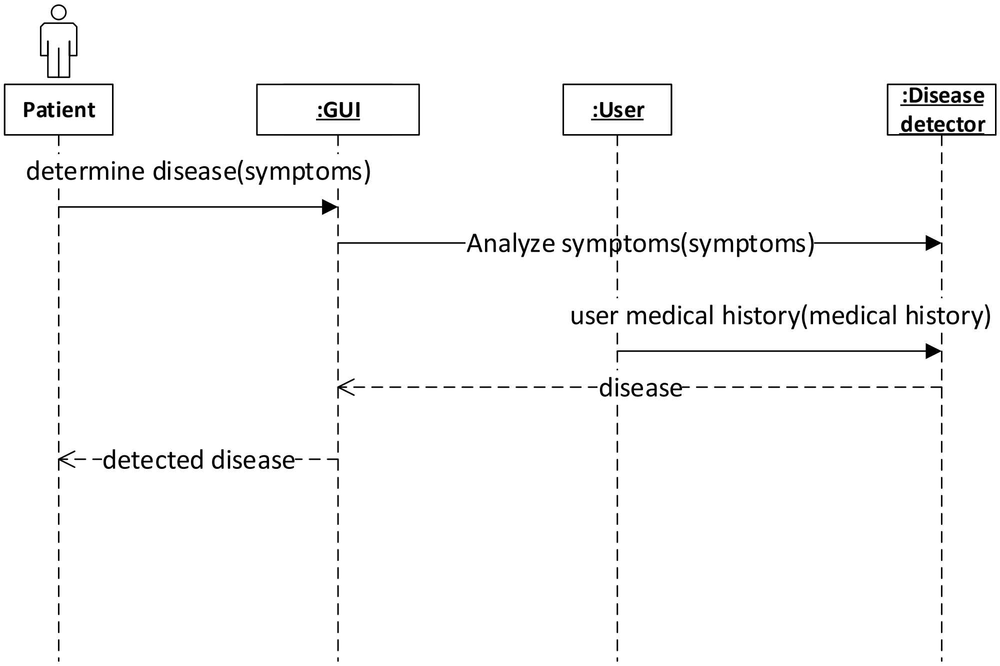

Check user provided information and send response e-mails
admin
System
Request login
Initiate login
Request for user provided information
is available?
Yes
No
Determine the accuracy of information
Display information
Record user attempt success
Send response e-mails to corresponding users

Activity diagram for updating information by admin


Update information
Admin
System
Request login
Initiate login
Request to update informartion
Initiate information update
do the update and save changes

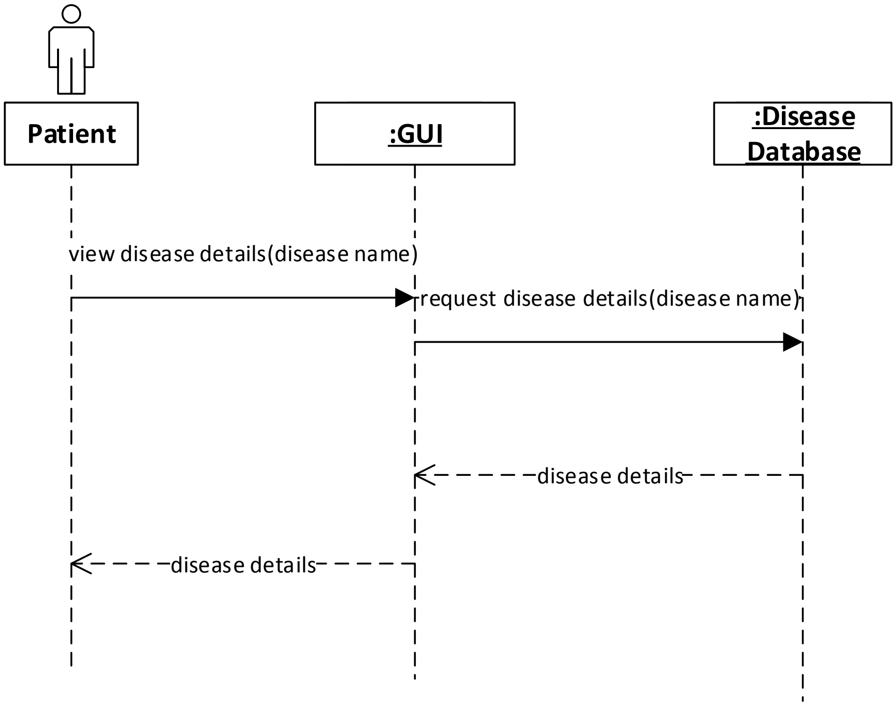
## 6.2 Sequence Diagrams

Sequence diagram for diagnosing diseases when symptoms are provided



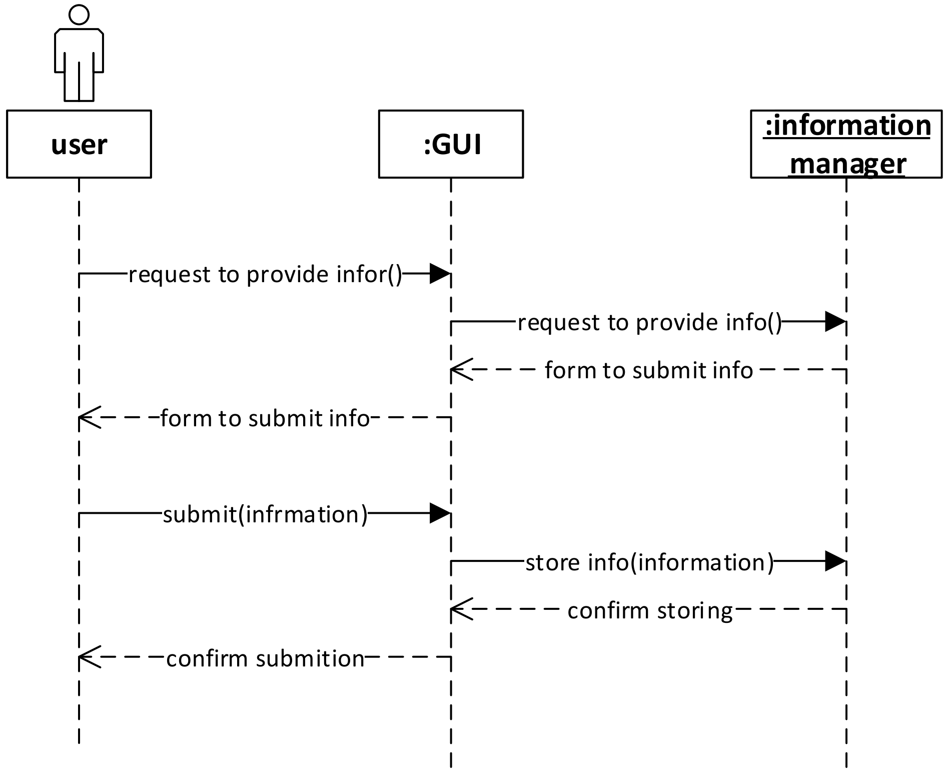
Patient
:GUI
:User
:Disease detector
determine disease(symptoms)
Analyze symptoms(symptoms)
user medical history(medical history)
disease
detected disease

Sequence Diagram for viewing details of a particular disease

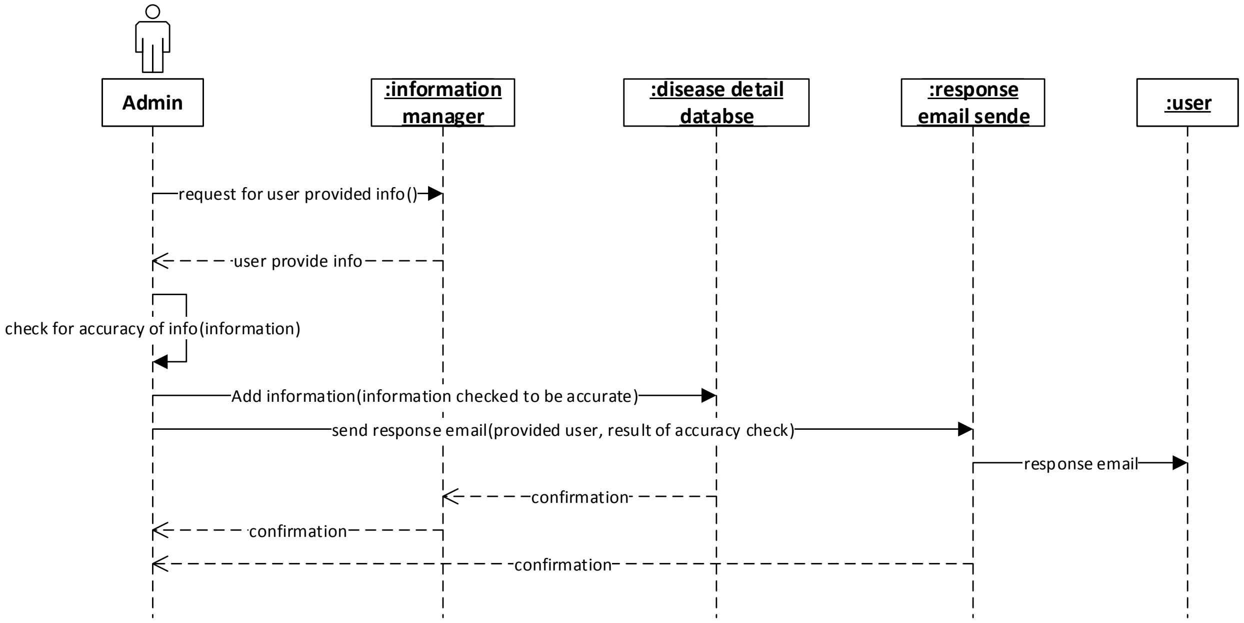


Patient
:GUI
:Disease Database
view disease details(disease name)
request disease details(disease name)
disease details
disease details

Sequence diagram for providing information by users



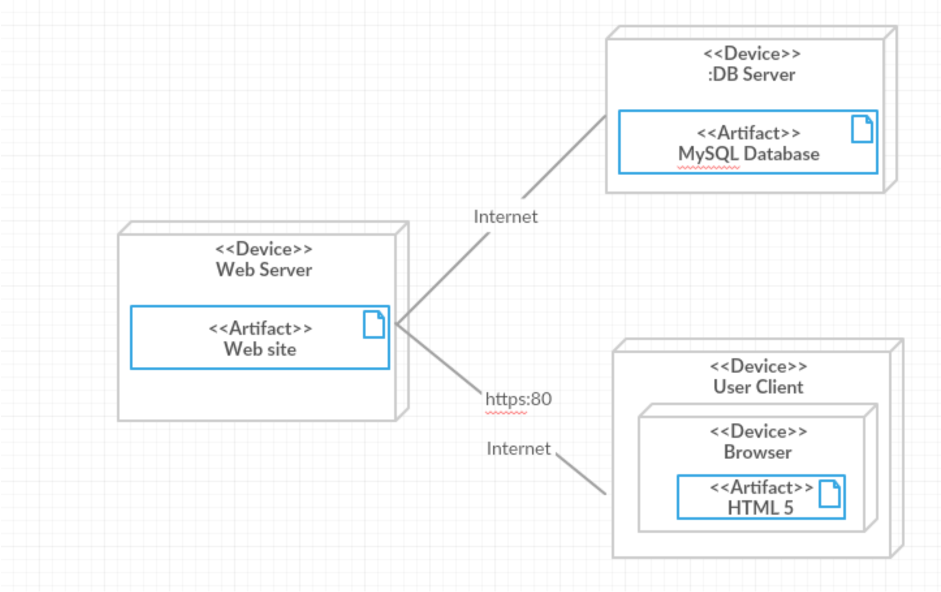
Sequence diagram for the process of validating user provided information and sending response emails to corresponding user.



Admin
:information manager
:disease detail databse
:response email sende
:user
request for user provided info()
user provide info
check for accuracy of info(information)
Add information(information checked to be accurate)
send response email(provided user, result of accuracy check)
response email
confirmation
confirmation
confirmation

# 7. Deployment View

The disease Tracker system is comprised of three physical nodes: the web server, the database server and the user browser. The deployment view of the system is as follows. All the users access the system via internet. Only the registered users will be allowed to use the system. According to the user role, the access privileges will be granted.



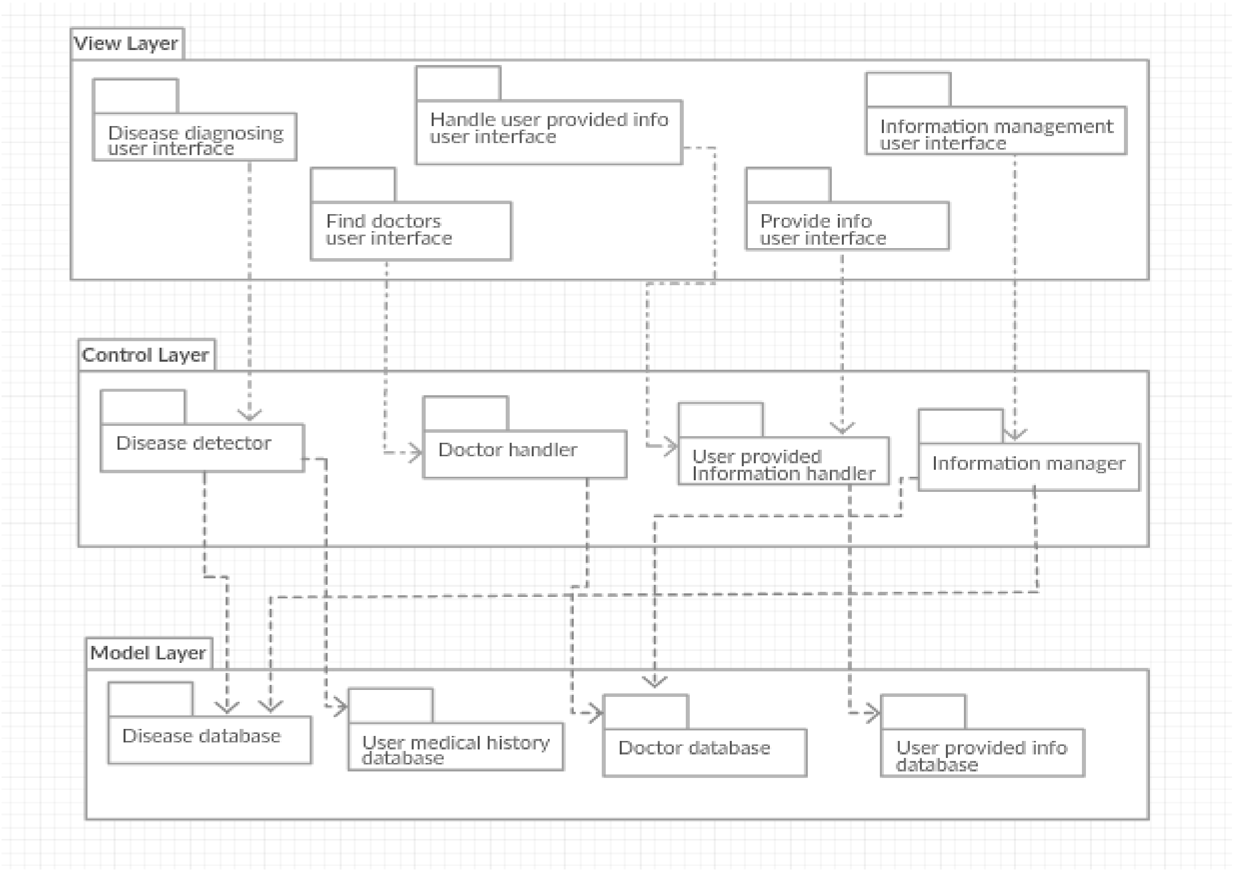
# 8. Implementation View

The implementation view if focused on the overall structure of the implementation model and the decomposition of the software into layers and sub systems. The disease tracker system is basically divided into 2 sub-systems; diagnosis sub system and information management sub system. The implementation view uses the MVC architecture in sub dividing the system into three layers; view layer, model layer and control layer.

## 8.1 Overview

The layers applied to the implementation view are the model layer, control layer and the view layer of MVC architecture.

### 8.1.1 *Package diagram*


View Layer
Disease diagnosing user interface
Handle user provided info user interface
Information management user interface
Find doctors user interface
Provide info user interface
Control Layer
Disease detector
Doctor handler
User provided Information handler
Information manager
Model Layer
Disease database
User medical history database
Doctor database
User provided info database

## 8.2 Layers

### 8.2.1 *The model layer*

The main function of the model layer is managing the data of the application. It responds to the request from the view layer and also to the instructions from the controller to update itself. The model layer includes model layer components of all two subsystems.

### 8.2.2 *The control layer*

The control layer is responsible for responding to user input and performs interactions on the data model objects. The controller receives the input; it validates the input and then performs the business operation that modifies the state of the data model. It also includes the controlling components of all the subsystems; the diagnosis subsystem and the information management subsystem.

### 8.2.3 *The view layer*

The view layer consists of the user interfaces of all the subsystems; the diagnosis subsystem and the information management subsystem.

## 9. Data View

The system widely uses the data storage to store the data of the users, diseases and doctors. The security of the data storage is crucial in the system since the accuracy of the stored data is significant. The data model should handle all the stored data efficiently. The database is intended to be implemented using MySQL database.

## 10. Size and Performance

Responsiveness of the UI elements:

The user interface must be responsive to keep the user engaged.

Detection and conversion time:

The time taken for the disease to be diagnosed and doctors to be found must be considerably low in order to make the product usable to the user.

Accuracy and consistency:

The system should diagnose the disease accurately and consistently to make the product useful.

## 11. Quality

Reliability-

The Disease Tracker system must be reliable such that every time the system diagnose a disease, it should provide consistent results with a low rate of errors.

Extensibility-

The application developed must be easily extendable and easily adaptable to add new functions and features.

Understandable-

The user interfaces must be clutter free and easy to follow for the users.