# MedLife

## Software Requirement Specification

**GROUP 8:**
Bipul Kumar (2014CS50282)
Kartar Singh (2014CS50286)
Prachi Singh (2014CS50289)
Gulshan Jangid (2014CS50285)

# Table of Contents

# 1. Introduction:

In daily based scenario, if a doctor recommends a patient to get some required tests done for a diagnosis, they first need to wait at the pathology labs for their turn for the test, then they have to wait to take their test reports from the labs and again make an appointment at the doctor's for the diagnosis. The time taken for this process usually takes days as he has to wait for appointment at every step and the process entirely resides on the patient (or his family member) being the mediator of reports from one end to the other. Also, the absence of having a common platform for the patients and doctors to maintain the medical history and prescriptions of each patient is a cause of inconvenience. In this document we intend to tackle these problems, and propose a seamless solution.

## 1.1. Purpose:

The purpose of this document is to detail the requirements of "Patient Report Application", a seamless framework that allows users to have an organised system to maintain the medical history of the patient. This platform thus also serves as a system where patient can directly get his reports from labs and comments on his reports from his doctor without making several visits and appointments. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications.

## 1.2. Scope:

MedLife is intended to be a cloud based and scalable free-to-use application that will be -
1. An easy to use framework, that allows the patient to maintain his entire medical history
2. A common platform between Labs, Patients and Doctors to facilitate the exchange of test reports.

3. A messaging system between Doctors and Patients so that patient can use this portal ask relevant doubts to his doctor.
Main aim is to coordinate the interaction between Pathology labs, doctors and patients. There is a host of features planned to streamline and improve user experience, including messaging, real-time notifications, and segregated profiles.

### 1.3. Overview:

The remainder of the document is organised into two sections - the Overall Description and the Requirements Specification. The first one provides an overview of the system functionality, system interactions and mentions the system constraints and assumptions. The second section describes the requirements specification in detailed terms and the features as stimulus-response pairs for each of these.

### 1.4. Definitions, Acronyms and Abbreviations:

| Term | Definition |
|---|---|
| User | Someone who interacts with the system: Patient, Lab or Doctor |
| RATIONALE | Rationale behind the requirement |
| GIST | A summary of the non-functional requirements |
| SCALE | Scale at which the non-functional requirement is measured |
| METER | Method of measuring the non-functional requirement |
| MUST | Minimum level of acceptable performance |
| PLAN | The level at which good success can be claimed |
| WISH | A desirable level of achievement that may not be attainable through available means |

### 1.5. References:

IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, 1998

# 2. Overall Description:

## 2.1. <u>Product Perspective:</u>

The proposed system aims to be self contained and standalone. Since a lot of personal data need to be stored securely, it will also be Closed source. User will get a website address where they may interact with their data.

### 2.1.1. System Interfaces:

This is a web based system and hence will require the operating environment for a client and server GUI.

### 2.1.2. User Interfaces:

The front-end for the web app will be developed in HTML, CSS, Javascript(and some of it's frameworks). The backend will be developed using PHP.

### 2.1.3. Hardware Interfaces:

Any operating system such as Windows, MacOs or Linux based which is capable of running modern web browsers such as Chrome, Firefox, Opera, etc. Also the device should have working Internet connection.

### 2.1.4. Software Interfaces:

The user's browser should be HTML5 compatible for a satisfactory user experience. It should also allow running Javascript. All data will be hosted online on cloud preferably Microsoft Azure.

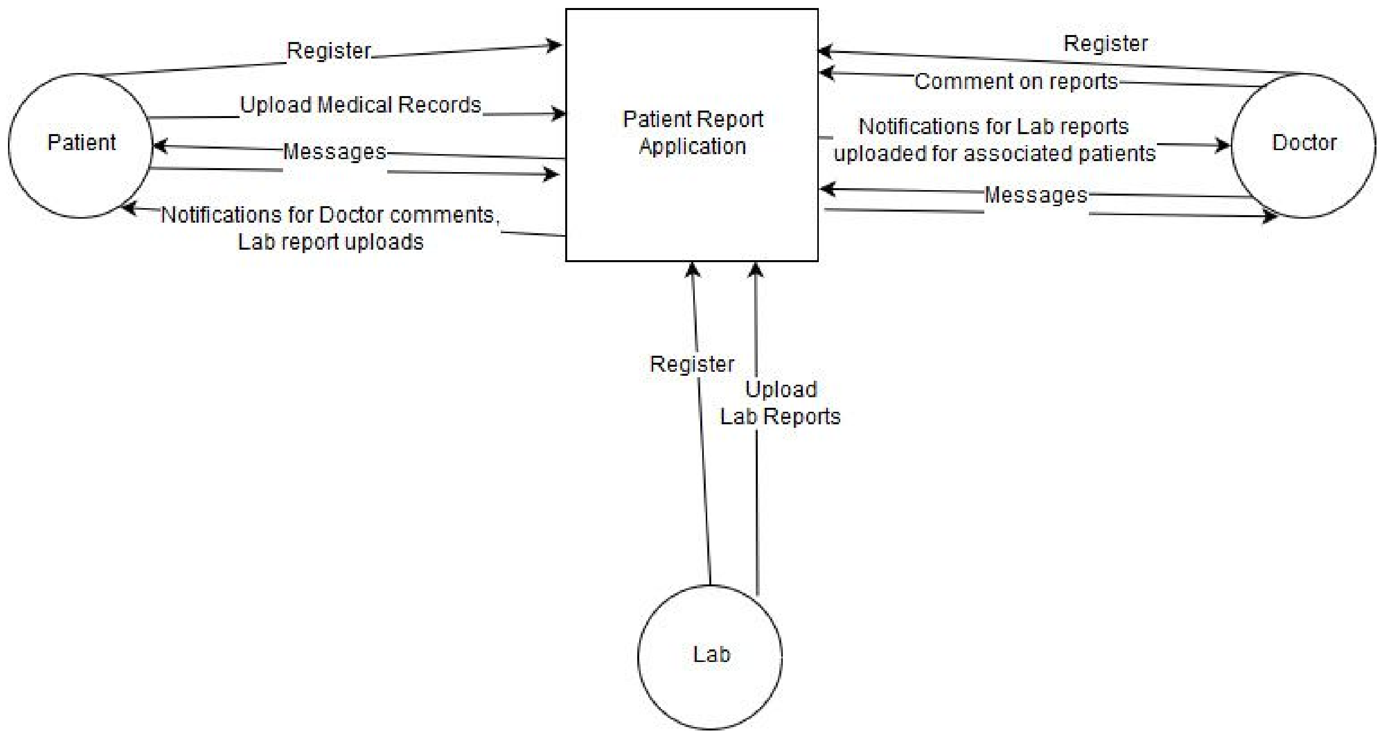
### 2.1.5. Communication:

Since we are allowing users to access their data through website only so the server-client communication will be purely over the internet, subject to protocols like TCP/IP, local network protocols, etc.

### 2.1.6. Memory:
All data will be stored in Microsoft Azure Cloud.  User data will be stored using Azure Storage Table service while images will be in Azure blob storage. The backend will be written in PHP and use the Azure Storage Table PHP Client Library. The website will serve data from this database, and this will be modified.

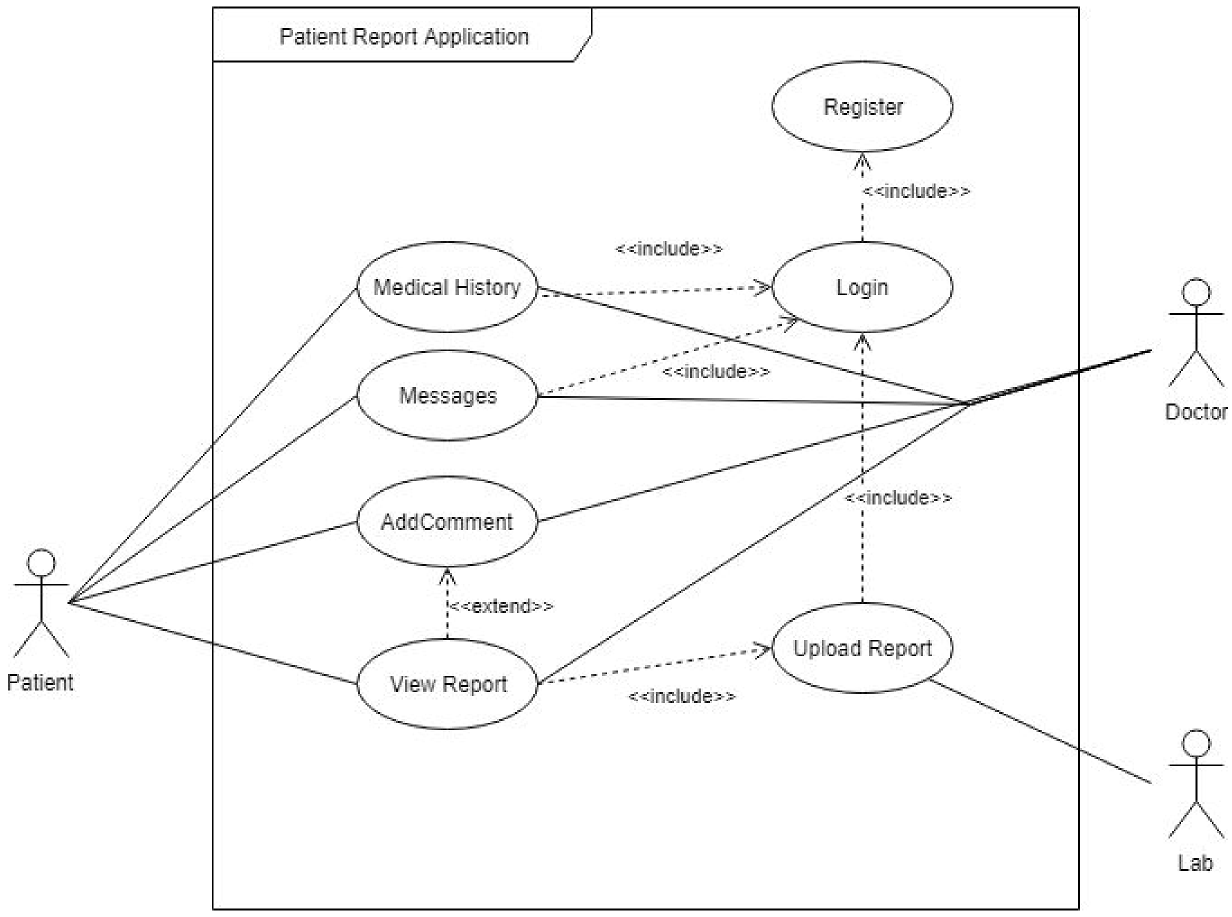
### 2.1.7. Operations:
The web application will run in three modes - patient, doctor, pathology lab. Patient will have option to provide access to his data to  doctor and pathology lab.

## 2.2. <u>Product Functions:</u>
### 2.2.1. Context diagram:



Register
Upload Medical Records
Patient
Messages
Notifications for Doctor comments, Lab report uploads
Patient Report Application
Register
Comment on reports
Notifications for Lab reports uploaded for associated patients
Doctor
Messages
Register
Upload Lab Reports
Lab

### 2.2.2. Use case diagram:



### 2.2.3. Use case description/ introduction:

1. **User registration on the website:** The user can register as Patient, Lab or Doctor. The user must provide username, password and the role he wish to register as.
2. **User log in:** The user can login as Patient, Lab or Doctor, if he has already registered. The user must provide the correct combination of username, password and his role to successfully log in into the system.
3. **Patient medical history:** After a user logs in himself as a patient, he will get to create/edit his profile by uploading his picture and relevant details as follows:
    - Age
    - Sex
    - Blood Type
    - Allergies (if any)
    - Habits (Smoking/ drinking/ drugs) (if any)
    - Family History of illness (if any)

    - Past major diseases
    - Past surgeries
4. **Adding Doctor and Lab:** The patient can add Doctor and Lab to his profile while searching with their username. Adding a doctor will allow the doctor to view the patient's profile and adding a lab will allow it to upload test reports and images that will be displayed on the patient's profile. The lab can give tags for the images which can be edited by the patient.
5. **Search reports with tags:** He can search for the reports with a given tag.
6. **Edit tags:** The patient can search and edit the tags to reports that have been uploaded by the labs.
7. **Messaging the doctor:** The patient can send a personal message to the doctor if he has any specific doubts regarding his prescription, diagnosis or appointment.
8. **Patient receives notification:**The patient receives a notification when either the lab uploads his reports or the doctor gives his comments.
9. **Doctor details:** After a user logs in himself as a doctor, he will get to create/edit his profile by uploading his picture and relevant details:
    - Age
    - Sex
    - Specialisation
    - Addresses of Clinics or Hospitals he works at
    - Working days and hours
    - Office contact numbers (if any)
10. **Searching and viewing patient's profile:** The doctor can search for the patient among all the patients that have given him access to view their profiles. He can then visit the required patient's profile and view the timeline of his reports. He can post his comment regarding the reports.
11. **Messaging the patient:** The doctor can reply to patient's personal messages.
12. **Lab details:** After a user logs in himself as a Lab, he will get to create/edit his profile by filling the relevant details:
    - Address
    - Working days and hours
    - Lab contact numbers (if any)

13. **Lab uploads reports for the patient:** Lab can search for the patient among all the patients who have given access to the lab and upload reports for the required patient with relevant tags.

## 2.3. <u>User Characteristics:</u>

There will be three kinds of users interacting with the system - patient, doctor and pathology lab.

The Patient will be anyone who wants to store his medical history and share it with his doctor conveniently.
Doctor will have their own profile, where they can view all his patients' profiles who have given him access to do so.
Pathology lab will have profile from which it can upload images of test result on a patient's profile if it has his id and access to his profile. The lab will also put various tags with the image to help find it even later.

## 2.4. <u>Constraints:</u>

- The website will require internet connection to access and download the images.
- User privacy requirements - The user data needs to be guarded safely and ought to be provided to legitimate doctors.
- Platform limitations - The website might not be suitable for mobile devices.

## 2.5. <u>Assumptions and Dependencies:</u>

- Users will have a modern browser, with Javascript enabled, for web app
- Users will have stable access to the internet when using the app. While the data will be persistent, without stable access, it would be impossible to view
- Data will be secure, on-cloud and will be stored reliably. Data backup and persistence will be handled by the database

# 3. Specific Requirements:

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

## 3.1. <u>External interface requirements:</u>

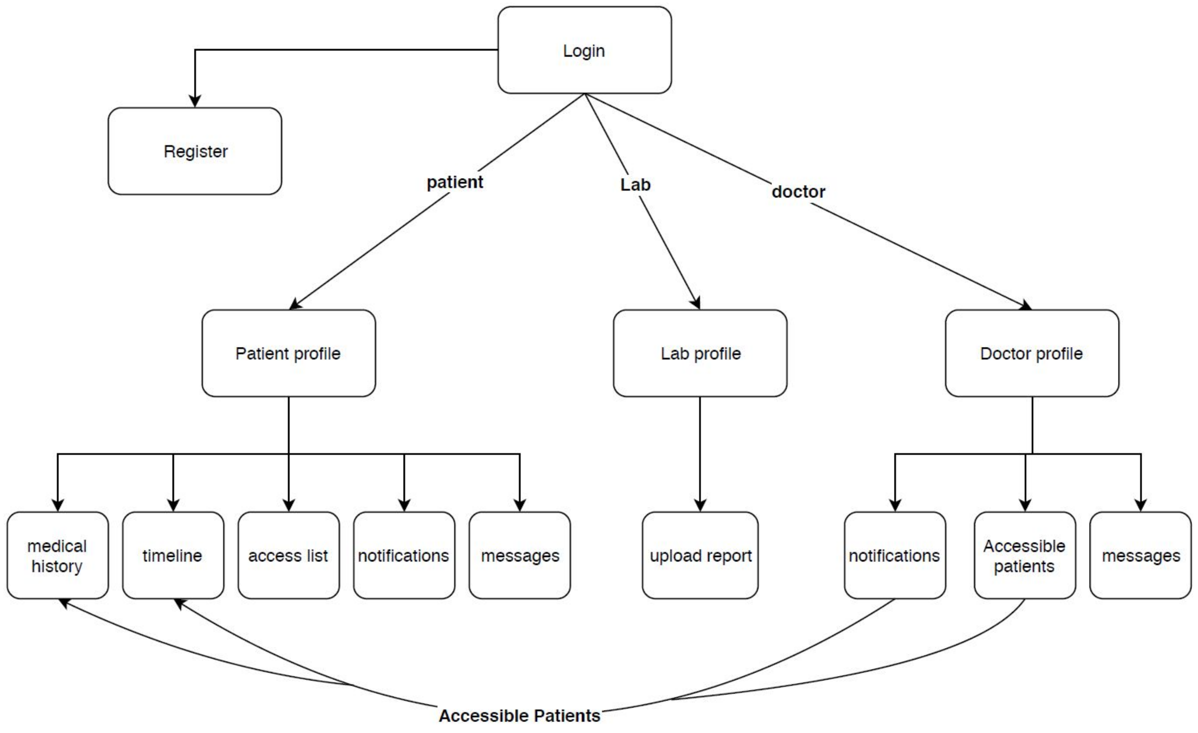
This section contains  a detailed description of all inputs and outputs to and  from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

### 3.1.1. User interfaces:

As of now the current interfaces consists of multiple screens given below:

- **Register**: User can register as patient/doctor/lab. Username, password, flag to check to register as lab, doctor or patient.
- **Login**: This will be a common interface for patient,doctors and labs. After login system  will decide where the user will land based on its role.
- **Patient profile**: Contains details of the patient who has logged-in. From here patient can navigate to  medical history tab, timeline tab, “Patient’s access list”, messages or notifications.
    - **Patient medical history**: This page contains the details filled by the patient about his history of illness, Family history of illness, habits, etc. This can be viewed by the doctors if  he has been given access by the patient.
    - **Patient timeline**: This contains patient’s records of all previous lab results and doctor’s comments (if any) sorted in chronological order. Timeline is also visible by the doctors patient has given access to.

    - **Patient access list:** Patient can manage the lists of doctors that he has given access to in this page.
    - **Messages:** Patient can exchange messages with his doctors.
    - **Notifications:** When lab uploads a report or when doctor comments on the report, patient receives the notification accordingly.
- **Doctor profile**: Contains details of the doctor who has logged-in. From here doctor can navigate to “Accessible patients", messages and notifications. Doctor can navigate to an accessible patient’s timeline and can add comment/prescription to a particular record.
    - **Accessible patients:** List of patients the doctor has access to their timelines.
    - **Messages:** Doctor’s can reply to the messages sent by patients.
    - **Notifications:** When lab uploads a report by tagging a doctor, then that doctor is notified about the upload.
- **Lab profile**: Contains details of the lab which has logged-in. From here lab can navigate to “upload report”.
    - **Upload report**: This page allows labs to upload report to a particular patient using patient id and tags the doctor using doctor id. A notification is sent to both patient and doctor after the upload.



User Interface flow

### 3.1.2. Hardware interfaces:

This is a web app, the app is displayed on a browser, which is to handle the rendering on the screen. Since the web app doesn't have any designated hardware, it does not have any direct hardware interfaces. The hardware connection to the database server is managed by the underlying operating system on the PC.

### 3.1.3. Software interfaces:

Since we are focusing on using Microsoft Azure as the database system. We are using Microsoft Azure Cloud Services PHP APIs to query/modify the database in need.

### 3.1.4. Communications interfaces:

The communication between the different parts of the system is important since they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying internet connection protocol or the browser.

## 3.2. Functional requirements:

This section includes the requirements that specify all the fundamental actions of the software system. We organise the functional requirements into 4 sub groups:

- User registration and logistics
- Patient's Profile
- Doctor's Profile
- Lab's Profile

### 3.2.1. User Registration and Logistics:

#### 3.2.1.1. Use Case 1: User Registration

a) **Brief Description-** This use case allows the user to register himself on the portal.
b) **Actors-** Patient, Doctor and Lab
c) **Pre-conditions-** None
d) **Basic Flow-**
    i) Starts when a new user opens a browser and accesses the website.
    ii) User need to enter the following details to register himself-
        1) Unique username

2) Password
3) Role (Patient/ Doctor/ Lab)

iii) The system checks the information and registers the user

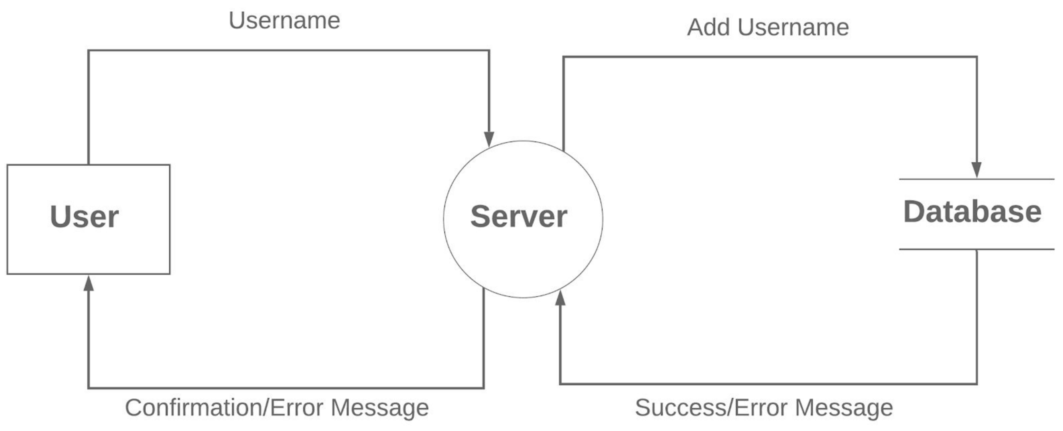
e) **Alternative Flow-**
   i) Username is not unique- If the username exists in the system, the user would have to come up with a different username which should be unique.

f) **Post-conditions-**
   If use case is successful, user account is created and the database is updated with the user information. Otherwise system remains unchanged.

g) **Special requirements-** None

h) **Use case relationships-** None

**DFD**



**3.2.1.2. Use Case 2: User Login**

a) **Brief Description-** This use case allows the user to login as Patient, Doctor or Lab.

b) **Actors-** Patient, Doctor and Lab

c) **Pre-conditions-** Use Case 1

d) **Basic Flow-**
   i) Starts when a user tries to login on the website.
   ii) The user provides username, password and his role for log in.
   iii) The System checks user information and validates the User and then subsequently displays the homepage on successful verification.

e) **Alternative Flow-**

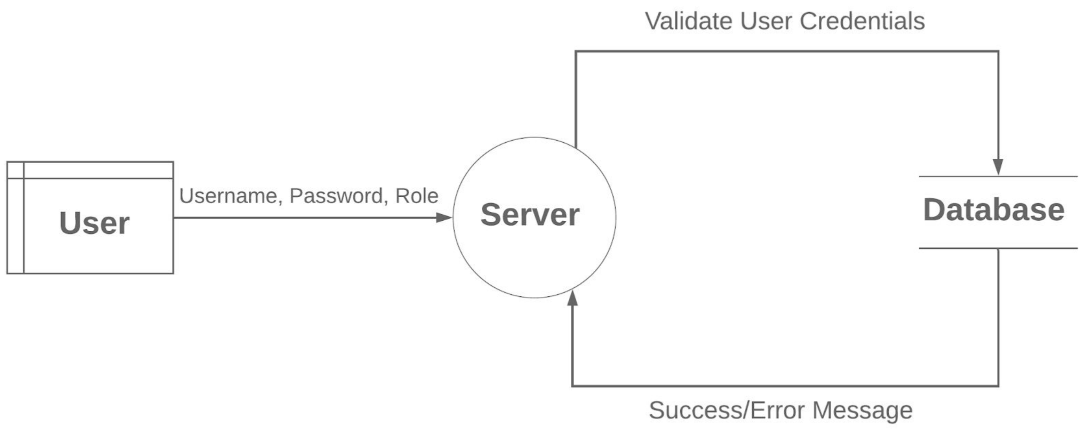
i) User does not exist- The user may enter different id and password and check his role before trying to login again.

**f) Post-conditions-**
If use case is successful, user is allowed to access his profile. Otherwise system remains unchanged.

**g) Special requirements-** None

**h) Use case relationships-** None

**DFD**



### 3.2.2. Patient Profile:

#### 3.2.2.1. Use Case 1: Patient Medical History

**a) Brief Description-** After a user logs in himself as a patient, he will get to create/edit his profile by uploading his picture and relevant details.

**b) Actors-** Patient

**c) Pre-conditions-** Patient must be logged in

**d) Basic Flow-**

i) Patient should enter his name and upload profile picture.

ii) The patient is required to enter the following details:

- Age
- Sex
- Blood Type

- Allergies (if any)
- Habits (Smoking/ drinking/ drugs) (if any)
- Family History of illness (if any)
- Past major diseases
- Past surgeries

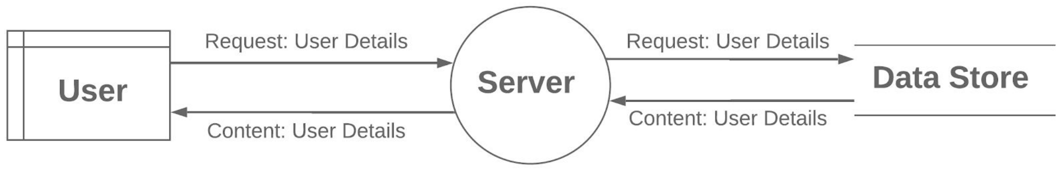
e) **Alternative Flow-** None

f) **Post-conditions-**
If use case is successful, patient's medical history is updated in the database. Otherwise system remains unchanged.

g) **Special requirements-** None

h) **Use case relationships-** None

**DFD**



**3.2.2.2. Use Case 2: Adding doctor and lab**

a) **Brief Description-** Adding doctor and lab to give them access to his profile.

b) **Actors-** Patient

c) **Pre-conditions-** Patient must be logged in

d) **Basic Flow-**
   i) The patient searches for Doctor and/or Lab by their username.
   ii) He adds a doctor to his profile to allow the doctor to view the patient's profile.
   iii) He adds a lab to allow it to upload test reports and images that will be displayed on the patient's profile.

e) **Alternative Flow-**
   i) Doctor does not exist- The patient may verify the username for the doctor and enter again.

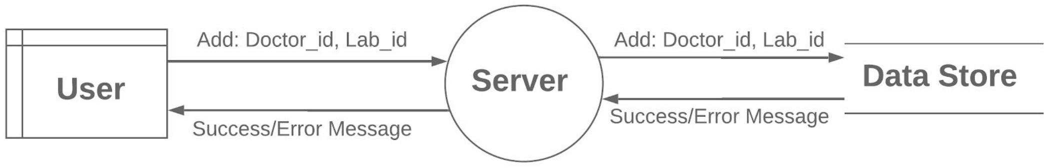
ii) Lab does not exist- The patient may verify the username for the Lab and enter again.

**f) Post-conditions-**
If use case is successful, the doctor and lab will be added to patient's list of people who can access his profile and the patient will be added to the access list of the doctor and lab. Otherwise system remains unchanged.

**g) Special requirements-** None

**h) Use case relationships-** None

**DFD**



### 3.2.2.3. Use Case 3: Search tags

**a) Brief Description-** The patient can search for reports related to a tag.

**b) Actors-** Patient

**c) Pre-conditions-** Patient must be logged in and there must be atleast one report uploaded to his profile.

**d) Basic Flow-**

i) The patient searches for reports by a given tag.

**e) Alternative Flow-**

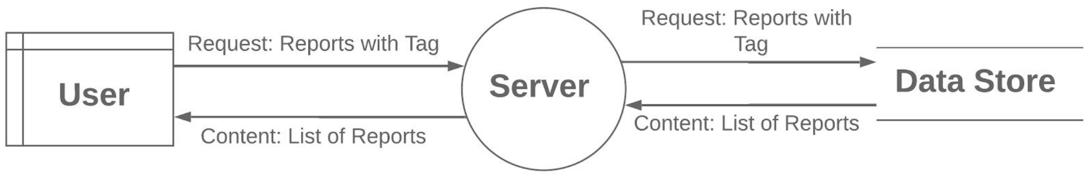
i) Tag does not exist- The patient may verify the tag and enter again.

**f) Post-conditions-** None

**g) Special requirements-** None

**h) Use case relationships-** None

**DFD**



### 3.2.2.4. Use Case 4: Edit tags

a) **Brief Description-** The patient can edit the tags to reports that have been uploaded by the labs.

b) **Actors-** Patient

c) **Pre-conditions-** Patient must be logged in and there must be atleast one report uploaded to his profile.

d) **Basic Flow-**

   i) The patient searches for reports by a given tag.

   ii) He can add or delete tags from the report.

e) **Alternative Flow-**

   i) Tag does not exist- The patient may verify the tag and enter again.

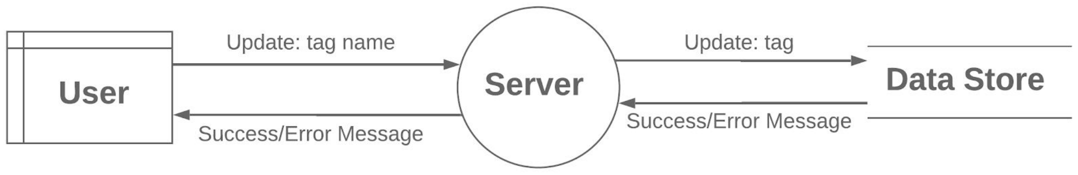
f) **Post-conditions-**

   If use case is successful, the tag corresponding to the report will be modified in the database. Otherwise system remains unchanged.

g) **Special requirements-** None

h) **Use case relationships-** None

**DFD**



### 3.2.2.5. Use Case 5: Messaging the doctor

a) **Brief Description-** The patient can send a personal message to the doctor if he has any specific doubts regarding his prescription, diagnosis or appointment.

b) **Actors-** Patient

c) **Pre-conditions-** Patient must be logged in and there must be at least one doctor added to his profile.

d) **Basic Flow-**

   i) The patient searches for doctor by his username.

   ii) The patient writes a message with his query and click send.

e) **Alternative Flow-**

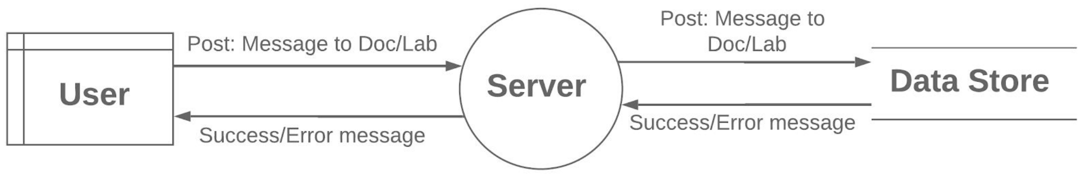
i) Doctor username does not exist- The patient may verify the doctor username and enter again.

f) **Post-conditions-**
If use case is successful, the doctor will receive the patient's message on his profile. Otherwise system remains unchanged.

g) **Special requirements-** None

h) **Use case relationships-** None

**DFD**



### 3.2.2.6. Use Case 6: Patient receives notification

a) **Brief Description-** The patient receives a notification when either the lab uploads his reports or the doctor gives his comments.

b) **Actors-** Patient

c) **Pre-conditions-** Patient must be logged in and there must be atleast one doctor or lab added to his profile.

d) **Basic Flow-**
   i) The patient clicks on the notification button and checks if a lab has uploaded his report or a doctor has added a comment.

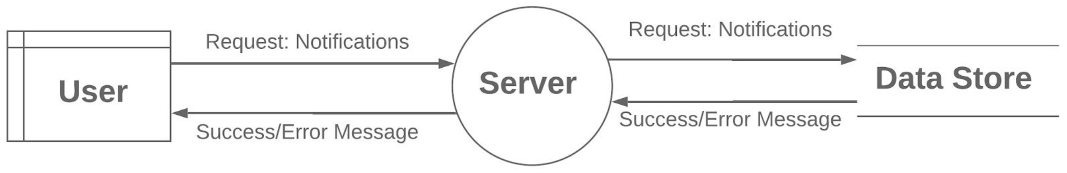
e) **Alternative Flow-** None

f) **Post-conditions-**
If use case is successful, the notification will be added to the set of patient's notifications in the database. Otherwise system remains unchanged.

g) **Special requirements-** None

h) **Use case relationships-** None

**DFD**



### 3.2.3. Doctor Profile:

#### 3.2.3.1. Use Case 1: Doctor details

a) **Brief Description-** After a user logs in himself as a doctor, he will get to create/edit his profile by uploading his picture and relevant details.

b) **Actors-** Doctor

c) **Pre-conditions-** Doctor must be logged in

d) **Basic Flow-**

  i) Doctor should enter his name and upload profile picture.

  ii) The doctor is required to enter the following details:

- Age
- Sex
- Specialisation
- Addresses of Clinics or Hospitals he works at
- Working days and hours
- Office contact numbers (if any)

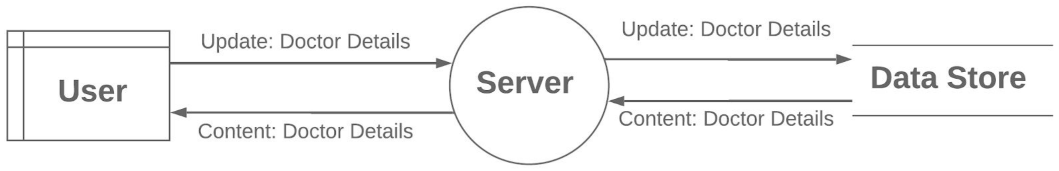
e) **Alternative Flow-** None

f) **Post-conditions-**
If use case is successful, doctor's details are updated in the database. Otherwise system remains unchanged.

g) **Special requirements-** None

h) **Use case relationships-** None

**DFD**



#### 3.2.3.2. Use Case 2: Searching and viewing patient's profile

a) **Brief Description-** Searching and viewing patient's profile

b) **Actors-** Doctor

c) **Pre-conditions-** Doctor must be logged in

**d) Basic Flow-**

   i) The doctor can search for the patient among all the patients that have given him access to view their profiles.
   ii) He can then visit the required patient's profile and view the timeline of his reports.
   iii) He can post his comment regarding the reports.

**e) Alternative Flow-**

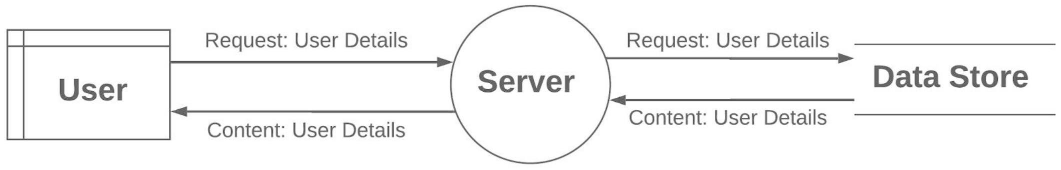
   i) Patient does not exist- The doctor may verify that the patient he is searching for has given him access to his profile.

**f) Post-conditions-**
If the doctor adds some comment then the comment is added to patients reports history for the particular report. Otherwise system remains unchanged.

**g) Special requirements-** None

**h) Use case relationships-** None

**DFD**



### 3.2.3.3. Use Case 3: Messaging the patient

**a) Brief Description-** The doctor can reply to patient's personal messages.

**b) Actors-** Doctor

**c) Pre-conditions-** Doctor must be logged in

**d) Basic Flow-**

   i) The doctor searches for patient by his username.
   ii) The doctor writes a message with his reply and clicks send.

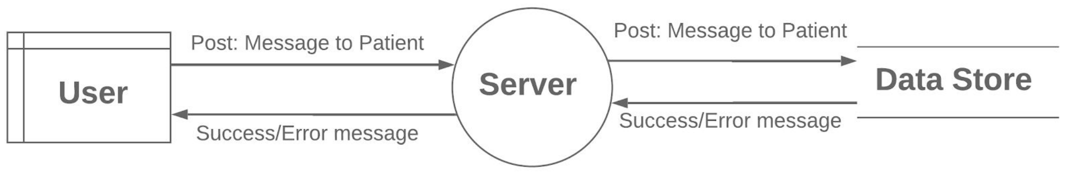
**e) Alternative Flow-**

   i) Patient username does not exist- The doctor may verify the patient username and enter again.

**f) Post-conditions-**
If use case is successful, the patient will receive the doctor's message on his profile. Otherwise system remains unchanged.

g) **Special requirements-** None
h) **Use case relationships-** None

**DFD**



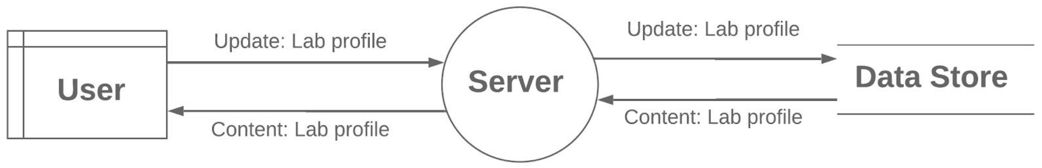
### 3.2.4. Lab Profile:

#### 3.2.4.1. Use Case 1: Lab details

a) **Brief Description-** After a user logs in himself as a Lab, he will get to create/edit his profile by filling the relevant details.
b) **Actors-** Lab
c) **Pre-conditions-** Lab must be logged in
d) **Basic Flow-**
   i) Lab should enter its name.
   ii) The lab is required to enter the following details:
      - Address
      - Working days and hours
      - Lab contact numbers (if any)

e) **Alternative Flow-** None
f) **Post-conditions-**
   If use case is successful, doctor's details are updated in the database. Otherwise system remains unchanged.
g) **Special requirements-** None
h) **Use case relationships-** None

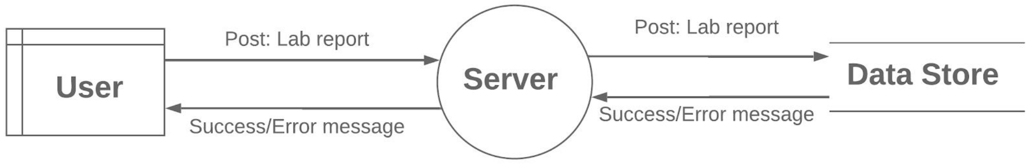
**DFD**



**3.2.4.2. Use Case 2: Upload reports**

a) **Brief Description-** Lab uploads reports for the patient
b) **Actors-** Lab
c) **Pre-conditions-** Lab must be logged in
d) **Basic Flow-**
    i) The lab can search for the patient among all the patients that have given it access to upload their reports.
    ii) Lab can choose relevant reports and click upload.
    iii) Lab can list the tags to add with the report.
e) **Alternative Flow-**
    i) Patient does not exist- The lab may verify that the patient it is searching for has given it access to his profile.
f) **Post-conditions-**
   If the use case is successful then the report is  added to patients reports history visible on his profilet. Otherwise system remains unchanged.
g) **Special requirements-** None
h) **Use case relationships-** None

DFD



## 3.3. <u>Performance requirements:</u>

### 3.3.1. Prominent search feature:

TITLE: Prominent search feature
DESCRIPTION: The search feature should be prominent and easy to find for the user.
RATIONALE: In order for user to find the search feature easily and ultimately improve user experience.

#### 3.3.2. Usage of search feature:

TITLE: Usage of search feature
DESCRIPTION: The different search options should be evident, simple and easy to understand.
RATIONALE: In order to for a user to perform a search easily

#### 3.3.3. Usage of patient timeline:

TITLE: Usage of patient timeline
DESCRIPTION: The patient's timeline should be friendly and easy to understand. Features like searching, editing tags, etc. should be at a click distance.
RATIONALE: In order to for a patient to use timeline easily.

#### 3.3.4. Response Times:

**ID: Performance requirement 3.3.4.1**
TITLE: Patient timeline load time
DESCRIPTION: The maximum time for the loading of patient timeline will be 5 seconds.
RATIONALE: Patient timeline will be one of the most used pages in the website, so to improve the overall user experience we should optimize its loading time.

**ID: Functional requirement 3.3.4.2**
TITLE: Login/Register response time
DESCRIPTION: User should be able to login/register within 1.5 seconds after sending request to server.
RATIONALE: To improve the overall user experience.

### 3.4. <u>Logical database requirements:</u>

All data will be saved in the Azure database which includes user accounts and profiles, lab report data, messages etc. Azure allows AP from CAP theorem. AP comprises of accessibility and partition tolerant so concurrent access and scalability won't be an issue later. Although eventual consistency will make sure data is coherent overall.

### 3.5. <u>Design constraints:</u>

#### 3.5.1. Programming language:
TAG: Programming Language
GIST: Programming Language
PLAN: Use HTML/CSS/Javascript for frontend and PHP for backend coding.

#### 3.5.2. Memory usage:
TAG: MemoryUsage
GIST: The amount of Operating System memory occupied by the application
SCALE: MB
METER: Observations done from the performance log during testing
MUST: No more than 20 MB
PLAN: No more than 16 MB
WISH: No more than 10 MB

## 3.6. <u>Software System Attributes:</u>
The software has the following components:

1. Azure web server
2. PHP application
3. Azure blob storage and sql database

#### 3.6.1. Reliability:
The system shall never crash or hang, other than as the result of an operating system error. The system shall also scale properly and should be able to deliver large image files to user without making user wait for too long.

#### 3.6.2. Availability:
The system should be available at all times and over the globe, except the downtime of the server. This means an user can access it using a web browser anytime and anywhere. In the instance of hardware or database failure, a proper page with relevant information should be displayed. Also in this case, backup db must be deployed within reasonable time by the server administrator.

### 3.6.3. Security:

We are dealing with very sensitive and personal data of users, so security should be one of the paramount concern here. The security aspects will be handled by Azure Cloud platform. Although there will be tradeoffs between security and usability but we will maintain the following security measure all the time:

1. Password of users will be stored in hashed form only. The hashing will be done using standard algorithm.
2. The user will have full control over his account and data. He may allow and disallow doctors to access his data.
3. We will also keep IP log of access of user data.

### 3.6.4. Maintainability:

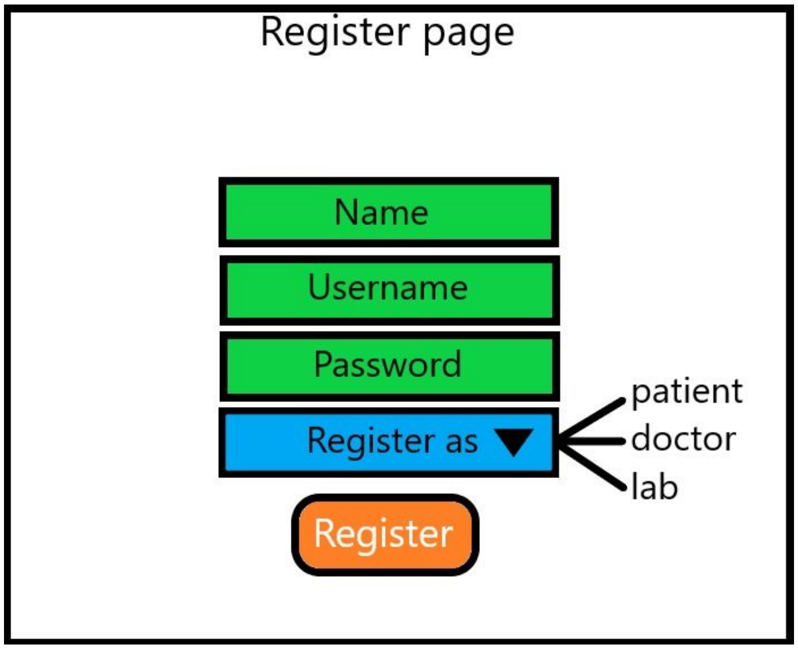
All code will be fully documented. All program files will include comments concerning authorship and date of last change. The code will be modular to permit future modifications. By using interfaces and inheritance, code reuse will be one of the priorities during software development.

### 3.6.5. Portability:

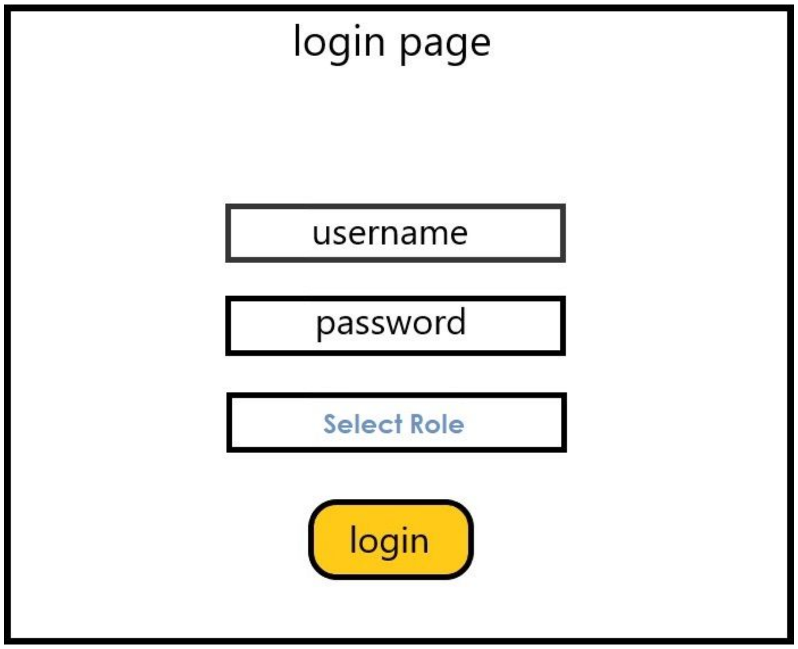
The software will be designed to run on all computer systems which have support of popular web browsers. Although we are sacrificing the backend portability by sticking to azure web services but we are getting benefit of concurrency and scalability of the Azure servers.

## 4. Screen Layouts:

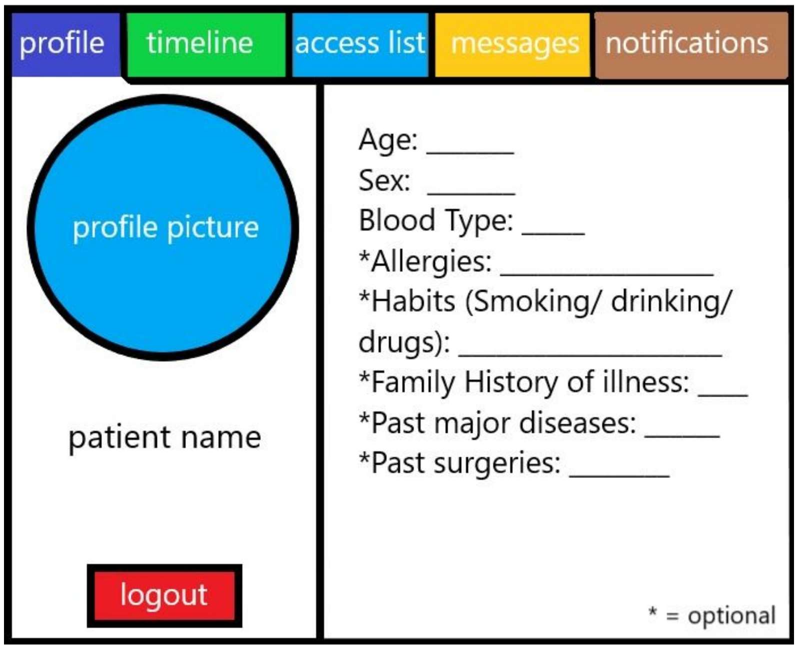
Register



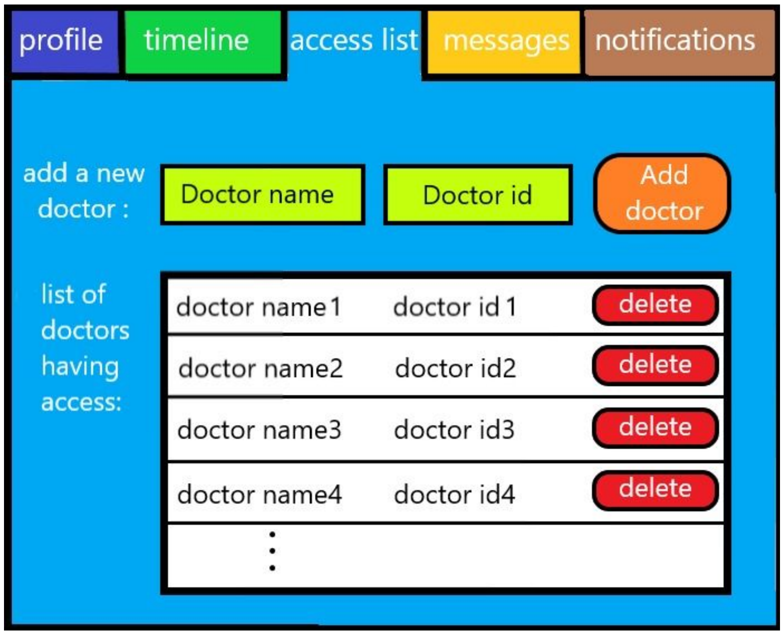
Login



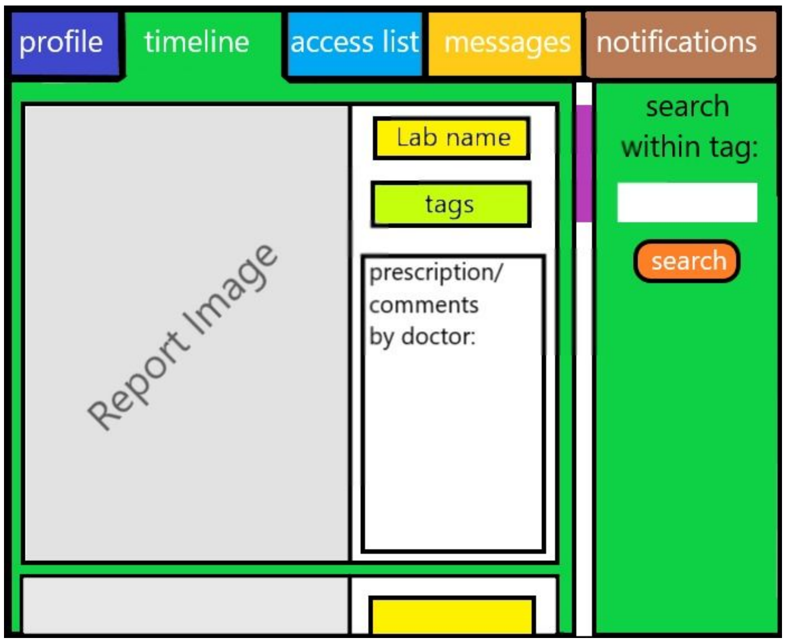
**Patient Profile**



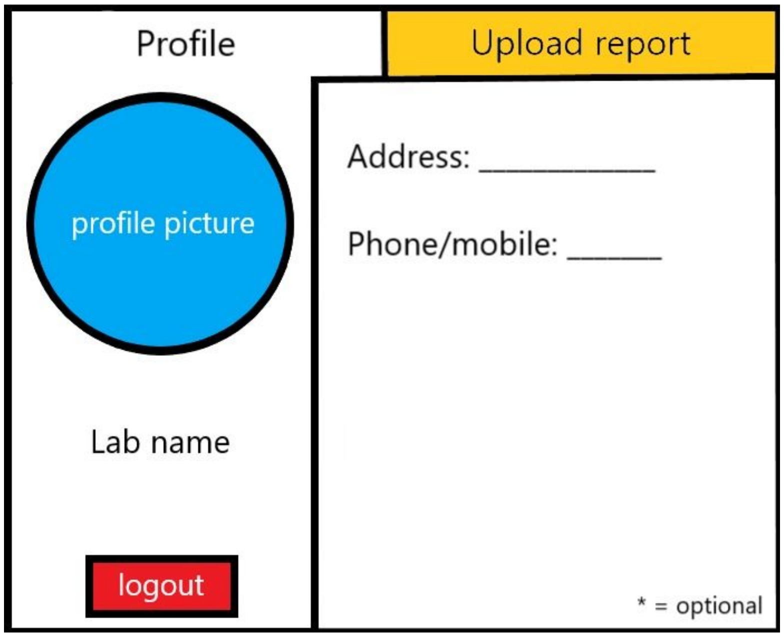
**Patient accessibility list**



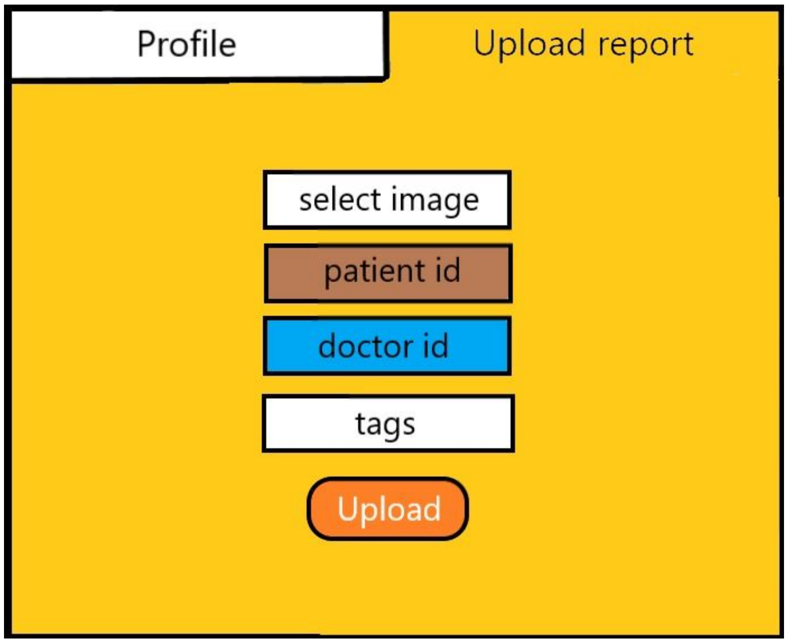
**Patient timeline**



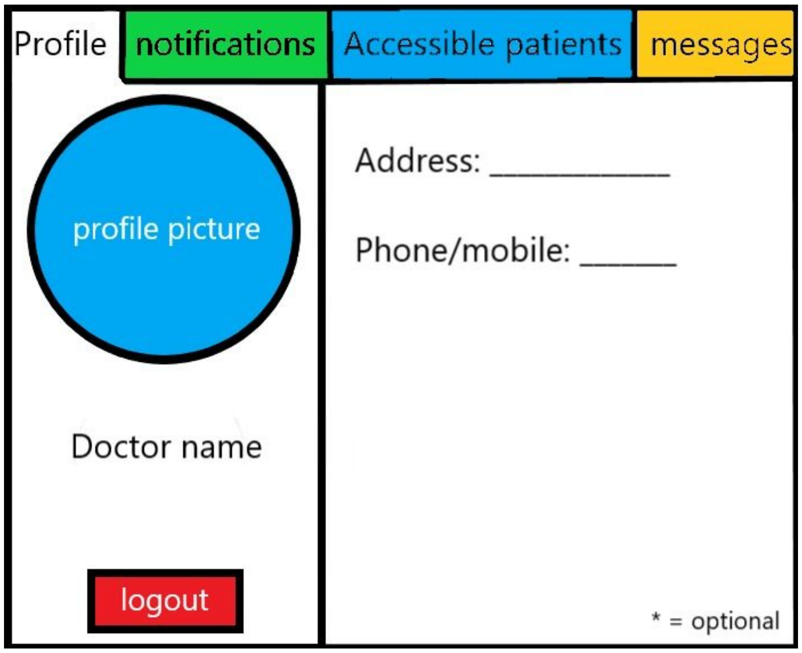
**Lab Profile**



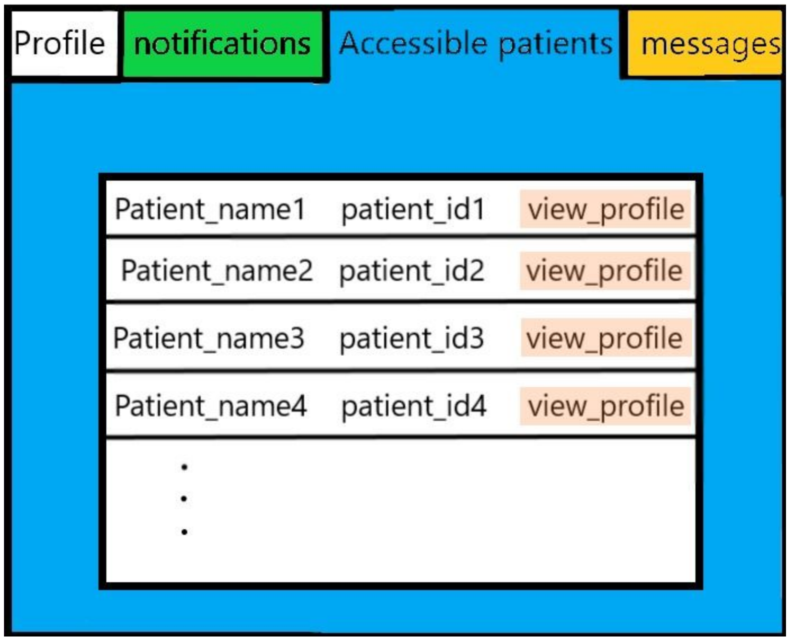
**Lab Upload:**



**Doctor Profile**



**Doctor accessibility list**


Profile
notifications
Accessible patients
messages
Patient_name1 patient_id1 view_profile
Patient_name2 patient_id2 view_profile
Patient_name3 patient_id3 view_profile
Patient_name4 patient_id4 view_profile