

Patient Report Application

Software Requirement Specification

GROUP 8:

Bipul Kumar (2014CS50282)

Kartar Singh (2014CS50286)

Prachi Singh (2014CS50289)

Gulshan Jangid (2014CS50285)



Table of Contents

Table of Contents	1
1. Introduction:	3
1.1. Purpose	3
1.2. Scope	3
1.3. Overview	4
1.4. Definitions, Acronyms and Abbreviations	4
1.5. References	4
2. Overall Description:	5
2.1. Product Perspective	5
2.1.1. System Interfaces	5
2.1.2. User Interfaces	5
2.1.3. Hardware Interfaces	5
2.1.4. Software Interfaces	5
2.1.5. Communication	5
2.1.6. Memory	6
2.1.7. Operations	6
2.2. Product Functions	6
2.3. User Characteristics	6
2.4. Constraints	7
2.5. Assumptions and Dependencies	7
3. Specific Requirements:	8
3.1. External interface requirements	8
3.1.1. User interfaces	8
3.1.2. Hardware interfaces	9
3.1.3. Software interfaces	10
3.1.4. Communications interfaces	10
3.2. Functional requirements	10
3.2.1. User Registration and Logistics	10
3.2.2. Patient Profile	11
3.2.3. Doctor Profile	12
3.2.4. Lab Profile	13
3.3. Performance requirements	13
3.3.1. Prominent search feature	13

3.3.2. Usage of search feature	13
3.3.3. Usage of patient timeline	14
3.3.4. Response Times	14
3.4. Logical database requirements	14
3.5. Design constraints	14
3.5.1. Programming language	14
3.5.2. Memory usage	15
3.6. Software System Attributes	15
3.6.1. Reliability	15
3.6.2. Availability	15
3.6.3. Security	15
3.6.4. Maintainability	16
3.6.5. Portability	16
4. Screen Layouts:	17



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:

"Patient Report Application" 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
ID	Unique identification of the requirement
DESCRIPTION	Description of the requirement
RATIONALE	Rationale behind the requirement
DEPENDENCY	Dependencies, other requirements on which this requirement depends
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. Product Perspective:

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. Product Functions:

The primary functions of the system are: storing patient information and making it accessible for later use. These tasks broadly utilise/offer the following functionality:

- Storing general information about a patient
 - Patient name, address
 - Age, Sex, Blood group
 - History of illness - Diabetes, Cholesterol
 - Family history - Anemia, Down syndrome
 - Habits - such as alcohol, smoke, tobacco
 - Desire and allergies - Allergy to pollen, nuts
- Storing past history of various tests
 - Shared with doctor when the patient visits
 - Test results uploaded by lab
- Inter-User communication
 - Notification to patient: when lab uploads test result
 - Notification to doctor: when patient gives profile access to doctor
 - Messaging facility between doctor and patients

2.3. User Characteristics:

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. Constraints:

- 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. Assumptions and Dependencies:

- 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. External interface requirements:

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.

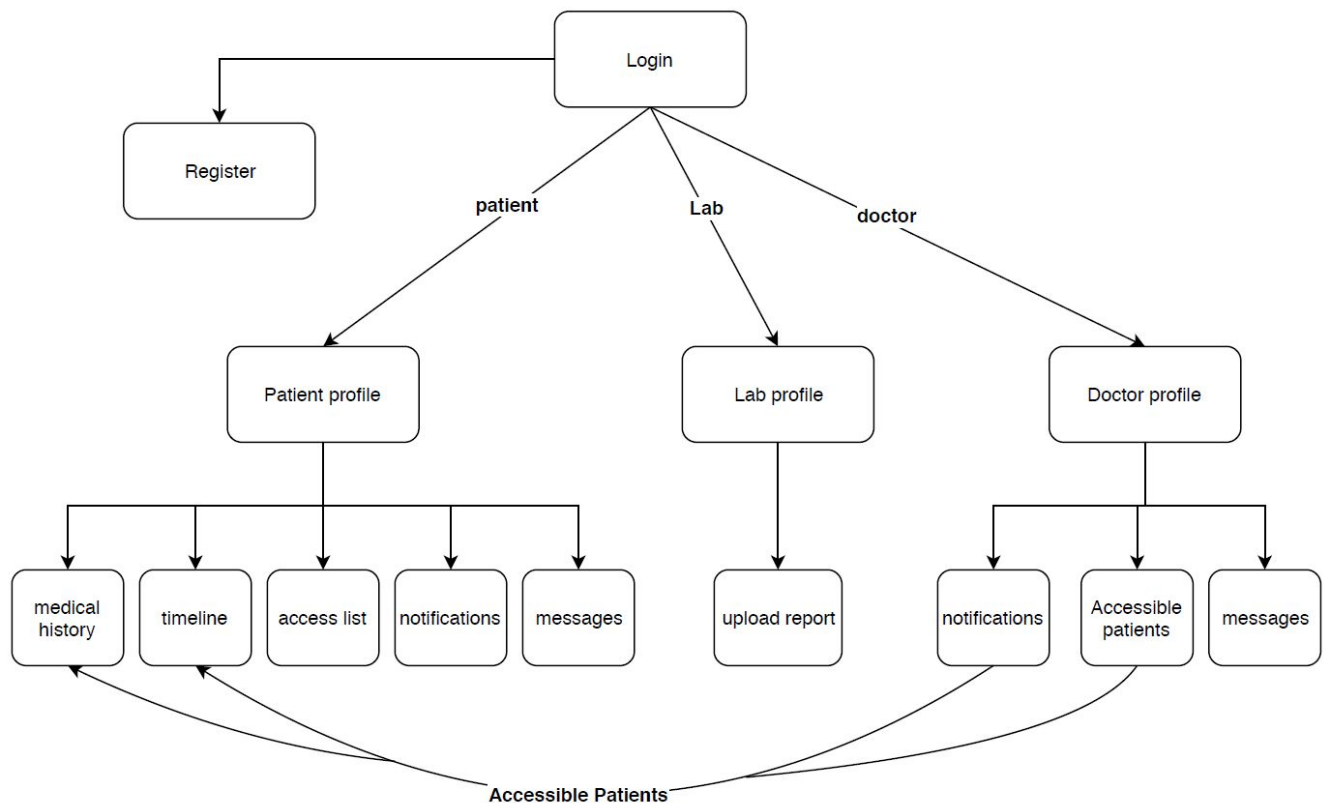
3.1.1. User interfaces:

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: These requirements is applicable to access the website, creating user profile, entering personal details, logging in.
- Patient's Profile
- Doctor's Profile
- Lab's Profile

3.2.1. User Registration and Logistics:

ID: Functional requirement 3.2.1.1

TITLE: User will get link to the website

DESCRIPTION: Every user must have the link to the website, or he might google it.

DEPENDENCY: None

ID: Functional requirement 3.2.1.2

TITLE: User registration on the website

DESCRIPTION: The user can register as Patient, Lab or Doctor. The user must provide username, password and the role he wish to register as.

DEPENDENCY: Functional requirement 3.2.1.2

ID: Functional requirement 3.2.1.3

TITLE: User log in

DESCRIPTION: 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.

DEPENDENCY: Functional requirement 3.2.1.2

3.2.2. Patient Profile:**ID: Functional requirement 3.2.2.1**

TITLE: Patient medical history

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 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

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2

ID: Functional requirement 3.2.2.2

TITLE: Adding Doctor and Lab

DESCRIPTION: 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.

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2

ID: Functional requirement 3.2.2.3

TITLE: Search or edit tags

DESCRIPTION: The patient can edit the tags to reports that have been uploaded by the labs. He can search for the reports with a given tag.

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2, 3.2.2.2

ID: Functional requirement 3.2.2.4

TITLE: Messaging the doctor

DESCRIPTION: The patient can send a personal message to the doctor if he has any specific doubts regarding his prescription, diagnosis or appointment.

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2, 3.2.2.2

ID: Functional requirement 3.2.2.5

TITLE: Patient receives notification

DESCRIPTION: The patient receives a notification when either the lab uploads his reports or the doctor gives his comments.

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2, 3.2.2.2

3.2.3. Doctor Profile:

ID: Functional requirement 3.2.3.1

TITLE: Doctor details

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:

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

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2

ID: Functional requirement 3.2.3.2

TITLE: Searching and viewing patient's profile

DESCRIPTION: 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.

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2, 3.2.2.2

ID: Functional requirement 3.2.3.3

TITLE: Messaging the patient

DESCRIPTION: The doctor can reply to patient's personal messages.

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2, 3.2.2.2, 3.2.2.4

3.2.4. Lab Profile:**ID: Functional requirement 3.2.4.1**

TITLE: Lab details

DESCRIPTION: 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)

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2

ID: Functional requirement 3.2.4.2

TITLE: Lab uploads reports for the patient

DESCRIPTION: 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.

DEPENDENCY: Functional requirements 3.2.1.1, 3.2.1.2, 3.2.2.2

3.3. Performance requirements:**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. Logical database requirements:

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. Design constraints:

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. Software System Attributes:

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.

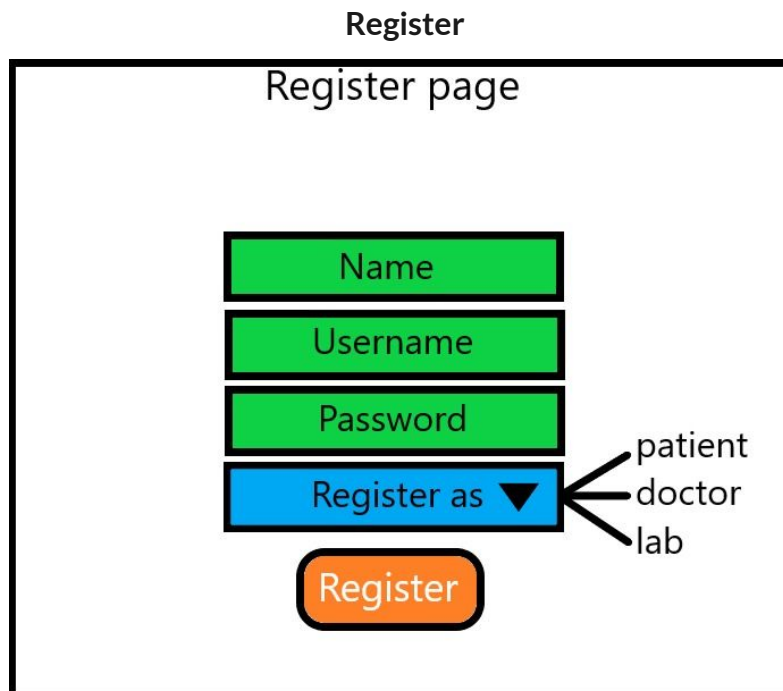
3.6.5. Portability:

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

Register page



The diagram shows a vertical stack of four input fields: 'Name', 'Username', 'Password', and 'Register as'. The 'Register as' field is a dropdown menu with a downward arrow, and three lines point to it from the right, labeled 'patient', 'doctor', and 'lab'. Below these fields is an orange rounded button labeled 'Register'.

Name

Username

Password

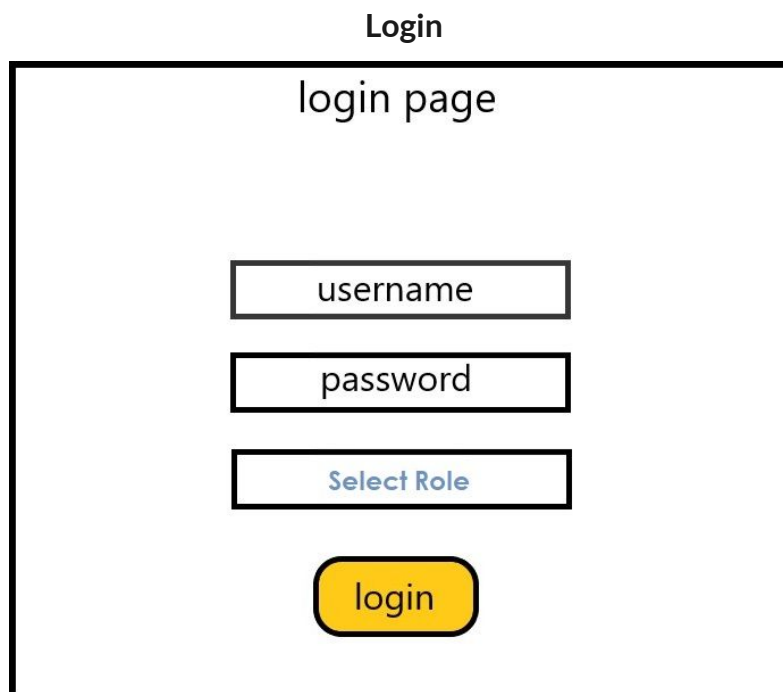
Register as ▼

patient
doctor
lab

Register

Login

login page



The diagram shows a vertical stack of three input fields: 'username', 'password', and 'Select Role'. The 'Select Role' field is a dropdown menu with a downward arrow. Below these fields is a yellow rounded button labeled 'login'.

username

password

Select Role ▼

login

Patient Profile

profile	timeline	access list	messages	notifications
<div>profile picture</div> <div>patient name</div> <div>logout</div>		<div>Age: _____</div> <div>Sex: _____</div> <div>Blood Type: _____</div> <div>*Allergies: _____</div> <div>*Habits (Smoking/ drinking/ drugs): _____</div> <div>*Family History of illness: _____</div> <div>*Past major diseases: _____</div> <div>*Past surgeries: _____</div> <div>* = optional</div>		

Patient accessibility list

profile	timeline	access list	messages	notifications															
<div>add a new doctor : <div>Doctor name</div> <div>Doctor id</div> <div>Add doctor</div></div> <div>list of doctors having access:</div> <table border="1"><tbody><tr><td>doctor name1</td><td>doctor id 1</td><td>delete</td></tr><tr><td>doctor name2</td><td>doctor id2</td><td>delete</td></tr><tr><td>doctor name3</td><td>doctor id3</td><td>delete</td></tr><tr><td>doctor name4</td><td>doctor id4</td><td>delete</td></tr><tr><td colspan="3">⋮</td></tr></tbody></table>					doctor name1	doctor id 1	delete	doctor name2	doctor id2	delete	doctor name3	doctor id3	delete	doctor name4	doctor id4	delete	⋮		
doctor name1	doctor id 1	delete																	
doctor name2	doctor id2	delete																	
doctor name3	doctor id3	delete																	
doctor name4	doctor id4	delete																	
⋮																			

Patient timeline

profile	timeline	access list	messages	notifications
<div>Report Image</div>		Lab name	<div>search within tag: <input type="text"/> <input type="button" value="search"/></div>	
		tags		
		prescription/ comments by doctor:		

Lab Profile

Profile	Upload report
<div>profile picture</div>	Address: _____
Lab name	Phone/mobile: _____
<input type="button" value="logout"/>	* = optional

Lab Upload:

Profile	Upload report
<div>select image</div> <div>patient id</div> <div>doctor id</div> <div>tags</div> <div>Upload</div>	

Doctor Profile

Profile	notifications	Accessible patients	messages
<div>profile picture</div> <div>Doctor name</div> <div>logout</div>	<div>Address: _____</div> <div>Phone/mobile: _____</div> <div>* = optional</div>		

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	
.	.	.	