

SMART HEALTH PREDICTION

TABLE OF CONTENTS

- TITLE OF THE PROJECT
- INTRODUCTION
 - Basic information of Project
 - Objective and Scope
 - Project Category
 - Tools and Technologies used
- SYSTEM ANALYSIS
 - Preliminary analysis & Information gathering
 - Input/Outputs
 - Feasibility Study
 - System Requirements Specification
 - Software Engineering Model Used
 - Project Scheduling
- SYSTEM DESIGN
 - Modules
 - System Architecture
 - Data Flow Diagram
 - E-R Diagram
 - Use case Diagram
 - Class Diagram
 - Sequence Diagram
 - Activity Diagram
 - Data Base Design
 - User interface Design Screen Shots
- CODING & TESTING
- IMPLEMENTATION & MAINTENANCE
- REFERENCES

INTRODUCTION

Basic Introduction of Project

In emergency situations when immediate medical assistance is required, individuals often face challenges in accessing doctors promptly due to various reasons. This delay in receiving medical guidance can have severe consequences on the well-being and health outcomes of individuals. Furthermore, there is a need for a reliable and efficient system that can provide instant support and consultation to users regarding their health issues, leveraging intelligent healthcare technology. The existing healthcare system may not always be able to offer immediate assistance and guidance, leading to increased anxiety and uncertainty among patients. Therefore, there is a need for a system that utilizes a comprehensive database of symptoms and diseases to provide instant guidance and recommendations to users, helping them assess their health conditions and seek appropriate medical attention in a timely manner.

It might have happened so many times that someone needs doctors help immediately, but they are not available due to some reason. This project is an end user support and consultation project. Here we propose a system that allows users to get instant guidance on their health issues through an intelligent health care system. The system is fed with various symptoms and the disease/illness associated with those symptoms. The system allows user to share their symptoms and issues. It then processes user's symptoms to check for various illness that could be associated with it. If the system is not able to provide suitable results, it informs the user about the type of disease or disorder it feels user's symptoms are associated with. If user's symptoms do not exactly match any disease in our database, it shows the diseases user could probably have judging by his/her symptoms. It also consists of doctor address, contacts along with Feedback and administrator dashboard for system operations. Patients are allowed to book empty slots and those slots are reserved in their name. The system manages the appointment data for multiple doctors for various date and times.

Objectives & Scope

SCOPE

This application can be used by anyone who needs immediate guidance on their health issues, especially when a doctor is not available. It can also be used by patients to book appointments with doctors and manage their medical records. Additionally, doctors can use this application to manage their appointment.

OBJECTIVES

The objectives of this application are:

1. To provide an intelligent healthcare system that allows users to get instant guidance on their health issues.
2. To process user symptoms and check for various illnesses that could be associated with them.
3. To inform users about the type of disease or disorder that the system feels the user's symptoms are associated with.
4. To show the diseases the user could probably have judging by their symptoms if they do not match any disease in the database.
5. To provide doctors' addresses, contacts, and appointment booking facilities for users.

Project Category

Mobile APPLICATION – It is a Mob App developed for Android Mobiles using Java, XML Language and SQLite database as the data source.

Tools and Technologies Used

Java Programming

Developers can write code once and utilize it multiple times, thanks to the valuable feature of code reusability provided by Java, a versatile programming language. Significant time and effort are saved by this aspect, enabling existing code created by others to be leveraged by developers for tasks that may exceed their own expertise or development capabilities. The need to reinvent the wheel is eliminated by the ability to reuse code and developers are empowered to efficiently accomplish complex functionalities.

The emergence of the Android platform predates the rich history of the Java programming language. The object-oriented programming paradigm, which emphasizes the organization and manipulation of data through objects and classes, has been widely used and recognized. Modular and maintainable code is promoted by this approach, making it easier for software projects to be designed, implemented, and managed by developers.

The need for a platform-independent language that could be employed to write code for various consumer electronic devices was one of the primary motivations behind the creation of Java. Developers were allowed to write code once and deploy it on different hardware and operating system configurations because the language was designed to be architecture-neutral. Software development was revolutionized by this platform independence, as it enabled developers to target multiple platforms without the need for extensive modifications or rewriting of code.

Its widespread adoption and popularity have been greatly contributed to by the versatility and platform independence offered by Java. An ideal choice for a wide range of applications, from enterprise systems to web development and, of course, Android app development, is made due to its robustness, scalability, and extensive library support. Moreover, learning, collaboration, and code sharing among developers are further facilitated by the vast Java community and extensive documentation available.

XML

Documents are specified by a markup language. One such markup language commonly used for creating structured information documents is XML (Extensible Markup Language). While data and its structure are defined by XML, data on the web is primarily displayed by HTML (Hypertext Markup Language). Distinct purposes are served by XML and HTML, with XML being designed to complement HTML rather than replace it. Authors are provided with the flexibility to design and organize their own document markups by XML, allowing for customized structures and data representation. On the other hand, the organizational structure of HTML documents is defined by HTML and predefined tags are used to mark up web pages. Predefined tag sets or inherent interpretations are not had by XML, unlike HTML. Writers are empowered by this characteristic to create their own tags and establish structural relationships as per their specific requirements. The development of customized and domain-specific document structures is allowed by XML, facilitating interoperability and data exchange between different systems and applications.

Android Operating System

A mobile operating system called Android, which is based on the Linux kernel, has been specifically designed for devices like smartphones and tablets. Several businesses, under the guidance of the Open Handset Alliance, developed it collaboratively, with a prominent role played by Google in its creation. It is found crucial by programmers to design applications for the Android platform because a standardized and consistent approach to managing mobile app development is offered by it. A diverse range of Android-powered devices can have their applications created by developers to work seamlessly, ensuring a wider reach and user base.

The beta versions of the Android Software Development Kit (SDK) were made available to developers by Google in 2007, followed by the release of the first commercial version, Android 1.0, in September of the same year. A significant milestone was marked in the evolution of Android as a fully-fledged mobile operating system. Developers can freely access the source code of Android, which is one of its notable aspects due to its open-source nature. Customization, innovation, and collaboration within the Android development community are allowed by this open approach. A vibrant ecosystem of developers has been fostered by the availability of the source code under open

The development of third-party applications and modifications is also encouraged by the open-source nature of Android, contributing to its widespread popularity and versatility. Opportunities for developers to create unique and tailored experiences for Android users have been created, leading to a diverse range of apps and functionalities available on the platform.

Android Studio

Google developed Android Studio, which is the official Integrated Development Environment (IDE) for Android app development. Developers are empowered by it to create a wide range of applications for the Android ecosystem. The development process is simplified by Android Studio, which provides a suite of powerful tools and features. A user-friendly interface is offered by it that streamlines various tasks, such as coding, debugging, and project management. Developers can write and edit code efficiently, benefiting from features like auto-completion and code refactoring with its advanced code editor. A powerful debugger is also included in the IDE, which assists in identifying and fixing issues during the development phase. Furthermore, the Android Software Development Kit (SDK) is seamlessly integrated with Android Studio, granting developers access to a vast collection of libraries, APIs, and resources. The full potential of the Android platform can be leveraged by developers through this integration, and feature-rich applications can be built. Both emulators and real hardware devices are supported for running code on by Android Studio. Developers are enabled by this flexibility to test and debug their applications in various scenarios and configurations, ensuring optimal performance and user experience across different devices. Furthermore, comprehensive documentation, tutorials, and a thriving developer community are provided by Android Studio. Developers are enabled by these resources to learn, collaborate, and seek assistance when facing challenges during the app development process.

SQLite Database

SQLite is a popular open-source, server less, self-contained, and embedded relational database management system (RDBMS). Unlike traditional database systems that require a separate server process, SQLite operates as a library that is directly integrated into the application. It is designed to be lightweight, fast, and easy to use, making it a preferred choice for various applications, especially in mobile and embedded systems. Here are some key features and characteristics of SQLite:

- **SQLite is totally free:** SQLite is open-source. So, no license is required to work with it.
- **SQLite is server less:** SQLite doesn't require a different server process or system to operate.
- **SQLite is very flexible:** It facilitates you to work on multiple databases on the same session on the same time.
- **Configuration Not Required:** SQLite doesn't require configuration. No setup or administration required.
- **SQLite is a cross-platform DBMS:** You don't need a large range of different platforms like Windows, Mac OS, Linux, and Unix. It can also be used on a lot of embedded operating systems like Symbian, and Windows CE.
- **Storing data is easy:** SQLite provides an efficient way to store data.
- **Variable length of columns:** The length of the columns is variable and is not fixed. It facilitates you to allocate only the space a field needs. For example, if you have a varchar(200) column, and you put a 10 characters' length value on it, then SQLite will allocate only 20 characters' space for that value not the whole 200 space.
- **Provide large number of API's:** SQLite provides API for a large range of programming languages. **For example:** .Net languages (Visual Basic, C#), PHP, Java, Objective C, Python and a lot of other programming language.

SYSTEM ANALYSIS

Preliminary analysis & Information gathering

A literature review is a critical and systematic examination of existing literature, research papers, articles, and other relevant sources related to a specific research topic. It aims to provide a comprehensive summary and evaluation of the existing knowledge and findings on the subject matter. In this section, the literature review provides insights into the shortcomings of the existing system, discusses the tools and technologies employed in previous research or projects, and outlines the hardware and software requirements necessary for the implementation of the proposed solution. By conducting a thorough literature review, researchers can establish a foundation of knowledge, identify research gaps, and gain a deeper understanding of the research landscape surrounding their topic.

1. Smart Health Consulting Android System

International Journal of Innovative Research in Science, Engineering and Technology

Consulting a doctor is a quite obvious thing in our day to day life, but the availability of the doctor during the time of our requirement is unpredictable. In order to overcome this issue a proposal of android application is made, this smart health application enables users to get instant report on their health issues through a intelligent health care application online. This E-health application enables user to express their symptoms and issues. It then processes user's issues and symptoms to check for various health issues that could be associated with the symptoms given by the user. If the application is unable to provide a particular solution then it urges the user to under-go tests like blood test, CITI scan accordingly.

This application allows user to get instant supervision on their health issues through an smart health care application online. The application is feed with various symptoms and the diseases associated with those systems. Patient can check their medical record Hence, this system provides Quality Health Care to everyone and error free and smooth communication to patients.

Reference

http://www.ijirset.com/upload/2017/march/89_SOLUTION.pdf

2. Smart Doctor: A Urgent Health Care System

International Journal of Engineering Research & Technology (IJERT)

“SMART DOCTOR: A URGENT HEALTH CARE SYSTEM” is the use of medical based communication and information technology to provide clinical health care from a distance. It has been used to overcome distance barriers and to improve access to medical services that would often not be consistently available in distant rural communities. It is also used to save lives in critical care and emergency situations. Although there were distant precursors to medicine, it is essentially a product of medical based communication and information technologies. Bandwidth and cost are major determinants for the success of any medical based program. In this medical based system, be developed which works on low bandwidth and is less costly to provide consultation services in rural and remote areas of India. Expected outcome: A web based consultation system which can work on low bandwidth mobile internet connection to provide online care for rural patients.

Reference

<https://www.ijert.org/research/smart-doctor-a-urgent-health-care-system-IJERTCONV6IS03025.pdf>

3. An online mobile/desktop application for supporting sustainable chronic disease self-management and lifestyle change

Health self-management has become a new trend in healthcare management due to its effectiveness in improving patient health, quality of life, and life satisfaction and simultaneously reducing the cost of care. To evaluate the potential of mobile health, we developed an online health self-management system for mobile or desktop environment to help patients self-manage their health in home settings. Certain elements (e.g. education, entertainment, and rewards) were built into the system to encourage patients to both adopt and continue using it. The system was shown to two groups of patients: an Internet-panel group of 198 patients with one or more serious chronic illnesses and 83 peripheral arterial disease patients in an in-person study group. A statistical model based on Unified Theory of Acceptance and use of Technology in a consumer context was used to analyze the results. The results from both groups confirmed that such systems, from the perspectives of patients (in a “pre-use” stage), are useful, beneficial, and rewarding to use.

Reference

<https://journals.sagepub.com/doi/10.1177/1460458220944334>

Input/Outputs

Input design and output design are two critical components of the system design phase in the software development process. They focus on defining how data is collected from users (input) and how the processed results are presented to users (output) within a software system

Input Design:

Input design involves determining how users interact with the software system to provide data or information. The goal is to create a user-friendly and efficient interface for data entry. Proper input design is essential to ensure accurate and valid data entry, minimize errors, and enhance user experience.

Output Design:

Output design focuses on presenting the processed information to users in a clear and meaningful manner. It aims to provide users with the necessary information in a format that is easily understandable and actionable. Effective output design enhances user productivity and decision-making.

Feasibility Study

Feasibility study is a procedure that identifies, describes and evaluates candidate android application and selects the best application feature for the job. An estimate is made whether the identified users need may be satisfied using the current networking facilities and hardware technologies. The study will decide whether the proposed android application will be cost effective from a business point of view and if it can be developed using the given existing budgetary constraints.

The key considerations involved in the feasibility analysis of the proposed application are the following:

- a. Operational Feasibility
- b. Technical Feasibility
- c. Economic Feasibility
- d. Schedule Feasibility

Operational Feasibility

Operational feasibility is necessary as it ensures that the project developed is a successful one.

The operational feasibility of this project is high since it is user friendly and the application provides all the expected outputs to the user.

Technical Feasibility

Technical feasibility analysis makes a comparison between the level of technology available and that is needed for the development of the project. The level of technology consists of the factors like software tools, and platform developed and so on. Since, the resources for the development of the project is available, the project is technically feasible.

Economic Feasibility

This is the most important part of the project because the terms and conditions for implementing the project have to be economically feasible. The risk of finance does not exist as the existing hardware that is smart phone is sufficient and the software is free of cost. So, it is believed that the system is economically feasible.

Schedule Feasibility

Schedule feasibility is defined as the likelihood of a project being completed within its desired timeframe. Since this project has a high likelihood of completion by the desired due date, schedule feasibility is considered to be high.

System Requirements Specification**FUNCTIONAL REQUIREMENTS**

A software system is significantly influenced by functional requirements, which are essential elements that determine its behaviour, features, and functions. Functional specifications are often referred to and provide a detailed explanation of what the software should be capable of and how it should respond to various inputs and scenarios. Specific information about the interactions between users and the system, as well as the expected outcomes and responses from the system, is offered by these requirements.

The app has 3 users:

1. Admin
2. Patient
3. Doctor

Admin

1. **Login** - With predefined username and password
2. **Add Doctor** - Admin can add a doctor with all details like name, field of specialization, contact number, mail id, and address.
3. **Add Disease** - Admin can add a disease along with its symptoms.
4. **View Doctor** - Admin can view the list of all the doctors who are added.
5. **View Patients** - Admin can view the list of all the patients who are registered.
6. **View Diseases** - Admin can view the list of all the diseases
7. **View Feedback** - Admin can view feedback given by patients
8. **Logout**

Patient

1. **Register** - Patient has to first register with the app with their details like Name, Mail, Mobile, Age, Gender, Username and Password.
2. **Login** - Patient can then login with the registered name and password
3. **My Details** - Displays all the details to the patient.
4. **Disease Prediction** - Given the symptoms of the disease, the app displays the suspected disease and also suggests doctors along with their details like Name, Address and Contact Number.
5. **Book Appointment** - Patient can book an appointment by giving the following details
 - a. Specialization of doctor
 - b. Date
 - c. Time

If the doctors are available, then the names of doctors will be displayed. Patient can then book an appointment with the doctor of his choice. An SMS will be sent to the

patient and the concerned doctor after booking

6. Search Doctor - Patient can search for doctors based on the following

- a. Name
- b. Specialization
- c. Address

7. Search Disease - Patient can search for a disease.

8. Feedback - Patient can give feedback to the admin on the service obtained.

9. Logout

Doctor

1. Login - Doctor can login to the app with predefined username and password

2. View Disease - Doctor can view the list of diseases

3. View Patient - Doctor can view his patient details

4. Notifications - It lists the details of the patient like his ID, symptoms, disease type, predicted disease, date and time.

5. My Details - Doctor can view his details that are added by the admin

Non-Functional Requirements

Performance

The app must be interactive and the delays involved must be less. So in every action-response of the app, there are no immediate delays. Resource consumption of this application should not reach an amount that renders the mobile device unusable. The application should be capable of operating in the background should the user wish to utilize other applications.

Usability

The app must be easy to handle and navigate in the most expected way with no delays. The app should transverse quickly between its states.

Reliability

The app should meet all of the functional requirements without any unexpected behavior. At no time should the gauge output display incorrect or outdated information without alerting the user to potential errors. The app is less prone to errors as it avoids errors as much as possible while entering data. Also appropriate error messages are displayed when invalid data is entered.

Availability

The app will be available at all times on the user's Android device, as long as the device is in proper working order.

Maintainability

The software should be written clearly and concisely. The code will be well documented. Particular care will be taken to design the software modularly to ensure that maintenance is easy.

Portability

This software will be designed to run on any Android operating system version 4.0 or higher. The software will be forward compatible for all currently released Android operating system versions.

Response time

The time taken by the system to complete a task given by the user must be very less. The application must respond instantly to the task requested.

SOFTWARE REQUIREMENTS

| | |
|--------------------------------------|---------------------|
| Operating System | Windows 7 or higher |
| Languages Used | Java, XML |
| Development Environment (IDE) | Android Studio |
| Database | SQLite |

HARDWARE REQUIREMENTS

| | |
|------------------|-----------------|
| Hard disk | Minimum of 40GB |
| RAM | 4 GB or higher |
| Device | Android mobile |

Software Engineering Model Used

The Software Engineering Model used in this project is the Iterative Model. It is also known as the Iterative and Incremental Development (IID) model. It is a more flexible and adaptive approach to software development. Instead of following a linear sequence, the development process is divided into smaller iterations or cycles. After each iteration, the software is reviewed, and feedback is collected. Based on this feedback, changes and improvements are incorporated in subsequent iterations. This process continues until the final product meets the desired requirements and quality.

Project Scheduling

| Android App | 16-12-2019 | 02-02-2020 | 18-01-2020 | 24-02-2020 | 18-03-2020 | 21-03-2020 |
|---|------------|------------|------------|------------|------------|------------|
| Synopsis Submission | | | | | | |
| Updated Synopsis Submission after review | | | | | | |
| Literature Survey | | | | | | |
| Literature Survey | | | | | | |
| Technology Survey | | | | | | |
| Feasibility Study | | | | | | |
| SRS | | | | | | |
| Design Analysis | | | | | | |
| UML diagrams | | | | | | |
| UI Design | | | | | | |
| Implementation and Testing | | | | | | |
| Implementation | | | | | | |
| Test Case Documentation | | | | | | |
| Testing | | | | | | |
| Documentation of Report | | | | | | |

SYSTEM DESIGN

The design document that we develop during this phase is the blueprint of the software. It describes how the solution to the customer problem is to be built. The design activity begins when the requirements document for the software to be developed is available. While the requirements specification activity is entirely in the problem domain, design is the first step in moving from the problem domain toward the solution domain. Design is essentially the bridge between requirements specification and the final solution for satisfying the requirements.

The design of a system is essentially a blueprint or a plan for a solution for the system. The design process for software systems, often, has two levels. At the first level, the focus is on deciding which modules are needed for the system, the specifications of these modules, and how the modules should be interconnected. This is what is called the system design or top-level design. In the second level, the internal design of the modules, or how the specifications of the module can be satisfied, is decided. This design level is often called detailed design or logic design. Detailed design essentially expands the system design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding.

System Architecture

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.



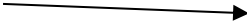

Architectural design

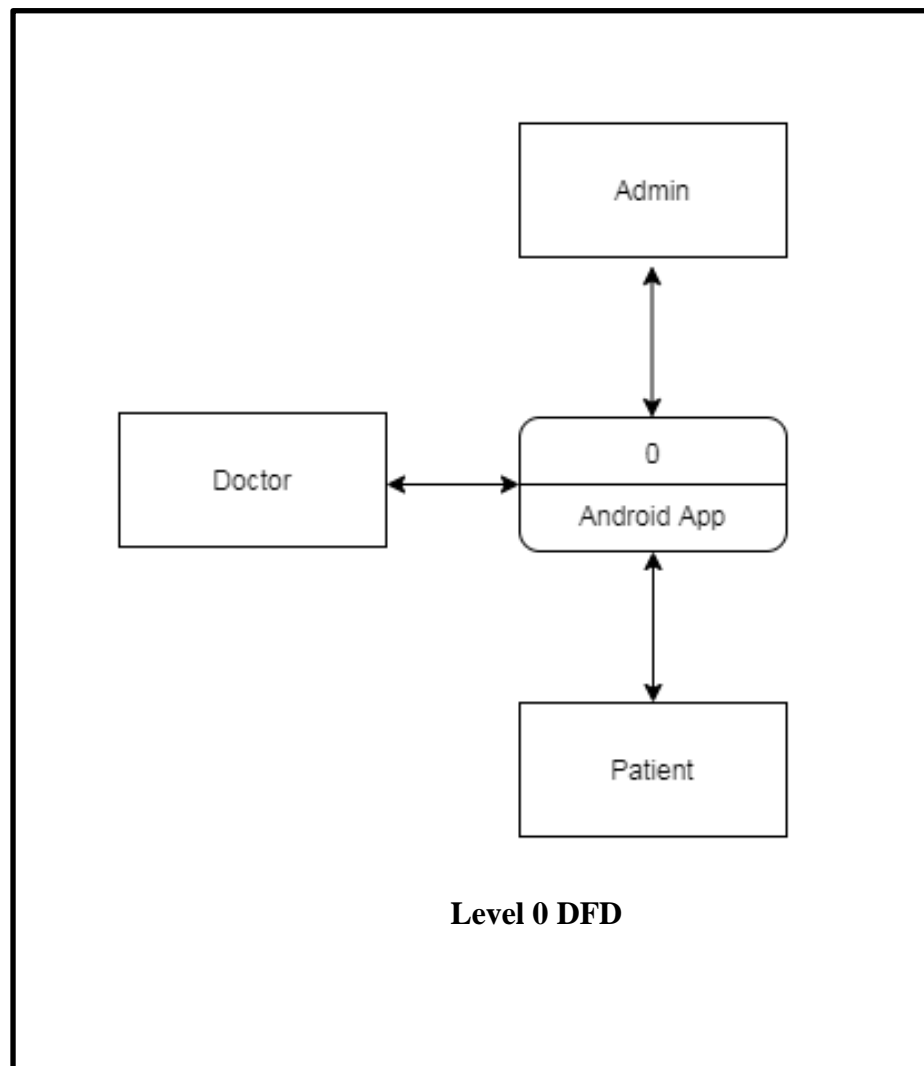


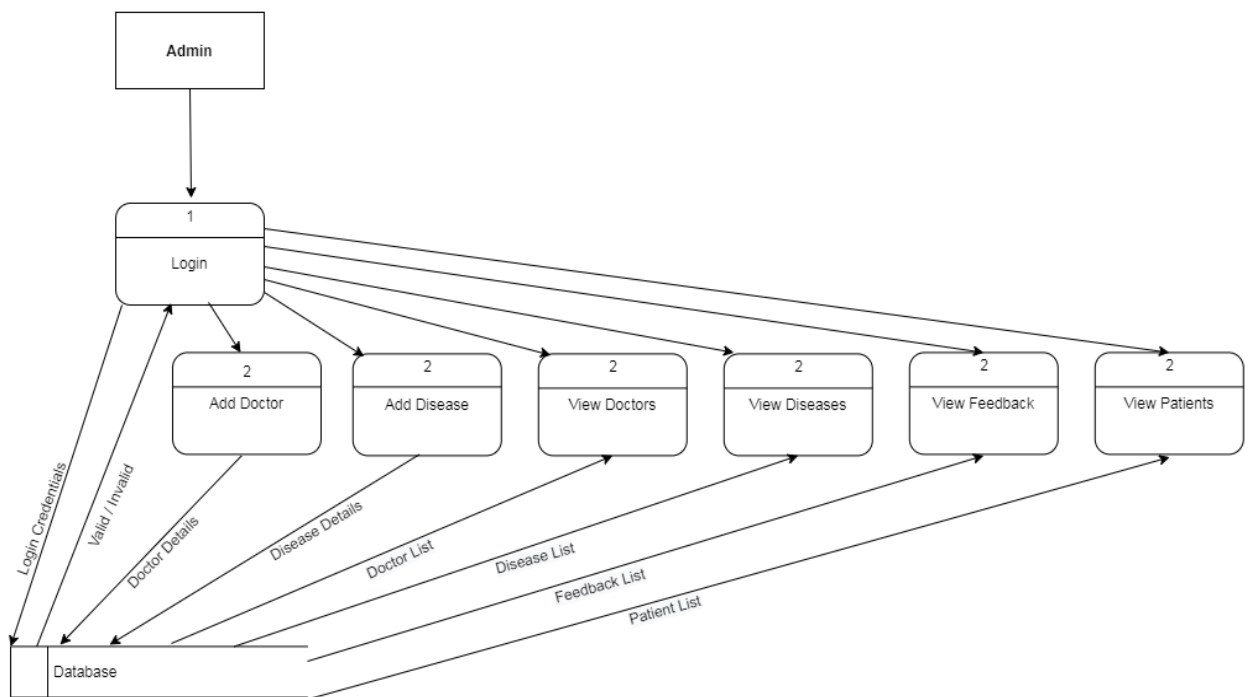
Dataflow Diagram

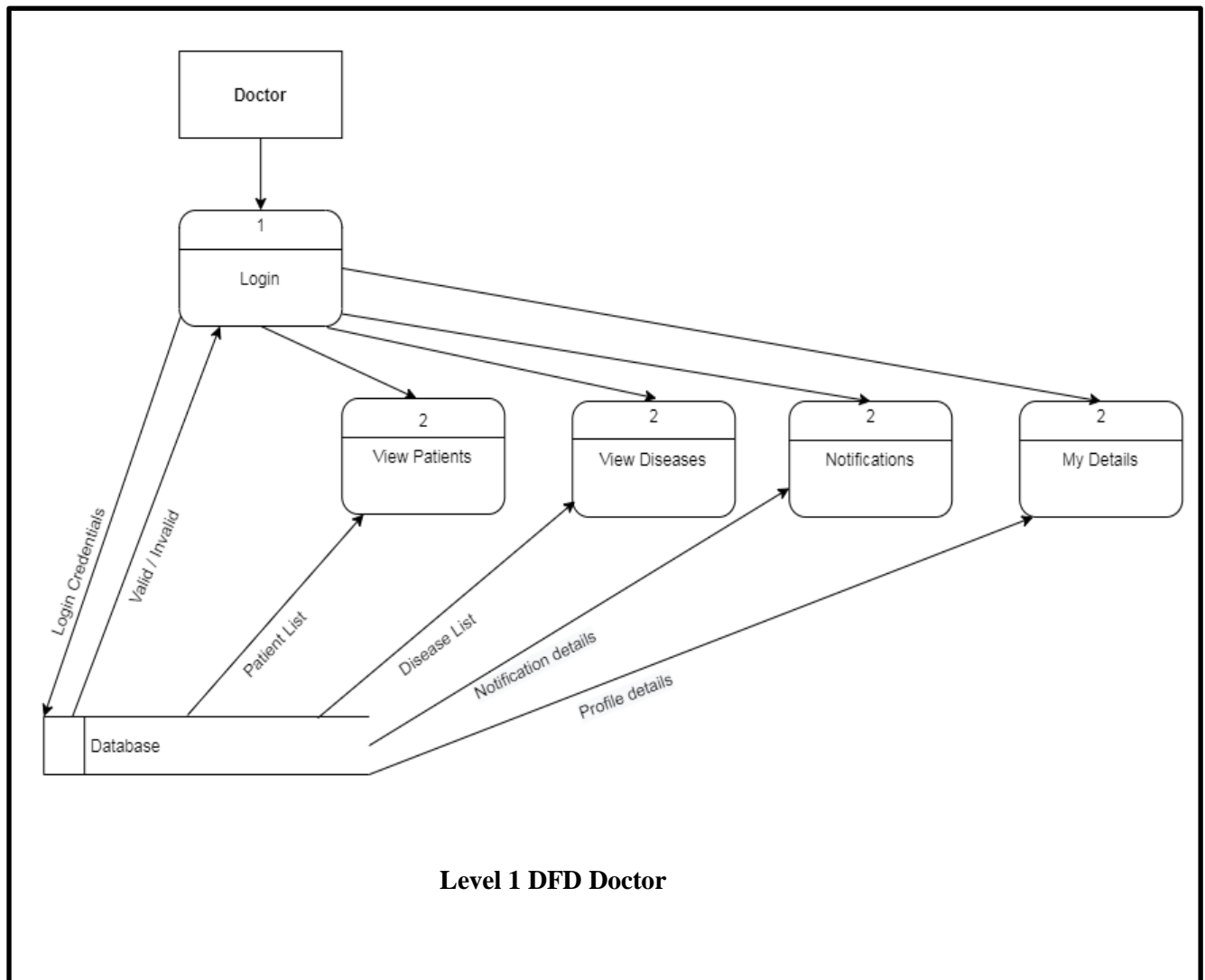
A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO.

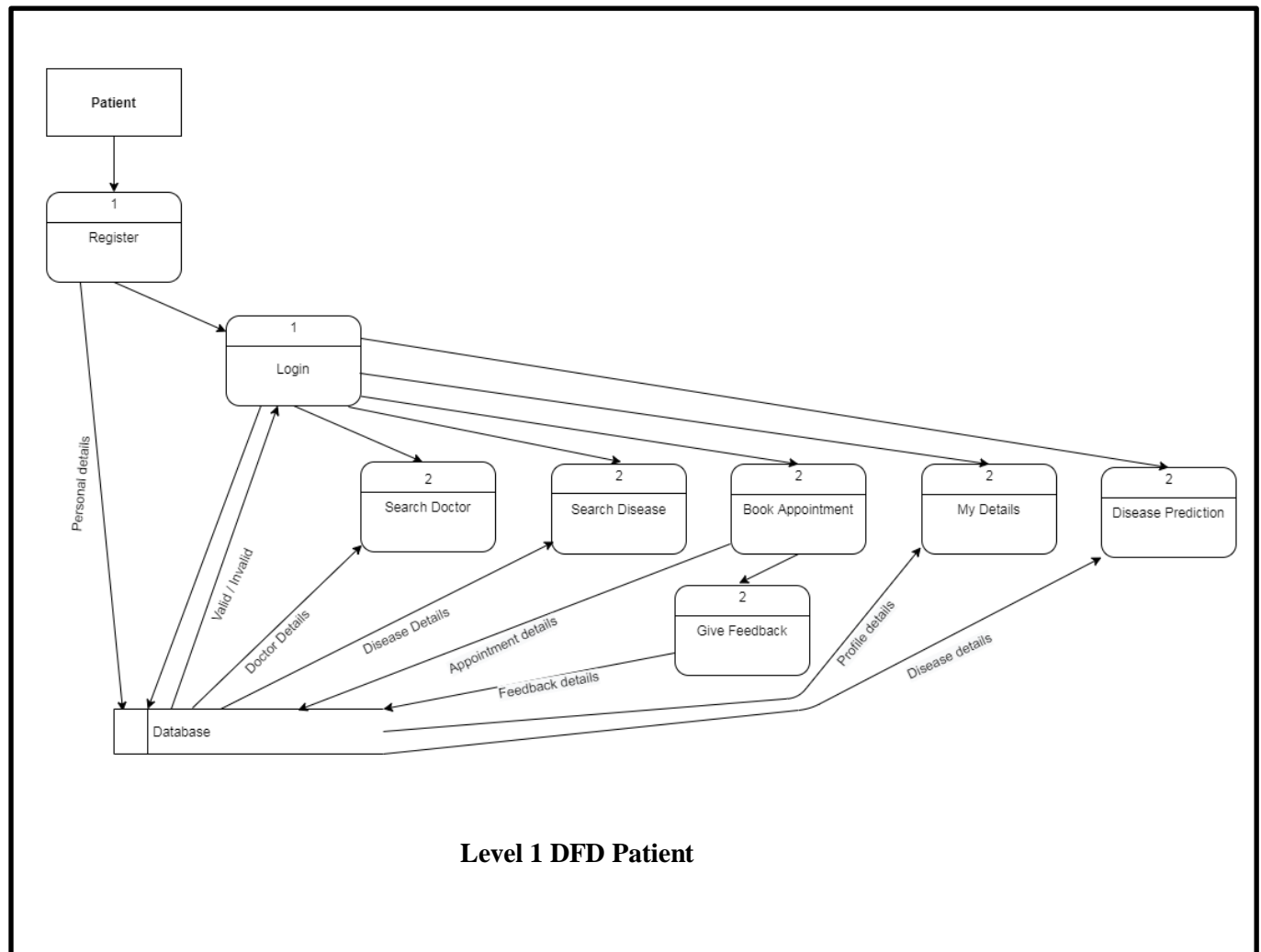
Symbols and descriptions of DFD

| | |
|---|-------------------|
|  | Process |
|  | External Entities |
|  | Dataflow |
|  | Data Store |



**Level 1 DFD Admin**


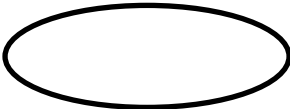
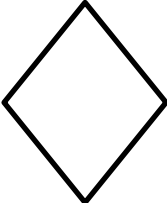
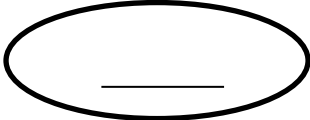



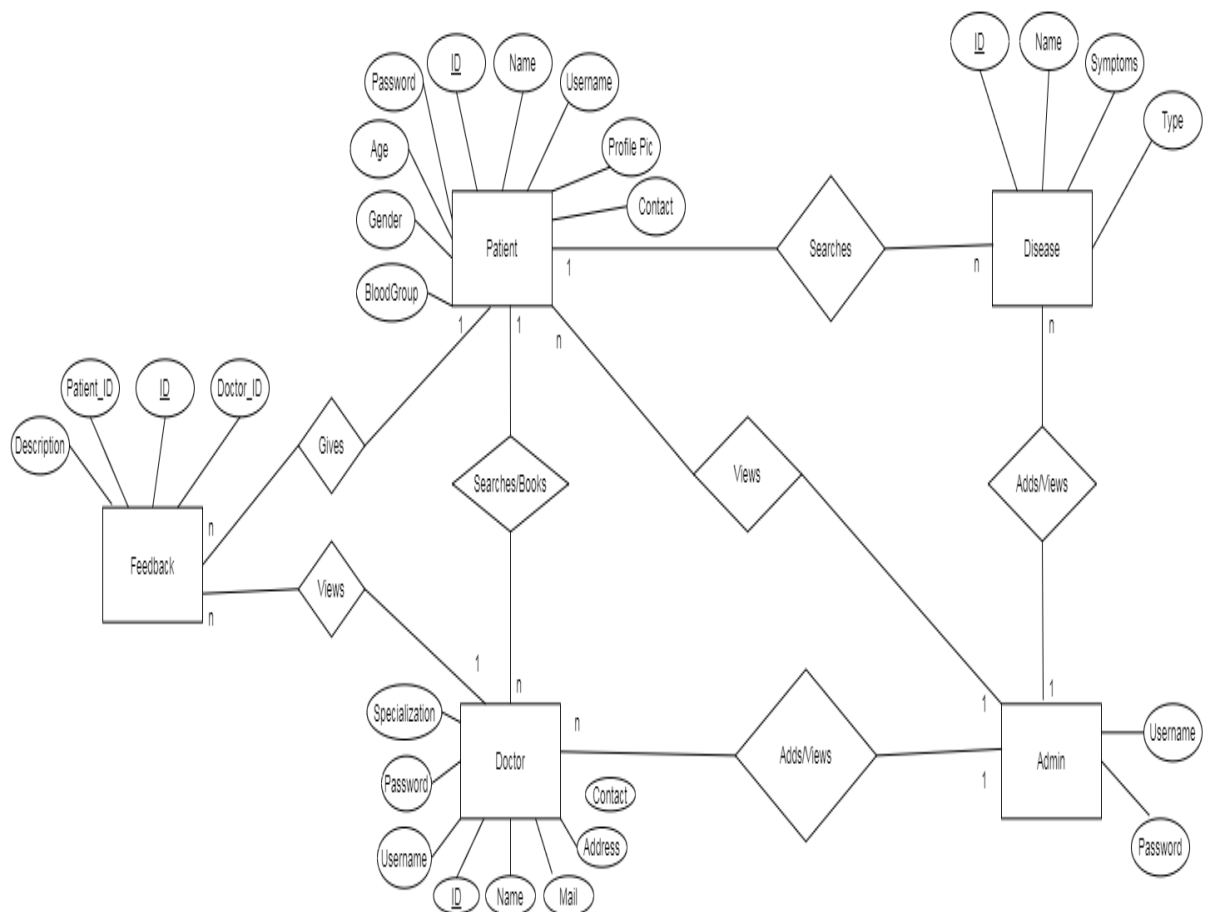


ER Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

Symbols and descriptions of ER Diagram

| | |
|---|---------------|
|  | Entity |
|  | Attribute |
|  | Relationship |
|  | Key Attribute |
|  | Link |



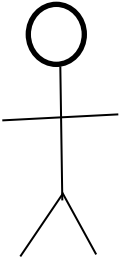


Use case Diagram

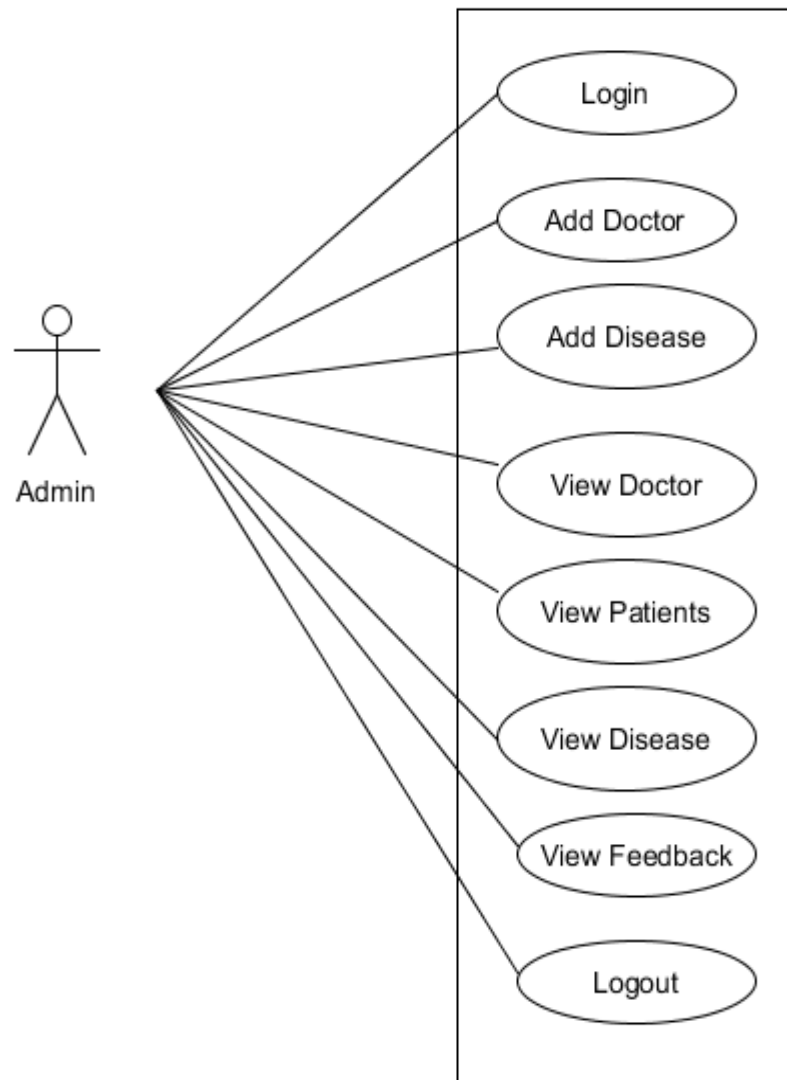
A use case diagram is a graphical representation of the user's interaction with the system. It can portray the different types of users of a system and the various ways they interact with the system. Use cases are diagrammed to be easily understood, no matter who is looking at the diagram.

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. An effective use case diagram helps to represent:

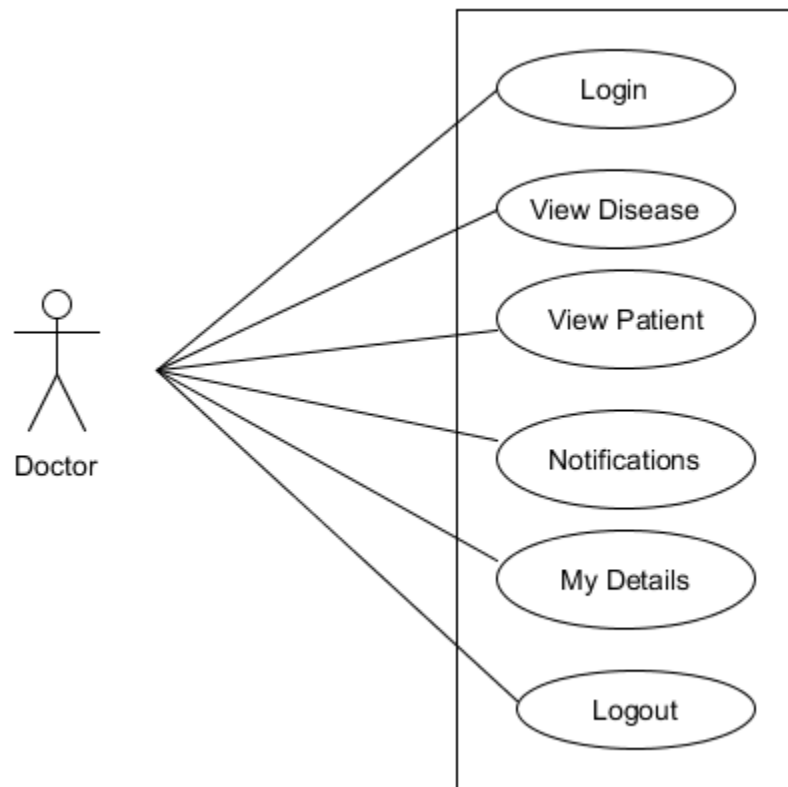
1. Scenarios in which your system or application interacts with people, organizations, or external systems
2. Goals that it helps those entities (known as actors) achieve
3. The scope of your system

Symbols and descriptions of usecase Diagram

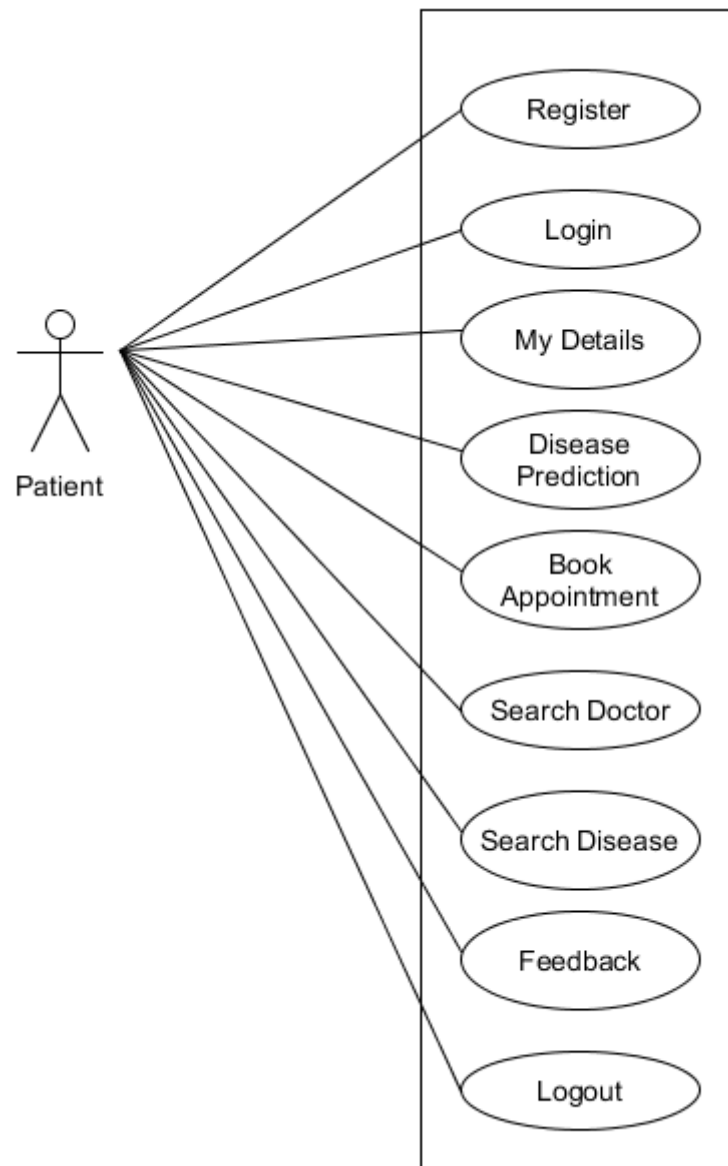
| | |
|---|-------------|
|  | Actor |
|  | Usecase |
|  | Association |



Usecase - Admin



Usecase - Doctor



Usecase - Patient

Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

A class represents a concept which encapsulates state (attributes) and behavior (operations). Each attribute has a type. Each operation has a signature. The class name is the only mandatory information.



Class Name:

- The name of the class appears in the first partition.

Class Attributes:

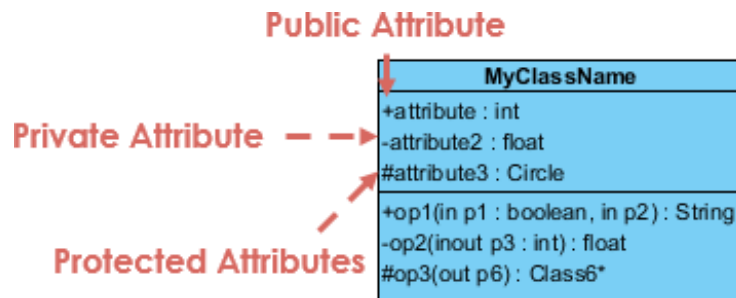
- Attributes are shown in the second partition.
- The attribute type is shown after the colon.
- Attributes map onto member variables (data members) in code.

Class Operations (Methods):

- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.
- The return type of method parameters are shown after the colon following the parameter name. Operations map onto class methods in code.

Class Visibility

The +, - and # symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.



- + denotes public attributes or operations
- - denotes private attributes or operations
- # denotes protected attributes or operations

Relationships between classes

A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:

