|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Patient Mangement System  Advisor Name: Dr. Ahmad Trabolsi  Course Name: Final Year Project  Course Code: CSCI420  Done By: Hadi Soufan - 1121050 |  |
|  |  |  |  |
| Pipette dropping liquid in a petri dish | | | |

Contents

[1](#_Toc159586139)

[Advisor Name: Dr. Ahmad Trabolsi 1](#_Toc159586140)

[Course Name: Final Year Project 1](#_Toc159586141)

[Course Code: CSCI420 1](#_Toc159586142)

[Done By: Hadi Soufan - 1121050 1](#_Toc159586143)

[1](#_Toc159586144)

[Patient Management System 3](#_Toc159586145)

[Introduction about the System: 3](#_Toc159586146)

[THE GENERAL OBJECTIVE OF THIS ARTICLE: 3](#_Toc159586147)

[THE GENERAL STRUCTURE OF THE PAPER: 3](#_Toc159586148)

[The Proposal 3](#_Toc159586149)

[Functional and Non-Functional Requirements 7](#_Toc159586150)

[Programmer Questions: 8](#_Toc159586151)

[THE SYSTEM 9](#_Toc159586152)

[The Admin 12](#_Toc159586153)

[The Users 15](#_Toc159586154)

[NON-FUNCTIONAL REQUIREMENTS 16](#_Toc159586155)

[GANTT CHART: 18](#_Toc159586156)

[Use Case 19](#_Toc159586157)

[WHAT IS THE USE CASE DIAGRAM? 19](#_Toc159586158)

[IMPORTANT POINT: 20](#_Toc159586159)

[Use Case 20](#_Toc159586160)

[Actor 20](#_Toc159586161)

[Association 21](#_Toc159586162)

[System 21](#_Toc159586163)

[Include 22](#_Toc159586164)

[Extend: 22](#_Toc159586165)

[Dependency 22](#_Toc159586166)

[Generalization 22](#_Toc159586167)

[THE COMMON THINGS FOR USERS: 24](#_Toc159586168)

[Login and Sign-up use case: 24](#_Toc159586169)

[Profile use case 24](#_Toc159586170)

[WHAT IS NARRATIVE USE CASE? 26](#_Toc159586171)

[A use case narrative is a largely text-based description of a use case that could be supplemented with decision trees or other easily understood notations. The description should be written in the use r’s language, and thus provides an important communication tool between developers of systems and the intended use rs. (GLOBAL, n.d.) 26](#_Toc159586172)

[NARRATIVE USE CASE FOR THE STORE 26](#_Toc159586173)

[ER Diagram 27](#_Toc159586174)

[WHAT IS DATA MODELING? 27](#_Toc159586175)

[WHAT IS ER DIAGRAM?](#_Toc159586176)

[The components and features of an ER diagram 28](#_Toc159586177)

# Patient Management System

## Introduction about the System:

A Patient Management Software (PMS) can help to streamline your practice and improve patient care. By automating many of the tasks associated with patient care, a **PMS** can save you time and money.

## THE GENERAL OBJECTIVE OF THIS ARTICLE:

The main objective of this article is to explain the idea of the program and explain

the smallest details about how it works and the things it provides.

## THE GENERAL STRUCTURE OF THE PAPER:

This page is divided into several sections in order to organize the process of explaining this system and thus ease of understanding the parts of the program and its precise mechanism of action. The sections will be displayed as follows:

1. The Proposal
2. Functional and non-functional requirements
3. Use case diagram
4. ER diagram
5. Process Modeling

# The Proposal

Aim of this Program

The primary aim of the Patient Management System (PMS) is to streamline and enhance the management of patient-related information and workflows within a healthcare facility. This encompasses a range of objectives aimed at improving operational efficiency, clinical outcomes, and patient satisfaction.

Problem Overview

Healthcare professionals have constantly faced challenges that can impact their work. From billing and health information to record-keeping to medication management, healthcare professionals need a system. It can help them manage their tasks smoothly and efficiently.

Effective communication among different departments within a healthcare facility is crucial to avoid confusion and frustration for patients and to maintain operational efficiency. Patient Management Software offers a centralized platform to address these challenges.

This software automates various tasks, such as sending appointment and prescription reminders, tracking lab results and medication dosages, and managing financial aspects of patient care. By streamlining communication and data sharing, clinicians can collaborate more effectively, identify patterns, and devise solutions efficiently.

By leveraging Patient Management Software, healthcare professionals can work collaboratively toward common goals, ensuring quality patient care while optimizing operational processes.

Medical Center Problems

Healthcare professionals have constantly faced challenges that can impact their work. From billing and health information to record-keeping to medication management, healthcare professionals need a system. It can help them manage their tasks smoothly and efficiently.

Poor communication between different departments can lead to confusion and frustration for the patients, as well as a decrease in efficiency.

The Solution

Patient Management Software can help solve these problems by providing a centralised platform.

The Patient Management app helps streamline communication by automating certain tasks. This includes sending reminders about appointments or prescriptions, tracking lab results and medication dosages, and keeping track of financials related to patient care.

Additionally, it allows clinicians to share data easily across different systems so that they can identify patterns and find solutions faster.

By usingPatient Management Software, healthcare professionals can confidently work together towards common goals while maintaining quality patient care.

How to select the perfect Patient Management system for your practice?

If you're thinking about implementing a Patient Management System (PMS), there are a few things to consider. Here are a few key factors to keep in mind when choosing the right PMS.

1. Cost - How much will it cost to buy and/or install the system? Is there an annual subscription fee, or does the system require periodic fees?
2. Flexibility - What capabilities does the system offer? Does it allow for easy customization of forms and reports, or is everything preset from the start?
3. Ease of Use - How user-friendly is the interface? Will users be able to easily fill out forms, submit data electronically, or access information they need quickly?
4. Support Services - Can you get help with configuring or troubleshooting the system if needed?
5. Do you only need basic functionality or would you like more features than most systems offer?
6. What type of data do you need to track? Do you primarily focus on appointments, medications, lab test results, etc.?

HOW TO VALIDATE THE SUCCESS

As the healthcare industry continues to grow and change, it is important for doctors and administrators to have reliable software. So, it can help them manage patient data.

One of the most popular tools for this task is Patient Management Software. Patient Management Software helps clinicians keep track of patients' records, relationships with other patients, and more.

It also allows doctors to collect information about their patients in a secure way so that they can efficiently provide care.

This system provides users with a wide range of features. This includes easy data entry, powerful reporting capabilities, and encrypted communication between users and administrators.

Additionally, Patient Management Software offers complete confidentiality while preserving patient privacy rights.

SYSTEM USERS

1. Anyone
2. Medical Center that wants to manage and maintain patients.

THE PLAN

1. Needs Assessment and Requirements Gathering: Conduct a comprehensive needs assessment to identify medical center's objectives, gather input from stakeholders, and understand workflow, pain points, and desired features for PMS.
2. Customization and Configuration: The software will be customized to meet the specific needs of the medical center, including adjusting workflows, templates, and user interfaces to suit their preferences.
3. Creating the Software Design.
4. Creating the database system.
5. Implementing all the required features
6. Link the program with database.
7. Testing and solving the problems.
8. Deployment of the Product and Maintenance and Operations.

TIME PLAN

Here will be a table that divide the tasks and the required time for each task.

RESOURCES

The software required for this system is the main website and its corresponding database. No associated hardware is needed.

# Functional and Non-Functional Requirements

One of the basic things that you must pay attention to in order to succeed in any system are the requirements.

The requirements are separated into two parts:

1. Functional Requirements.
2. Non-Functional Requirements

In software engineering and systems engineering, a functional requirement defines a function of a system or its component, where a function is described as a specification of behaviour between inputs and outputs. (Wikipedia, Wikipedia, n.d.) However, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements. (Wikipedia, Wikipedia, n.d.)

 After a lengthy study was conducted with a group of experts and after a group of random opinions were taken from the people in order to achieve success for this project.  And after doing some searches on the internet. A set of basic points have been established that the system must follow.

 These points are under the requirements label. In order to achieve the best success of this project, these requirements will be sorted into sections:

1. The system.
2. The admin.
3. The users.
4. The main software.

REQUIREMENTS GATHERING PROCESS DETAILS

## Programmer Questions:

**Question 1:** How do you plan to architect the Patient Management System using ASP.NET Core Web API for the backend and React.js for the frontend?

Possible Answers:

I propose utilizing ASP.NET Core Web API to develop a RESTful backend that exposes endpoints for managing patient data, appointments, and other system functionalities. With React.js on the frontend, we can create a dynamic and responsive user interface that interacts seamlessly with the backend API, providing a modern and intuitive user experience for healthcare providers and staff.

My approach involves leveraging ASP.NET Core Web API to build a robust and scalable backend infrastructure capable of handling complex data processing and business logic. By integrating React.js on the frontend, we can develop interactive user interfaces with reusable components, state management, and client-side routing, ensuring optimal performance and user engagement throughout the Patient Management System.

I intend to architect the Patient Management System using ASP.NET Core Web API for the backend, taking advantage of its powerful features for building secure and high-performance APIs. Complemented by React.js on the frontend, we can create a dynamic and responsive user interface that enhances usability and productivity for healthcare professionals, while ensuring seamless integration with the backend services for efficient data retrieval and management.

**Question 2**: How do you plan to approach system scalability and performance optimization without relying on cloud services to accommodate the growing needs of the medical center and ensure optimal user experience for healthcare providers and patients?

Possible Answers:

To address scalability concerns without leveraging cloud services, I will design the Patient Management System with a modular architecture that allows for easy horizontal scaling on traditional on-premises infrastructure. This involves breaking down the system into smaller, independently deployable components that can be scaled individually as demand increases. Additionally, I will optimize database indexing and query performance to handle larger volumes of data efficiently.

System performance optimization will be achieved through meticulous code optimization, caching strategies, and resource pooling techniques. By minimizing resource usage and optimizing algorithms, we can ensure that the Patient Management System operates smoothly even under heavy load. Furthermore, I will implement distributed caching mechanisms and in-memory data stores to reduce latency and improve response times for critical operations.

Without relying on cloud-based autoscaling features, I will implement manual scaling strategies that involve closely monitoring system metrics and proactively adding or removing hardware resources as needed. This may involve setting up redundant servers, load balancers, and database replicas to distribute workloads and ensure high availability. Additionally, I will conduct regular performance testing and tuning to identify potential bottlenecks and optimize system components for peak efficiency.

FUNCTIONAL REQUIREMENTS

## THE SYSTEM

1. **User Authentication and Authorization:**

The system shall provide user authentication mechanisms to verify the identity of healthcare providers, administrators, and patients.

Users shall be assigned role-based access permissions to ensure appropriate access levels to system functionalities and patient data.

1. **Medical History Management:**

The system shall allow healthcare providers to create, update, and maintain comprehensive electronic medical records (EMRs) for each patient.

Healthcare providers shall be able to record and track medical history, diagnoses, medications, allergies, lab results, and other relevant health information.

1. **Appointment Scheduling:**

The system shall enable healthcare providers to schedule appointments for patients, specifying date, time, location, and provider.

Patients shall have the ability to request appointments through the patient portal, indicating preferred dates and times.

The system shall support the management of recurring appointments, cancellations, and rescheduling requests.

1. **Billing System:**

The system shall generate invoices and statements for patient visits, procedures, and services rendered.

Integration with insurance providers shall allow for automatic verification of coverage, claims submission, and adjudication.

Patients shall have access to their billing statements and payment history through the patient portal.

1. **Patient Portal:**

The system shall provide patients with secure access to their medical records, appointment schedules, and billing information.

Patients shall be able to update personal information, complete electronic forms, and communicate with healthcare providers securely.

The patient portal shall support features such as prescription refills, lab results viewing, and educational resources.

1. **Appointment System with Customizable Slots and Reminders:**

Healthcare providers shall be able to define customizable appointment slots based on their availability and preferences.

The system shall send real-time appointment reminders to patients via email, SMS, or in-app notifications.

Patients shall have the option to confirm or cancel appointments directly from the reminder notifications.

1. **Waitlist Management:**

The system shall maintain a waitlist of patients requesting appointments for fully booked time slots.

Healthcare providers shall be notified of available slots and be able to offer appointments to patients on the waitlist.

1. **Customizable Notifications:**

The system shall support customizable notifications for appointment reminders, billing statements, test results availability, and other relevant events.

Healthcare providers and patients shall have the ability to configure notification preferences based on their communication preferences.

1. **Telemedicine System:**

The system shall facilitate virtual consultations between healthcare providers and patients through secure video conferencing.

Patients shall be able to schedule telemedicine appointments, complete pre-appointment assessments, and join virtual sessions from the patient portal.

The system shall integrate with the appointment scheduling and billing systems to streamline telemedicine workflows.

1. **Billing System with Invoice, Insurance, and Patient Billing:**

The system shall generate itemized invoices for services rendered, including consultation fees, procedures, and medications.

Integration with insurance providers shall allow for automated claims submission, adjudication, and payment processing.

Patients shall receive detailed billing statements with information on insurance coverage, co-payments, and outstanding balances.

The system shall support flexible payment plans and financing options for patients requiring assistance with healthcare expenses.

## The Admin

1. **User Management:**

Create, update, and delete user accounts for healthcare providers, administrative staff, and other system users.

Assign roles and permissions to users based on their responsibilities and access requirements.

Manage password policies, including password complexity requirements and expiration periods.

1. **Patient Management:**

Add new patients to the system and maintain patient records with demographic information, contact details, and medical history.

Update patient information as needed, including changes in personal details, insurance coverage, and emergency contacts.

Merge duplicate patient records and resolve data inconsistencies to ensure accurate and up-to-date patient information.

1. **Appointment Management:**

View and manage appointment schedules for healthcare providers and clinics, including appointment slots, availability, and booking status.

Reschedule or cancel appointments on behalf of patients or healthcare providers as necessary.

Monitor appointment attendance rates, no-shows, and waitlist statuses to optimize scheduling and resource allocation.

1. **Billing and Financial Management:**

Generate and review billing statements, invoices, and financial reports for patient visits, procedures, and services rendered.

Manage insurance claims submission, processing, and reconciliation to ensure accurate and timely reimbursement.

Monitor accounts receivable, outstanding balances, and payment collections to maintain financial health and compliance.

1. **System Configuration and Settings:**

Customize system settings and preferences, including appointment types, duration, and scheduling rules.

Configure billing codes, fee schedules, and payment processing settings to align with organizational policies and regulatory requirements.

Manage system integrations with third-party software applications and services, such as electronic health records (EHR) systems and payment gateways.

1. **Reporting and Analytics:**

Generate and analyze reports on key performance indicators (KPIs), such as appointment utilization, patient demographics, and revenue trends.

Monitor system metrics and performance indicators to identify areas for improvement and optimize operational efficiency.

Export reports in various formats for further analysis and sharing with stakeholders, including healthcare providers, administrators, and regulatory agencies.

1. **Security and Compliance:**

Implement access controls, encryption mechanisms, and data backup procedures to safeguard against unauthorized access, data breaches, and data loss.

System Maintenance and Support:

Monitor system performance, availability, and uptime to proactively address issues and minimize downtime.

Coordinate software updates, patches, and maintenance tasks to ensure the stability and reliability of the system.

Provide technical support and troubleshooting assistance to users, including healthcare providers, staff, and patients, to resolve issues and optimize system usability.

## The Users

1. **Administrators**

Manage user accounts: Administrators can create, update, and delete user accounts for healthcare providers, administrative staff, and other system users. They can also assign roles and permissions to control access to system functionalities and patient data.

Configure system settings: Administrators can customize system settings, preferences, and configurations, including appointment types, scheduling rules, billing codes, and payment processing settings.

Manage patient records: Administrators can oversee patient management tasks, such as adding new patients to the system, updating patient information, and resolving data inconsistencies.

Monitor system performance: Administrators can monitor system metrics, performance indicators, and security logs to ensure the stability, reliability, and security of the system.

Generate reports: Administrators can generate and analyze reports on key performance indicators (KPIs), patient demographics, appointment utilization, billing statistics, and other relevant metrics.

1. **Healthcare Providers (doctors, nurses …):**

View patient records: Healthcare providers can access and review patient medical records, including medical history, diagnoses, medications, allergies, lab results, and treatment plans.

Schedule appointments: Healthcare providers can schedule appointments for patients, specifying date, time, location, and appointment type. They can also manage appointment schedules, cancellations, and rescheduling requests.

Document patient encounters: Healthcare providers can document patient encounters, including progress notes, assessments, treatments, and follow-up recommendations.

Communicate with patients: Healthcare providers can communicate with patients securely through the system, send messages, share test results, and provide instructions or educational materials.

Perform telemedicine consultations: Healthcare providers can conduct virtual consultations with patients through secure video conferencing, document telemedicine encounters, and prescribe medications or treatment plans as needed.

1. **Patients:**

View personal health information: Patients can access and view their personal health information, including medical history, lab results, prescriptions, and upcoming appointments.

Schedule appointments: Patients can request appointments through the system, indicating preferred dates, times, and providers. They can also view available appointment slots and confirm or cancel appointments as needed.

Communicate with healthcare providers: Patients can communicate securely with their healthcare providers, ask questions, request prescription refills, and provide updates on their health status.

Access educational resources: Patients can access educational materials, health tips, and resources provided by the healthcare facility to support self-care and health management.

Make payments: Patients can view billing statements, invoices, and payment options through the system. They can make payments online, set up payment plans, and view their payment history.

# NON-FUNCTIONAL REQUIREMENTS

1. **Performance:**

The system should respond to user interactions within 2 seconds under normal load conditions.

It should be capable of handling concurrent user sessions without significant degradation in response time.

The system should support a minimum of 1000 concurrent users without performance bottlenecks.

1. **Scalability:**

The system should be horizontally scalable to accommodate an increase in the number of users and data volume.

It should support seamless scaling by adding additional servers or resources to handle growing demands.

The system should be able to scale up or down dynamically based on workload fluctuations.

1. **Reliability:**

The system should have a minimum uptime of 99.9% to ensure availability and reliability for users.

It should have built-in fault tolerance mechanisms to minimize downtime and service interruptions.

The system should be capable of recovering from hardware failures or software crashes without data loss or corruption.

1. **Security:**

The system should enforce robust authentication and authorization mechanisms to protect sensitive patient information.

It should encrypt data transmissions using industry-standard encryption protocols (e.g., SSL/TLS) to prevent unauthorized access.

The system should comply with regulatory standards such as HIPAA to ensure patient data privacy and confidentiality.

1. **Usability:**

The system should have an intuitive user interface that is easy to navigate and use for healthcare providers, administrative staff, and patients.

It should support accessibility standards (e.g., WCAG) to accommodate users with disabilities or special needs.

The system should provide contextual help, tooltips, and user guides to assist users in performing tasks effectively.

1. **Compatibility:**

The system should be compatible with commonly used web browsers such as Chrome, Firefox, Safari, and Edge.

It should support multiple operating systems, including Windows, macOS, and Linux, for both desktop and mobile devices.

The system should be responsive and adaptable to different screen sizes and resolutions for optimal user experience across devices.

1. **Maintainability:**

The system should be modular and well-documented to facilitate future enhancements, updates, and maintenance tasks.

It should adhere to coding standards and best practices to promote code readability, maintainability, and extensibility.

The system should have version control and change management processes in place to track and manage code changes effectively.

# GANTT CHART:

WHAT IS GANTT CHART:

A Gantt chart is a type of bar chart that illustrates a project schedule, named after its

popularizer, Henry Gantt, who designed such a chart around the years 1910–1915. Modern

Gantt charts also show the dependency relationships between activities and the current

schedule status. (Wikipedia, Wikipedia, n.d.)

The Tasks

1. General analysis and requirements gathering.
2. Creating the Product Design.
3. Creating the database system.
4. Generating and connect the system with database.
5. Establishment of the system and the mechanism of communication.
6. Create a space for people to communicate with each other and to present what they do and find useful.
7. link the program with database.
8. Testing and solving the problems.
9. Deployment of the Product and Maintenance and Operations.

The tasks will be displayed in the table based on the letters present before each of the tasks presented above.

TABLE FOR GRANT CHART

# Use Case

INTRODUCTION ABOUT THE USE CASE

## WHAT IS THE USE CASE DIAGRAM?

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures. (Wikipedia, Use Case, n.d.)

## IMPORTANT POINT:

The main thing that must be constantly present during the use of the program, which is

also, a prerequisite for using the program is the login process.

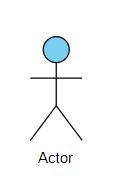
THE DESCRIPTION OF THE USE CASE:

## Use Case



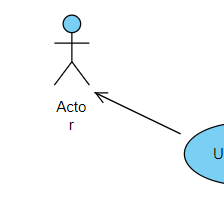
A use case represents a user goal that can be achieved by accessing the system or software application. In Visual Paradigm, you can make use of the sub-diagram feature to describe the interaction between user and system within a use case by creating a sub-sequence diagram under a use case. You can also describe the use case scenario using the Flow of Events editor.

## Actor



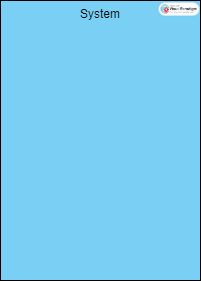
Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association correspond to a sequence of actions between the actor and use case in achieving the use case.

## Association



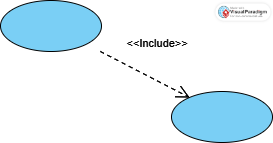
Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association correspond to a sequence of actions between the actor and use case in achieving the use case.

## System



The scope of a system can be represented by a system (shape), or sometimes known as a system boundary. The use cases of the system are placed inside the system shape, while the actor who interact with the system are put outside the system. The use cases in the system make up the total requirements of the system.

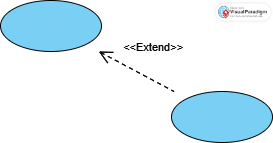
## Include



An include relationship specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.

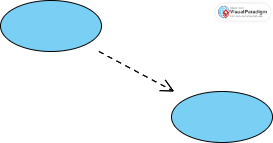
## 

## Extend:



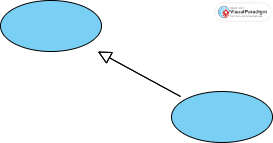
An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.

## Dependency



A dependency relationship represents that a model element relies on another model element for specification and/or implementation.

## Generalization



A generalization relationship is used to represent inheritance relationship between model elements of same type. The more specific model element shares the same specification with. the more general the model element but carries more details in extra.

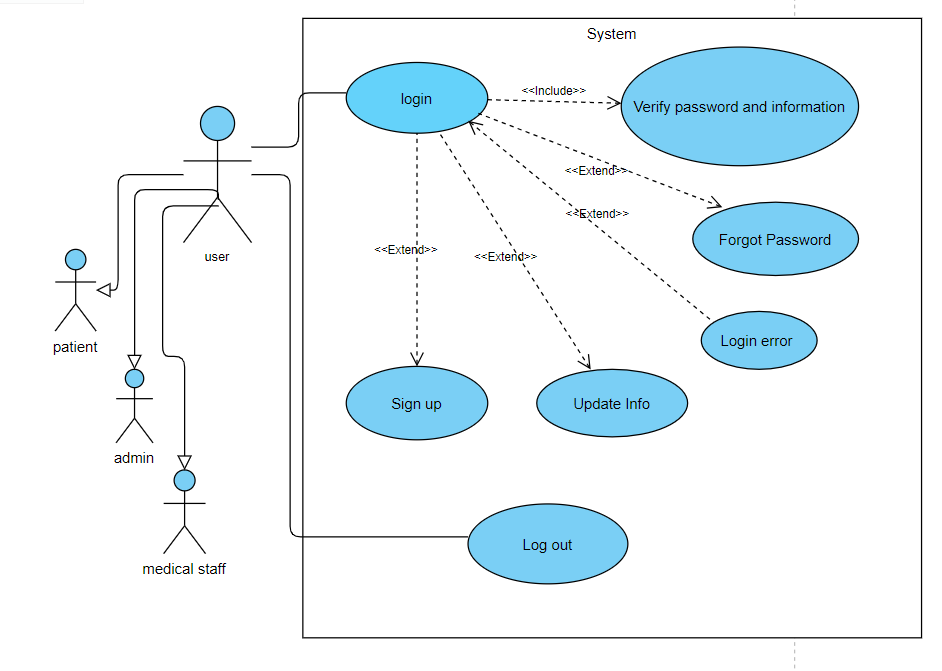
To see more details and forms of the use case, you can go to the following website: <https://online.visual-paradigm.com>

INFORMATION PRESENTATION

## THE COMMON THINGS FOR USERS:

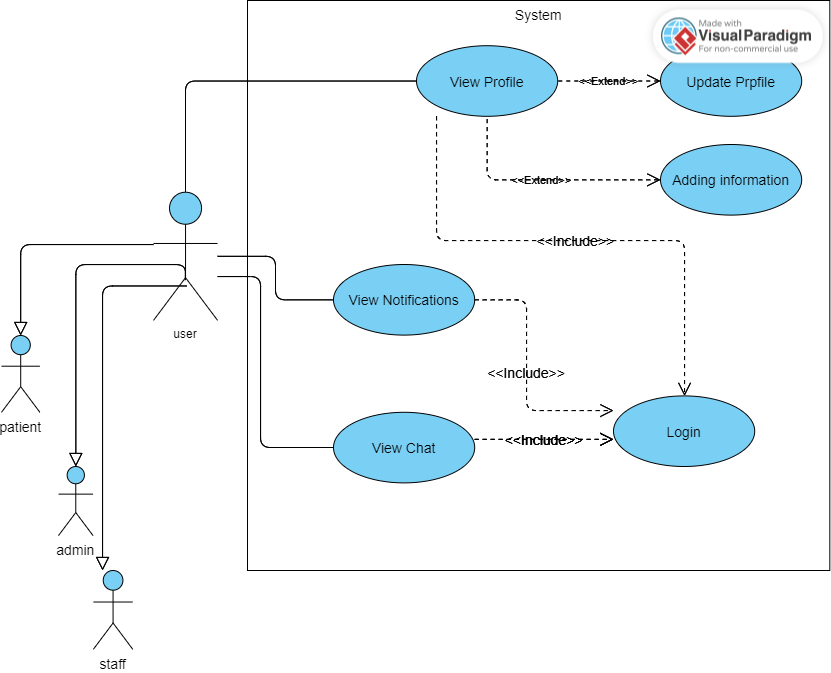
Login and Sign-up use case:

This image shows how the system provides users with the ability to submit an account registration request and how to log in and out.



## Profile use case

This image shows how the system offers users the ability to control their profile and how to search through the system sections.



NARRATIVE USE CASE

## WHAT IS NARRATIVE USE CASE?

## A use case narrative is a largely text-based description of a use case that could be supplemented with decision trees or other easily understood notations. The description should be written in the use r’s language, and thus provides an important communication tool between developers of systems and the intended use rs. (GLOBAL, n.d.)

## NARRATIVE USE CASE FOR THE STORE

* 1. Main Actors
* Admin
* Patient
* Staff
  1. Preconditions
* The user has sign up.
* The user enters his name.
* The user enters his email address.
* The user enters his password.
  1. Normal flow of action
* The user logs in.
* The user entered their email addresses.
* The user entered the password.
* The user can access the pages he is authorized to access base on their role and permission.
  1. Alternative flow of action:
* The user may enter the wrong email.
* The user may enter the wrong password.
* The user may ask to change the password in case he forgot the original one.
* The system will send him a verification message.

This paper explains in a simplified way to all the people who can see it how the system works in detail. Also, some references have been added that the reader can see, which will help him understand the idea more clearly and effectively.

# ER Diagram

INTRODUCTION ABOUT ER DIAGRAM

## WHAT IS DATA MODELING?

Discover how data modeling uses abstraction to represent and better understand the nature of data flow within an enterprise information system. (IBM, n.d.) Like any design process, database and information system design begins at a high level of abstraction and becomes increasingly more concrete and specific. Data models can generally be divided into three categories, which vary according to their degree of abstraction. The process will start with a conceptual model, progress to a logical model and conclude with a physical model.

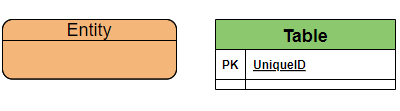
## WHAT IS ER DIAGRAM?

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. (lucidchart, n.d.)

## 

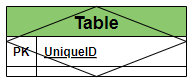
## The components and features of an ER diagram

1.1.1.1 Entity:



A definable thing—such as a person, object, concept or event—that can have data stored about it. Think of entities as nouns. Examples: a customer, student, car or product. Typically shown as a rectangle

1.1.1.2 Associative entity:



An associative entity associates entities (or elements) within an entity set.

1.1.1.3 Cardinality:

Defines the numerical attributes of the relationship between two entities or entity sets. The three main cardinal relationships are one-to-one, one-to-many, and many-many.

##### What is used in this graph is:

One-to-One



One or Zero



One-to-Many



Many-to-Many



One-to-One or Many



One-to-Zero or one or Many



Generalization



1.1.1.4 Primary Key:

Primary Key is a set of attributes (or attribute) which uniquely identify the tuples in relation or table. The primary key is a minimal super key, so **there is one and only one primary key in any relationship**. It will be denoted in the table as (**PK**).

1.1.1.5 Foreign Key:

A foreign key is a set of attributes in a table that refers to the primary key of another table. The foreign key links these two tables. It will be denoted in the table as (**FK**).

1.1.1.6 Attribute:

A descriptive property or characteristic of an entity. Synonyms include element, property, and field.

Example:

Student entity have {First name, Last name, Address, …….}

1.1.1.7 Data type:

A property of an attribute that identifies what type of data can be stored in that attribute.

Example:

Number 🡪 Integer

Text 🡪 Varchar

Explanation of the tables and what they include:

* Center Table:

This table will store the different centers in different regions.

Table 1 Center Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| center\_id | int (pk) | yes | Unique key for the center. |
| center\_location | varchar (50) | yes | Center Location stored in English. |
| center\_locationAr | varchar (50) | yes | Center location stored in Arabic. |
| created\_at | datetime timestamp | no | Center created at time |
| updated\_at | datetime timestamp | no | Center updated profile time. |
| is\_deleted | bool | no | Store True if the user is center, otherwise false. |
| center\_phone\_id | int (fk) | yes | Center phone number id foreign key |

* Center Phone Table:

This table is the (1:M) relation between the phone number and the center, each center may have 1 or more phone numbers.

Table 2 Center Phone Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| center\_phone\_id | int (pk) | yes | Unique key for the center phone number. |
| center\_phone | varchar (12) | yes | Center Phone Number stored in English. |
| center\_phone\_id | varchar (12) | yes | Center location stored in Arabic. |
| created\_at | datetime timestamp | no | Center Phone created at time |
| updated\_at | datetime timestamp | no | Center phone updated time. |
| is\_deleted | bool | no | Store True if the user is phone, otherwise false. |
| center\_id | int (fk) | yes | Center id foreign key. |

* Building Table:

This table will store all the buildings that belong for each center. Each center may have 1 or more building (1:M).

Table 3 Building Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| building\_id | int (pk) | yes | Unique key for the building. |
| building\_name | varchar (50) | yes | Building name stored in English. |
| building\_nameAr | varchar (50) | yes | Building name stored in Arabic. |
| created\_at | datetime timestamp | no | Building created at time |
| updated\_at | datetime timestamp | no | Building updated profile time. |
| is\_deleted | bool | no | Store True if the user is building, otherwise false. |
| center\_id | int (fk) | yes | Center id foreign key. |

* Floor Table

This table will store all the floors in the building. Each building may have 1 or more floor (1:M).

Table 4 Floor Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| floor\_id | int (pk) | yes | Unique key for the floor. |
| floor\_number | int | yes | Floor number. |
| room\_count | int | yes | Room count stored in Arabic. |
| room\_countAr | int | yes | Room count stores in Arabic. |
| created\_at | datetime timestamp | no | Room created at time |
| updated\_at | datetime timestamp | no | Room updated time. |
| is\_deleted | bool | no | Store True if the room is deleted, otherwise false. |
| building\_id | int (fk) | yes | Building id foreign key. |

* Room Table

This table will store all the rooms in the floor. Each floor may have 1 or more rooms (1:M).

Table 5 Room Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| room\_id | int (pk) | yes | Unique key for the room. |
| room\_code | varchar (10) | yes | Room code. |
| created\_at | datetime timestamp | no | Room created at time. |
| updated\_at | datetime timestamp | no | Room updated time. |
| is\_deleted | bool | no | Store True if the room is deleted, otherwise false. |
| floor\_id | int (fk) | yes | Floor id foreign key. |

* Department Table

This table will store all the departments in the building. Each building may have 1 or more department (1:M).

Table 6 Department Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| department\_id | int (pk) | yes | Unique key for the department. |
| department\_name | varchar (50) | yes | Department name stored in English. |
| department\_nameAr | varchar (50) | yes | Department name stored in Arabic. |
| department\_phone | varchar (12) | yes | Department phone number stored in English. |
| department\_phoneAr | varchar (12) | yes | Department phone number stored in Arabic. |
| department\_email | varchar (50) | yes | Department email. |
| department\_fax | varchar (12) | yes | Department fax stored in English. |
| department\_faxAr | varchar (12) | yes | Department fax stored in Arabic. |
| created\_at | datetime timestamp | no | Department created at time |
| updated\_at | datetime timestamp | no | Department updated time. |
| is\_deleted | bool | no | Store True if the department is deleted, otherwise false. |
| floor\_id | int (fk) | yes | Floor id foreign key. |

* User Table

This table will store all the info needed for the users. These attributes will be shared for all user types (doctors, staff, patients…).

Table 7 User Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| user\_id | int (pk) | yes | Unique key for the user. |
| username | varchar (50) unique | yes | Username of the user. |
| password (hashed) | varchar (max) | yes | User’s password. |
| user\_email | varchar (50) | yes | User’s email. |
| first\_name | varchar (50) | yes | User’s first name stored in English. |
| first\_nameAr | varchar (50) | yes | User’s first name stored in Arabic. |
| father\_name | varchar (50) | yes | User’s father name stored in English. |
| father\_nameAr | varchar (50) | yes | User’s father name stored in Arabic. |
| last\_name | varchar (50) | yes | User’s last name stored in English. |
| last\_nameAr | varchar (50) | yes | User’s last name stored in Arabic. |
| mother\_name | varchar (50) | yes | User’s mother name stored in English. |
| mother\_nameAr | varchar (50) | yes | User’s mother name stored in Arabic. |
| date\_of\_birth | date | yes | User’s date of birth. |
| phone\_number | varchar (12) | yes | User’s phone number stored in English. |
| Phone\_numberAr | varchar (12) | yes | User’s phone number stored in Arabic. |
| nationality | varchar (50) | yes | User’s nationality stored in English. |
| NationalityAr | varchar (50) | yes | User’s nationality stored in Arabic. |
| nationality\_id | int | yes | User’s nationality id stored in English. |
| Nationality\_idAr | int | yes | User’s nationality id stored in Arabic. |
| education | varchar (50) | yes | User’s education level stored in English. |
| educationAr | varchar (50) | yes | User’s education level stored in Arabic. |
| gender | varchar (6) | yes | User’s gender stored in English. |
| genderAr | varchar (6) | yes | User’s gender stored in Arabic. |
| marital\_stauts | varchar (15) | yes | User’s marital status (single, divorced, married) |
| marital\_stautsAr | varchar (15) | yes | User’s marital status stored in Arabic. |
| blood\_type | char (2) | yes | User’s blood type stored in English. |
| blood\_typeAr | char (2) | yes | User’s blood type stored in Arabic. |
| address | varchar (50) | yes | User’s address stored in English. |
| addressAr | varchar (50) | yes | User’s address stored in Arabic. |
| city | varchar (50) | yes | User’s city stored in English. |
| cityAr | varchar (50) | yes | User’s city stored in Arabic. |
| zipcode | int | no | User’s state zipcode stored in English. |
| zipcodeAr | int | no | User’s state zip code stored in Arabic. |
| state | varchar (50) | yes | User’s state stored in English. |
| stateAr | varchar (50) | yes | User’s state stored in Arabic. |
| occupation | varchar (50) | yes | User’s occupation (Job Title) stored in English. |
| occupationAr | varchar (50) | yes | User’s occupation stored in Arabic. |
| insurance \_number | int | no | User’s insurance id stored in English. |
| insurance \_numberAr | int | no | User’s insurance id stored in Arabic. |
| role | varchar (50) | yes | User’s role in the software. |
| rank | int default 0 | yes | User’s rank (from 1 till 10) |
| created\_at | datetime timestamp | no | User’s created at time |
| updated\_at | datetime timestamp | non | User’s updated profile time. |
| is\_deleted | bool | no | Store True if the user is deleted, otherwise false. |

* Doctor Table

This table will store all the info about a doctor.

Table 8 Doctor Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| doctor\_id | int (pk) | yes | Unique key for the doctor. |
| doctor\_license\_id | varchar (50) | yes | License id stored in English. |
| doctor\_license\_idAr | varchar (50) | yes | License id stored in Arabic. |
| created\_at | datetime timestamp | no | Doctor created at time. |
| updated\_at | datetime timestamp | no | Doctor profile updated time. |
| is\_deleted | bool | no | Store True if the doctor is deleted, otherwise false. |
| user\_id | Int (fk) | yes | Doctor info from the user id |
| department\_id | int (fk) | yes | Department id foreign key. |

* Nurse Table

This table will store all the info about a nurse.

Table 9 Staff Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| nurse\_id | int (pk) | yes | Unique key for the nurse. |
| nurse \_license\_id | varchar (50) | yes | License id stored in English. |
| nurse \_license\_idAr | varchar (50) | yes | License id stored in Arabic. |
| created\_at | datetime timestamp | no | Nurse created at time. |
| updated\_at | datetime timestamp | no | Nurse profile updated time. |
| is\_deleted | bool | no | Store True if the nurse is deleted, otherwise false. |
| user\_id | Int (fk) | yes | nurse info from the user id |
| department\_id | int (fk) | yes | Department id foreign key. |

* Staff Table

This table will store all the info about a staff member.

Table 10 Staff Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| staff\_id | int (pk) | yes | Unique key for the staff member. |
| created\_at | datetime timestamp | no | Staff created at time. |
| updated\_at | datetime timestamp | no | Staff profile updated time. |
| is\_deleted | bool | no | Store True if the staff is deleted, otherwise false. |
| user\_id | Int (fk) | yes | staff info from the user id |
| department\_id | int (fk) | yes | Department id foreign key. |

* Patient Table

This table will store all the info about a patient, each patient may have 1 or more billing record and only 1 medical record stored in the medical center db.

Table 11 Patient Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| patient\_id | int (pk) | yes | Unique key for the patient. |
| medical\_record\_id | int (fk) | yes | Medical record id. |
| billing\_record\_id | int (fk) | yes | Bill record id. |
| created\_at | datetime timestamp | no | Patient created at time. |
| updated\_at | datetime timestamp | no | Patient profile updated time. |
| is\_deleted | bool | no | Store True if the patient is deleted, otherwise false. |
| user\_id | Int (fk) | yes | Patient info from the user id |
| center\_id | int (fk) | yes | center id foreign key. |

* Medical History Record Table

This table will store all the info about medical history record related for a particular patient, each patient may have 1 medical record stored in the medical center db.

Table 12 Medical History Record Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Data type | Required | Description |
| medical\_record\_id | int (pk) | yes | Unique key for the patient. |
| height | Decimal |  |  |
| weight | Decimal |  |  |
| medical\_problems | text |  |  |
| medical\_problemsAr | text |  |  |
| mental\_health\_issues | text |  |  |
| mental\_health\_issuesAr | text |  |  |
| medicines | text |  |  |
| medicinesAr | text |  |  |
| allergic\_to\_medicine | text |  |  |
| allergic\_to\_medicineAr | text |  |  |
| surgeries\_history | text |  |  |
| surgeries\_historyAr | text |  |  |
| immunization | text |  |  |
| immunizationAr | text |  |  |
| diagnosis | text |  |  |
| diagnosisAr | text |  |  |
| symptoms | text |  |  |
| symptomsAr | text |  |  |
| tests\_performed | text |  |  |
| tests\_performedAr | text |  |  |
| treatment\_plan | text |  |  |
| treatment\_planAr | text |  |  |
| Family\_medical\_history | text |  |  |
| family\_medical\_historyAr | text |  |  |
| prescription\_id | int (fk) |  |  |
| created\_at | datetime timestamp | no | Patient created at time. |
| updated\_at | datetime timestamp | no | Patient profile updated time. |
| is\_deleted | bool | no | Store True if the patient is deleted, otherwise false. |
| patient\_id | Int (fk) | yes | Patient info from the user id |