April 29, 2023

PATIENT TRACKING SYSTEM

1 Introduction

The purpose of this document is to define the requirements for the Patient Tracking System. The Patient Tracking System is a web-based application that will allow healthcare providers to track the movement of patients throughout the healthcare facility. This system will provide real-time updates on patient location, status, and care team assignments.

1.1 Purpose

The purpose of this document is to describe all the requirements of patient tacking system. A patient tracking system is a system used by healthcare providers to track the progress of their patients.It typically consists of a database and a series of software applications that are used to record and analyze information about patients.

Increasingly people as well as biopsies and associated equipment are tagged in various ways, for example with radio -frequency identification tags.

A given floor or ward may use a white board as its system to track the status of all the people being cared for; for example in an obstetrics ward, each mother in labour is listed, along with her status and the time she was last checked.

1.2 Scope

Patient tracking is a new technology in hospitals that hugely facilitate control over patient flow, treatment progress, as well as basic hospital processes such as patient acceptance and discharge.It improves the bed turnover rate at the hospital,allowing doctors to provide more people with treatment.

1.3 Problem Definition

The current patient tracking system in our healthcare facility is inefficient and does not provide accurate real-time data. This leads to delays in patient care and makes it difficult for healthcare providers to manage their workflows effectively. As a result, there is a need for a more effective patient tracking system that can provide real-time data on patient location, status, and care needs, as well as streamline communication between healthcare providers. The new system should be user-friendly, accessible from mobile devices, and integrated with existing electronic health record systems to ensure seamless data exchange. Additionally, it should comply with privacy and security regulations to protect patient data.

1.4 System Overview

The system consists of several components, including hardware devices and software applications. The hardware devices include RFID (Radio Frequency Identification) tags, barcode scanners, and mobile devices such as tablets or smartphones. The software applications include a patient tracking system, an electronic health record system, and a communication system.

The patient tracking system is the core component of the system, which tracks the movement and status of patients in real-time. It uses RFID tags or barcode labels attached to patients, which are scanned by RFID readers or barcode scanners to identify the patient and track their location within the healthcare facility. The system can also track other information such as vital signs, medication administration, and care plans.

The electronic health record system is another component of the patient tracking system that stores patient health information, including medical history, test results, and medication orders. This system can be integrated with the patient tracking system to provide healthcare providers with a complete picture of the patient's health status.

The communication system is a critical component of the patient tracking system that enables healthcare providers to communicate with each other in real-time. It can include messaging, voice communication, and video conferencing capabilities, allowing healthcare providers to collaborate effectively and coordinate patient care.

Overall, the patient tracking system is designed to improve patient care, optimize the use of hospital resources, and streamline communication between healthcare providers.

1.5 References

1. Institute of Medicine. Health IT and patient safety: building safer systems for better care . Washington, DC: National Academies Press; 2012.

2. Gandhi , TK, Kachalia A, Thomas EJ, Puopolo AL, Yoon C, Brennan TA, et al. Missed and delayed diagnoses in the ambulatory setting: a study of closedmalpractice claims. Ann Intern Med 2006;145:488–96.

3. Department of Health and Human Services. HIPAA—frequently asked questions. Available at: http://www.hhs.gov/ocr/privacy/hipaa/faq. Retrieved August 24, 2012.

2 Overall Description

One of the important functions of the hospital patient tracking systems is defining a patient's location in real-time. After coming to a hospital, the patient is provided with the bracelet with the tag that contains detailed information about him – name, identifier, current state, results of medical tests, and others.

While moving around the hospital, the readers, located on the territory, detect the signals from the tags and send them to the cloud.

Later, the information gets into the computer or the mobile application on the doctor's smartphone.

2.1 Product Perspective

To identify current patient identification techniques and approaches used worldwide in today's healthcare environment. To identify challenges associated with improper patient identification.

Patient identification is the process of correctly matching a patient to appropriately intended interventions and communicating information about the patient's identity accurately and reliably throughout the continuum of care.

2.2 Product Functions

The indoor position system is based on beacons or tags that are installed in buildings or placed on tracked objects. Special readers receive signals from these devices and send them to the server that processes the obtained data and directs the information with the exact coordinates to the computer or mobile application. Such a solution makes it possible to build routes, send push notifications with advertising or useful tips to users.

It is possible to react quickly to incidents. If, while going downstairs, a patient suddenly falls, the platform will analyze his movements and send the corresponding notification to the nurse. Another scenario is fire.

Another scenario if fire. In this case, the hospital staff can timely locate the hot spot and assure that all the visitors have promptly left the building. The analysis of emergencies and patient tracking lets the hospital management define potentially dangerous places and prevent incidents in the future.

2.3 User Characteristics

The users of a patient tracking system can vary depending on the healthcare facility or system where the system is implemented. However, some general user characteristics of a patient tracking system may include:

1. Healthcare providers: Healthcare providers such as doctors, nurses, and other clinical staff will be the primary users of the patient tracking system. They will use the system to track patient movement, monitor vital signs, administer medications, and manage care plans.

2. Administrative staff: Administrative staff such as registration clerks, admissions coordinators, and billing staff will use the system to manage patient admission, discharge, and billing information.

3. Patients and their families: Patients and their families may use the patient tracking system to check the status of their appointments, view their medical records, and communicate with healthcare providers.

4. IT staff: IT staff will be responsible for the implementation, maintenance, and troubleshooting of the patient tracking system. They will ensure that the system is up-to-date and running smoothly.

5. Hospital management: Hospital management will use the patient tracking system to monitor hospital operations, track patient flow, and optimize the use of hospital resources.

The user characteristics of a patient tracking system may vary depending on the healthcare facility or system where the system is implemented. Therefore, it is essential to design a user-friendly system that meets the needs of all the users and provides effective communication and collaboration between them.

2.4 Constraints

Constraints for a patient tracking system may include technical, financial, regulatory, and organizational constraints. Here are some examples:

1. Technical constraints: Technical constraints can include the availability of compatible hardware, software, and network infrastructure to support the patient tracking system. This may also include the need for data security and privacy measures to comply with industry regulations.

2. Financial constraints: Financial constraints can include budget limitations, cost-effectiveness, and the need to balance costs with the benefits of the system. This may also include the cost of hardware, software, and ongoing maintenance and support.

3. Regulatory constraints: Regulatory constraints can include compliance with laws and regulations related to healthcare information management, data security, and privacy. The system must also adhere to standards and regulations for interoperability and data exchange with other healthcare providers.

4. Organizational constraints: Organizational constraints can include the need to integrate the patient tracking system with existing systems, processes, and workflows. The system must also be designed to meet the needs of different departments and user groups within the organization.

Overall, it is essential to consider these constraints during the development and implementation of a patient tracking system to ensure that the system meets the needs of the healthcare facility or system and its users while complying with industry regulations and standards.

2.5 Assumption

Assumptions for a patient tracking system may include:

1. The patient tracking system assumes that all patients will wear or carry an RFID tag or barcode label that can be scanned to track their location and care needs.
2. The system assumes that the hardware and software components are reliable, and any technical issues or failures will be promptly addressed to minimize downtime.
3. The system assumes that the users of the system will have the necessary skills and training to use the system effectively, and any training gaps will be addressed promptly.
4. The system assumes that the communication system will work effectively to enable real-time communication and collaboration between healthcare providers.
5. The system assumes that the patient tracking data will be accurate and up-to-date to enable effective decision-making by healthcare providers.
6. The system assumes that the patient tracking system will improve patient care and optimize the use of hospital resources, resulting in improved patient outcomes and increased efficiency.

Overall, these assumptions are necessary to ensure that the patient tracking system functions effectively and provides the intended benefits to the healthcare facility or system and its users. However, it is important to monitor these assumptions regularly and make necessary adjustments as needed to ensure the system's continued effectiveness.

2.6 Dependencies

Dependencies for a patient tracking system may include:

1. Hardware and software infrastructure: The patient tracking system is dependent on the availability and compatibility of hardware and software components, such as RFID tags, barcode scanners, servers, and databases.
2. Data management: The patient tracking system is dependent on the quality and availability of patient data, such as medical records, care plans, and medication orders, to enable effective patient tracking and care management.
3. Communication systems: The patient tracking system is dependent on communication systems, such as messaging and voice communication, to enable real-time communication and collaboration between healthcare providers.
4. Regulatory compliance: The patient tracking system is dependent on compliance with regulatory requirements related to healthcare data management, privacy, and security.
5. Staffing and training: The patient tracking system is dependent on adequate staffing and training of healthcare providers to use the system effectively and ensure accurate and timely patient tracking and care management.
6. Organizational support: The patient tracking system is dependent on organizational support from hospital management and IT staff to ensure that the system is integrated with existing workflows and processes and that ongoing maintenance and support are provided.

Overall, the patient tracking system is dependent on a range of factors, including infrastructure, data management, communication systems, regulatory compliance, staffing and training, and organizational support, to function effectively and provide the intended benefits to the healthcare facility or system and its users.

3 Specific Requirements

3.1 External Interfaces

External interfaces for a patient tracking system may include:

1. Electronic Health Record (EHR) systems: The patient tracking system may interface with the EHR system to access patient data, such as medical history, allergies, and current medications, to enable accurate patient tracking and care management.

2. Laboratory Information Management Systems (LIMS): The patient tracking system may interface with LIMS to enable tracking of lab orders and results and ensure that the results are incorporated into the patient's care plan.

3. Pharmacy Information Systems (PIS): The patient tracking system may interface with PIS to enable tracking of medication orders and administration and ensure that the patient receives the correct medication at the right time.

4. Radiology Information Systems (RIS): The patient tracking system may interface with RIS to enable tracking of radiology orders and results and ensure that the results are incorporated into the patient's care plan.

5. Health Information Exchange (HIE): The patient tracking system may interface with HIE to exchange patient data with other healthcare providers and enable continuity of care.

6. Billing and Claims systems: The patient tracking system may interface with billing and claims systems to enable accurate billing and claims processing and ensure that the patient is charged correctly for the services received.

Overall, the patient tracking system's external interfaces are critical to enable effective communication and collaboration between different healthcare systems and providers and ensure that patients receive timely and appropriate care.

3.2 Functions

The following non-functional requirements should be met by the system-

Here are some functions that could be included in a patient tracking system:

1. Registration: The system should allow patients to register and provide their personal information such as name, age, gender, contact information, medical history, and insurance information.

2. Appointment Scheduling: Patients should be able to schedule their appointments with the healthcare provider of their choice through the system. The system should also allow healthcare providers to schedule appointments for patients.

3. Patient Check-In: The system should allow patients to check-in for their appointments using the system, either at the healthcare facility or remotely.

4. Patient Tracking: The system should be able to track patient progress and their medical history over time, including appointments attended, medications prescribed, and test results.

5. Alerts and Notifications: The system should send alerts and notifications to healthcare providers and patients about upcoming appointments, medication schedules, test results, and other relevant information.

3.3 Performance Requirements

Performance requirements for a patient tracking system may vary depending on the specific needs and goals of the system, but here are some general performance requirements that should be considered:

1. Availability: The system should be available 24/7 to ensure that healthcare providers and patients can access patient information and scheduling information at any time.

2. Response Time: The system should have fast response times to ensure that healthcare providers and patients can access patient information quickly and efficiently.
3. Scalability: The system should be able to handle increasing numbers of patients, appointments, and data as the system grows.
4. Reliability: The system should be reliable and perform consistently without crashing or losing data.

3.4 Logical database Requirements

1. Healthcare databases are systems into which healthcare providers routinely enter clinical and laboratory data.
2. One of the most commonly used forms of healthcare databases are electronic health records (EHRs).
3. Practitioners enter routine clinical and laboratory data into EHRs during usual practice as a record of the patient's care.
4. Other healthcare databases include claims databases, which are maintained by payers for reimbursement purposes, pharmacist databases (see Pharmacy and Health Insurance Databases) and patient registries (see Patient Registries).
5. Healthcare databases can be used as data sources for the generation of real-world evidence (RWE).

3.5 Design Constraints

1. Theory of constraints is an approach to identifying the most important limiting factor (ie constraint) that prevents any system or process from meeting its goal and then systematically improving that constraint until it is no longer the limiting factor. Within healthcare systems these constraints may show up as bottlenecks within the process.
2. While the bottleneck is evidence of a constraint, the constraint is usually related to equipment, staff or a policy which is stopping the process from functioning effectively.
3. The link between constraints and bottlenecks is particularly important as a bottleneck determines the pace at which the whole process can work.Just as the strength of a chain is determined by its weakest link, the limiting step and its constraint determine the work rate (throughput) of a team, process or hospital. Knowing where the constraints are enables you to focus your improvement efforts and employ specific techniques to increase and maintain throughput.

3.6 Key Features

1. A patient tracking system is really exactly as it sounds. It is a system that is put in place to monitor patient movements throughout their time in the hospital. Over the years there have been many different ways that clinicians tracked their patients, including pen and paper spreadsheets, and now with the use of RFID technology and the Internet of Things (IoT).
2. The process of tracking patients has evolved immensely and for good reason. Spreadsheets that contain large volumes of information can quickly get convoluted when shared with multiple staff members and leaves too much room for error.

4 Validation Check

4.1 Validity Check

1. The paper validates the integration of a generic real-time wireless telemedicine system utilising Global System for Mobile Communications (GSM), BLUETOOTH protocol and General Packet Radio Service (GPRS) for cellular network in clinical practice.

2. In the first experiment, the system was tested on 24 pacemaker patients at Aalborg Hospital (Denmark), in order to see if the pacemaker implant would be affected by the system. I the second experiment, the system was tested on 15 non risky arrhythmia heart patients, in order to evaluate and validate the system application in clinical practice, for patient monitoring.

3. Electrocardiograms were selected as the continuously monitored parameter in the present study. The results showed that the system had no negative effects on the pacemaker implants.

4.2 Consistency Check

The initial consistency check determines whether:

- The release number in an SAP kernel matches the release number stored in the database system
- The character set specified in the SAP kernel matches the character set specified in the database system
- Critical structure definitions that are defined in both the data dictionary and the SAP kernel are identical. The structures to be checked include SYST, T100, TSTC, TDCT and TFDIR.

4.3 Realism Check

When the patient has complied with the prerequisites outlined above, the pharmacy service is requested to prepare the medicine. However, in routine clinical practice, there are specific situations which may compromise the care process. The principal problems detected that can occur are the following:

1. While the medication has been received by the day hospital, the patient's whereabouts in the hospital are unknown.
2. The medication has gone missing during its transport from the pharmacy to the Day Hospital.
3. Neither the nursing staff or the patient has been notified of the arrival of the medicine.

4.4 Verifiability Check

1. The verification process evaluates the capture and transference of a sensor-generated signal into collected data. Verification demonstrates that a sensor technology meets a set of design specifications, ensuring that (A) the sensors it contains are capturing analog data appropriately, and (B) the firmware that modifies the captured data are generating appropriate output data.
2. In lay terms, the process of verification protects against the risk of 'garbage in, garbage out' when making digital measurements of behavioral or physiologic functions. BioMeTs include sensors that sample a physical construct; for example, acceleration, voltage, capacitance, or light.
3. Verification is a bench evaluation that demonstrates that sensor technologies are capturing data with a minimum defined accuracy and precision when compared against a ground-truth reference standard, consistently over time (intra-sensor comparison) and uniformly across multiple sensors (inter-sensor comparison).

4.5 Completeness Check

1. The development of a system of mathematical equations to express the completeness of both individual patient records and database/subgroup patient records.
2. The development of a tool that assesses the data completeness of patient records based on the aforementioned framework.
3. The development of a framework to understand problems regarding patient data completeness, accuracy, and interrelatedness that provides a foundation for future work in the area.

5 Validation Techniques

5.1 Requirement Reviews

1. A prototyping approach was used to determine the essential system requirements of a computerised patient record information system for a typical township primary health care clinic.

2. A pilot clinic was identified and the existing manual system and business processes in this clinic was studied intensively before the first prototype was implemented.

3. Interviews with users, incidental observations and analysis of actual data entered were used as primary techniques to refine the prototype system iteratively until a system with an acceptable data set and adequate functionalities were in place.

4. Several non-functional and user-related requirements were also discovered during the prototyping period.

5.2 Prototyping Techniques

1. As an example application of the proposed process, this section presents the development of the graphical user interface and the behavioral model of the SC 7000 patient monitor [19].

2. In this particular case, the purpose is making a virtual prototype that can be used both to validate an existing product and to train operators [20].

3. As usual in real practice, detailed product specifications are not available, so the model has been built on information garnered from the operating manual and experimentation on the real device.

4. In the development of a new device,the model would be based on written specifications and interaction with prospective users.

5.3 Testcase Genaration

1. The developer may think that the behavior of the callee function is understood completely. The developer may test interoperation between the callee function and the caller function with scenarios of the callee function behavior that the developer understands.

2. The testing approach above cannot strictly test complicated medical systems because the function behavior used by other functions is ignored and handled as a black-box.

3. A patient-controlled analgesia infusion pump (PCA infusion pump) is given as an example of a medical system. The PCA infusion pump infuses fluids into patients continuously.

4. When the patients feel a high level of pain, they trigger a bolus dose by pressing a button. The pump is programmed with a basal rate and bolus dose using a user console input device by doctors .

UML DIAGRAMS ON PATIENT TRACKING SYSTEM

UML INTRODUCTION

A software development method consists of a modeling language and a process. The Unified Modeling Language (UML) is called a modeling language, not a method. The modeling language is the notation that methods use to express designs. The process describes the steps taken in doing a design. The Unified Modeling Language (UML) is developed as a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive.

Types of UML Diagrams

So what are the different UML diagram types? There are two main categories; structure diagrams and behavioral diagrams. Click on the links to learn more about a specific diagram type.

Structure Diagrams

- Class Diagram
- Component Diagram
- Deployment Diagram
- Object Diagram
- Package Diagram
- Profile Diagram
- Composite Structure Diagram

Behavioral Diagrams

- Use Case Diagram
- Activity Diagram
- State Machine Diagram
- Sequence Diagram
- Communication Diagram
- Interaction Overview Diagram
- Timing Diagram

1.CLASS DIAGRAM

AIM: To draw a Class Diagram for Patient Tracking System

DESCRIPTION:

A class diagram for patient tracking system would represent the various classes, their attributes, and their relationships with each other. Here is a brief description of the class diagram:

1. Classes
2. Relationships

A class diagram for patient tracking is a UML diagram that represents the classes, interfaces, and relationships involved in tracking patients within a healthcare system. The diagram provides a visual representation of the different components and their interactions in the system.

The main classes in a patient tracking system would typically include a Patient class, which would contain attributes such as name, date of birth, and medical record number. Other important classes might include a Doctor class, a Nurse class, and a Receptionist class, each with their own attributes and methods.

The relationships between these classes might include an association between the Patient class and the Doctor class, indicating that a patient is assigned to a particular doctor for treatment. There might also be an association between the Patient class and the Nurse class, indicating that a patient is assigned to a particular nurse for care. Additionally, there might be an association between the Patient class and the Receptionist class, indicating that a patient checks in with a receptionist upon arrival at the healthcare facility.

In addition to these basic classes and relationships, a class diagram for patient tracking might also include other elements such as interfaces, abstract classes, and packages. For example, there might be an interface for the scheduling system, which would allow doctors and nurses to schedule appointments with patients. There might also be an abstract class for medical procedures, which would provide a base class for specific procedures such as surgeries or diagnostic tests. In addition to the basic classes and relationships, a class diagram for patient tracking might also include additional classes for managing patient data and tracking patient progress. For example, there might be a class for managing patient medical history, a class for tracking patient vital signs, or a class for managing patient medications.

The diagram might also include additional relationships between classes, such as an aggregation relationship between the Patient class and the Medical History class, indicating that a patient's medical history is made up of multiple pieces of information.

Visibility of the classes' attributes and methods can also be represented in the diagram using symbols such as "+" for public, "-" for private, and "#" for protected. For example, the Patient class might have public attributes such as name and date of birth, but private attributes such as social security number.

The class diagram for patient tracking is an important tool for healthcare professionals and software developers, as it provides a clear and concise representation of the different components and interactions involved in the system. It can be used in the early stages of software development to help design the system and ensure that it meets the needs of healthcare professionals and patients, and it can also be used later in the development process to generate code automatically or to verify that the code conforms to the design.

Overall, a class diagram for patient tracking provides a clear and concise representation of the classes and relationships involved in tracking patients within a healthcare system. It helps developers and healthcare professionals to understand the system more clearly, and to identify any potential issues or areas for improvement.

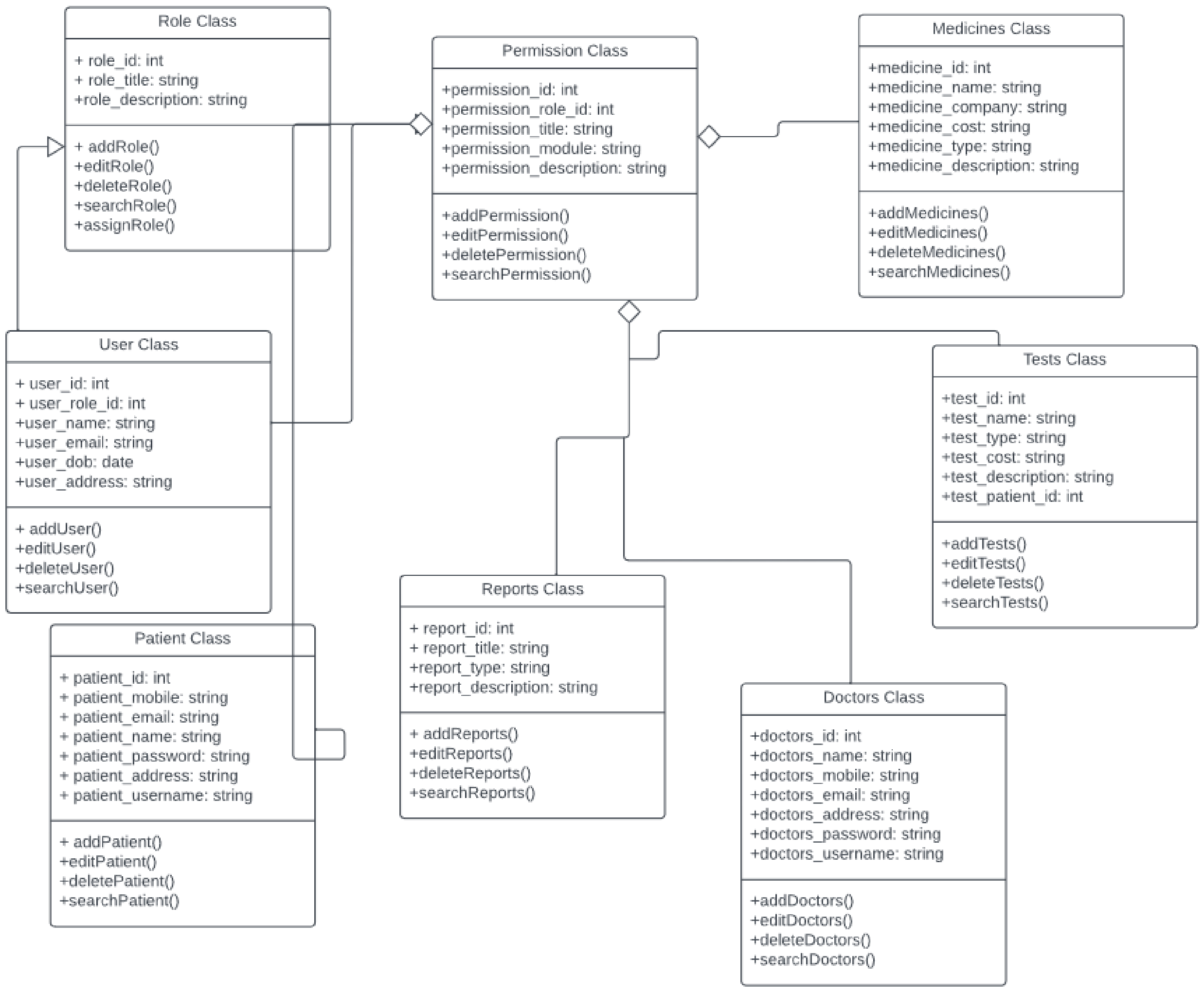


Figure 1: CLASS DIAGRAM

2.COMPONENT DIAGRAM

AIM: To draw a Component Diagram for Patient Tracking System

DESCRIPTION:

A component diagram for a patient tracking system would show the various components of the system and their dependencies. Here is a brief description of the component diagram:

1. Components
2. Dependencies

A component diagram for patient tracking is a UML diagram that represents the various components of a patient tracking system and their relationships. The diagram provides a high-level view of the system architecture, including the software components and the interfaces between them.

The main components in a patient tracking system might include a Patient Management component, a Doctor Management component, a Nurse Management component, and a Receptionist Management component. Each of these components would be responsible for managing the interactions with patients, doctors, nurses, and receptionists, respectively.

The component diagram would show the interfaces between these components, which would allow them to communicate and exchange data. For example, the Patient Management component might interface with the Doctor Management component to schedule appointments, and with the Nurse Management component to record patient vitals.

In addition to these main components, a component diagram for patient tracking might also include other components, such as a Medical Records Management component, which would be responsible for managing patient medical records, or a Reporting component, which would generate reports on patient data.

The diagram might also show the dependencies between components, which would indicate which components are required for others to function properly. For example, the Doctor Management component might depend on the Patient Management component to retrieve patient information, and the Receptionist Management component might depend on the Medical Records Management component to retrieve patient medical history.

In addition to showing the main components and their interfaces, a component diagram for patient tracking might also include other elements, such as interfaces, ports, and connectors.

Interfaces define the methods and data that a component provides or requires. For example, the Patient Management component might provide an interface for scheduling appointments, which would be used by the Doctor Management component to schedule appointments for patients.

Ports represent the points of connection between components and the external environment. For example, the Patient Management component might have a port for receiving appointment requests from patients or doctors.

Connectors represent the relationships between components and can be used to indicate the type of interaction between components. For example, a client-server connector might be used to indicate that the Doctor Management component is a server that responds to requests from the Patient Management component, which is a client.

Visibility of the components' interfaces, ports, and connectors can also be represented in the diagram using symbols such as "+" for public, "-" for private, and "#" for protected.

The component diagram for patient tracking is an important tool for healthcare professionals and software developers, as it provides a clear and concise representation of the system architecture and its components. It can be used in the early stages of software development to help design the system and ensure that it meets the needs of healthcare professionals and patients, and it can also be used later in the development process to verify that the system is implemented correctly and that it meets the requirements.

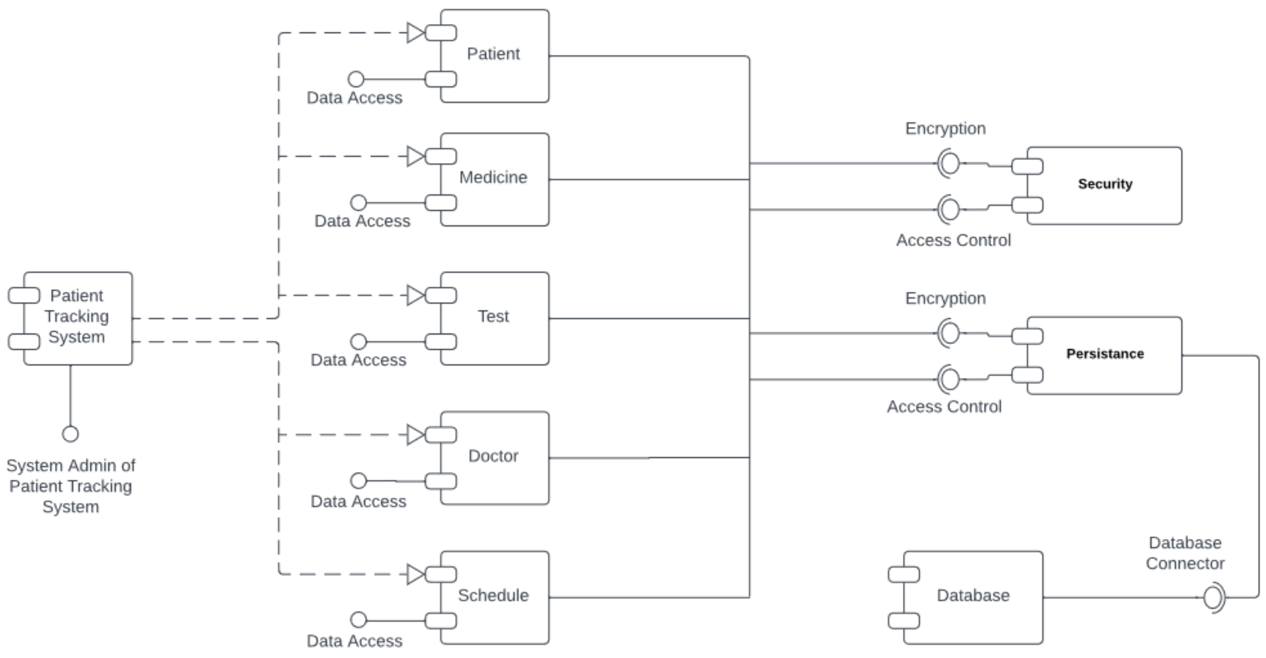


Figure 2: COMPONENT DIAGRAM

3.DEPLOYMENT DIAGRAM

AIM: To draw a Deployment Diagram for Patient Tracking System

DESCRIPTION:

A deployment diagram for a patient tracking system would show the physical deployment of the system's components on hardware and software platforms. Here is a brief description of the deployment diagram:

1. Nodes
2. Relationships

A deployment diagram for patient tracking is a UML diagram that represents the physical architecture of a patient tracking system and how the software components are deployed across different hardware devices or nodes. The diagram shows the relationships between the software components and the hardware devices or nodes on which they are deployed.

The main components in a patient tracking system might include a web server, a database server, and various client devices such as laptops, desktops, tablets, and mobile phones. The deployment diagram would show how these components are deployed across the hardware devices or nodes.

For example, the web server might be deployed on a dedicated server machine, while the database server might be deployed on a separate database server machine. The client devices might access the patient tracking system through a web browser or a mobile app, which would communicate with the web server over the internet.

The diagram would also show the connections between the different hardware devices or nodes, indicating how data is transferred between them. For example, the web server might communicate with the database server over a local area network (LAN), while the client devices might communicate with the web server over the internet.

In addition to showing the main components and their deployment, a deployment diagram for patient tracking might also include other elements, such as deployment targets, artifacts, and deployment specifications.

Deployment targets represent the physical devices or nodes on which the software components are deployed. For example, a server machine might be represented as a deployment target.

Artifacts represent the files or other resources that are deployed to the deployment targets. For example, the web server might be represented as an artifact that is deployed to the server machine.

Deployment specifications define the configuration and deployment requirements for the software components. For example, the database server might require a specific version of a database management system or a certain amount of disk space.

Overall, a deployment diagram for patient tracking provides a high-level view of the physical architecture of the system and how the software components are deployed across different hardware devices or nodes. It helps developers and healthcare professionals to understand the physical infrastructure required to support the system and to identify potential issues or areas for improvement. The diagram can be used throughout the software development process to ensure that the system is designed and implemented correctly and that it meets the needs of healthcare professionals and patients.

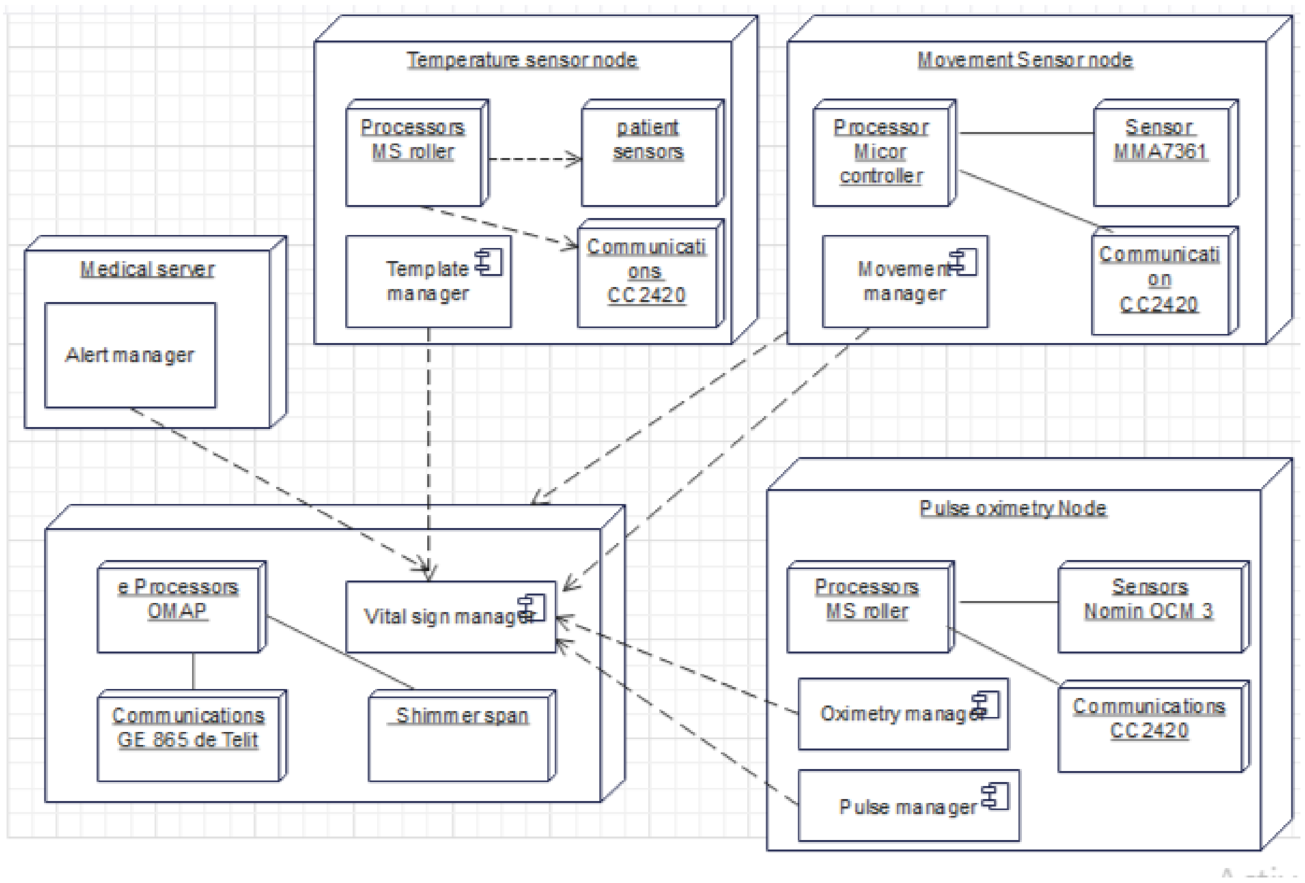


Figure 3: DEPLOYMENT DIAGRAM

4.USECASE DIAGRAM

AIM: To draw an Usecase Diagram for Patient Tracking System

DESCRIPTION:

A use case diagram for patient tracking system would show the various actors and their interactions with the system. Here is a brief description of the use case diagram:

1. Actors
2. Usecases
3. Relationships

A use case diagram for patient tracking is a UML diagram that represents the various ways in which users (actors) interact with the patient tracking system. The diagram provides a high-level view of the system's functionality, showing the different use cases and the actors who are involved in them.

The main actors in a patient tracking system might include patients, doctors, nurses, receptionists, and administrators. Each of these actors would have different roles and responsibilities within the system, and would interact with it in different ways.

The use case diagram would show the different use cases that each actor can perform within the system. For example, the Patient actor might have use cases for viewing their medical records, scheduling appointments, and receiving alerts from the system. The Doctor actor might have use cases for reviewing patient records, prescribing medication, and communicating with other healthcare professionals.

In addition to showing the actors and their use cases, a use case diagram for patient tracking might also include other elements, such as use case relationships, actors' roles, and use case descriptions.

Use case relationships indicate how different use cases are related to each other. For example, the "Schedule Appointment" use case might be related to the "View Calendar" use case, as scheduling an appointment requires checking the availability of the doctor.

Actors' roles define the specific responsibilities of each actor within the system. For example, a Nurse actor might be responsible for recording patient vitals and communicating them to the Doctor actor.

Use case descriptions provide a detailed explanation of each use case, including its inputs, outputs, and pre- and post-conditions. For example, the "View Medical Records" use case might require the patient to log in to the system, and would display the patient's medical history once they are authenticated.

Overall, a use case diagram for patient tracking provides a clear and concise representation of the system's functionality and the actors who are involved in it. It can be used in the early stages of software development to help design the system and ensure that it meets the needs of healthcare professionals and patients, and it can also be used later in the development process to verify that the system is implemented correctly and that it meets the requirements.

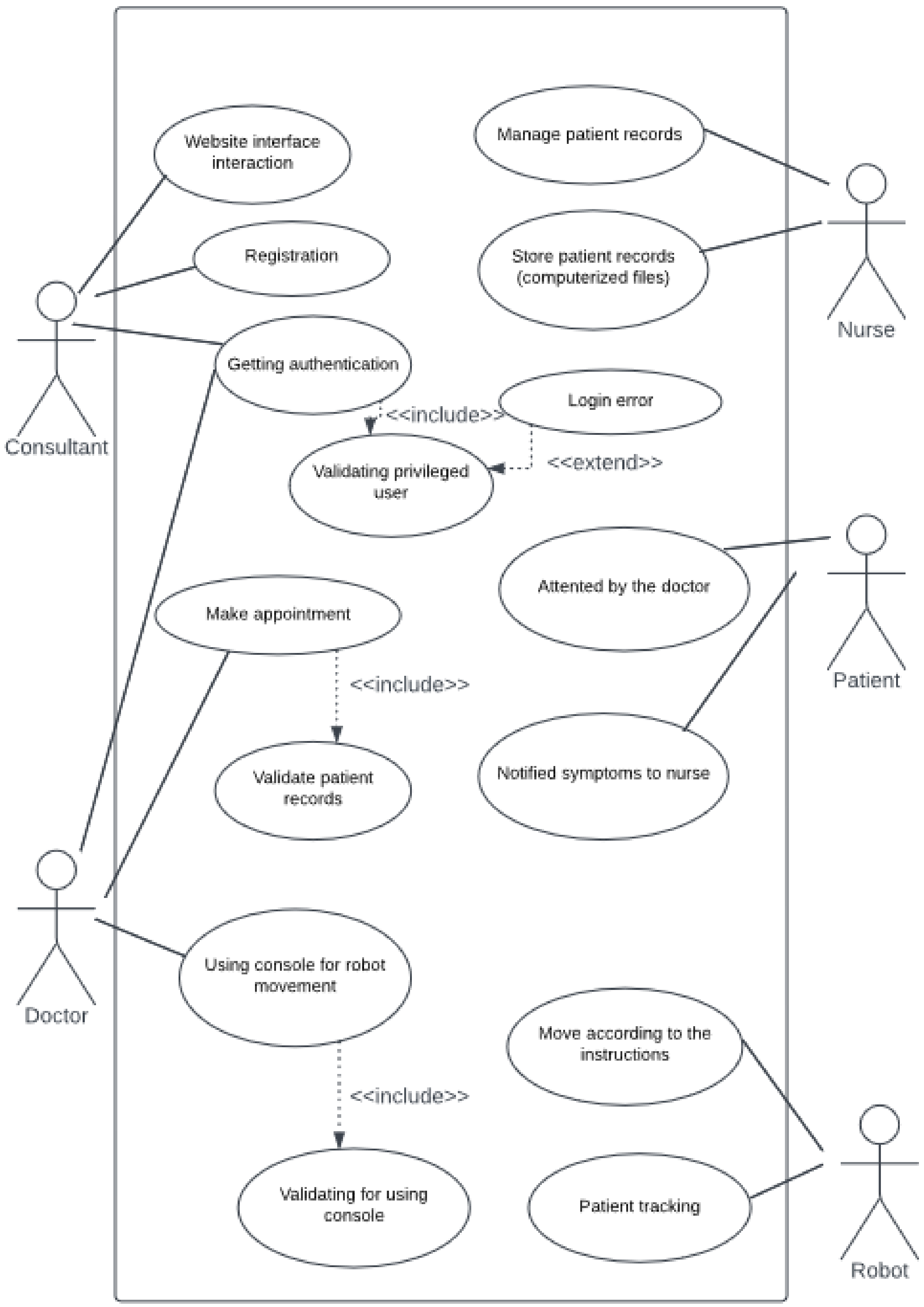


Figure 4: USECASE DIAGRAM

5.ACTIVITY DIAGRAM

AIM: To draw an Activity Diagram for Patient Tracking System

DESCRIPTION:

An activity diagram for a patient tracking system would show the flow of activities and actions performed by the system and its users to track and manage patient information. Here is a brief description of the activity diagram:

1. Activities
2. Actions

An activity diagram for patient tracking is a UML diagram that represents the flow of activities and actions involved in managing patient information and tracking their progress through the healthcare system. The diagram shows the steps involved in the patient tracking process, including the inputs and outputs of each activity and the decisions that are made along the way.

The activity diagram for patient tracking might start with the process of patient registration, where a new patient is added to the system. This might involve entering their personal information, medical history, and insurance details.

From there, the diagram would show the various activities involved in managing the patient's care, such as scheduling appointments, recording vital signs, ordering tests and medications, and communicating with other healthcare professionals.

For example, the diagram might show a branch for scheduling appointments, with different paths depending on whether the appointment is with a doctor, nurse, or other healthcare professional. The diagram might also show a loop for recording vital signs, which would repeat until all relevant data has been collected.

Throughout the diagram, decisions would be made based on the patient's condition and the medical history that is available. For example, if the patient's vital signs indicate that they are in distress, the system might automatically alert a doctor or other healthcare professional.

The activity diagram would also show the inputs and outputs of each activity, indicating the data that is being collected, processed, and stored by the system. For example, the input to the "Order Test" activity might be the type of test that is required, while the output might be the results of the test once they are available.

Overall, the activity diagram for patient tracking provides a detailed view of the patient tracking process, showing the different activities and decisions involved in managing patient information and tracking their progress through the healthcare system. It can be used to identify potential bottlenecks or areas for improvement in the patient tracking process, and to ensure that the system is designed and implemented in a way that meets the needs of healthcare professionals and patients.

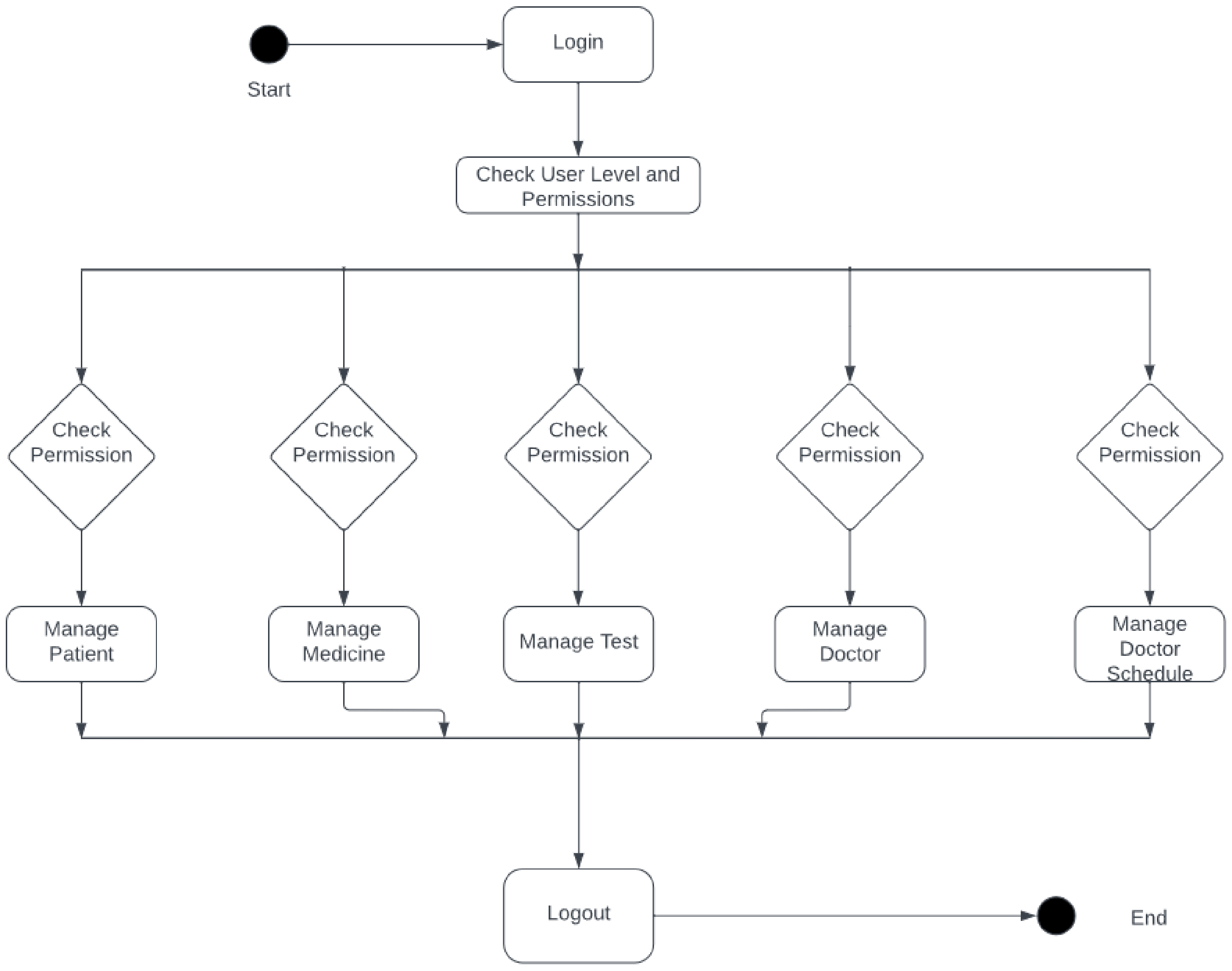


Figure 5: ACTIVITY DIAGRAM

6.STATECHART DIAGRAM

AIM: To draw a Statechart Diagram for Patient Tracking System

DESCRIPTION:

A statechart diagram for a patient tracking system would show the different states and transitions that a patient's information can go through in the system. Here is a brief description of the statechart diagram:

1. States
2. Transitions

A statechart diagram for patient tracking is a UML diagram that represents the different states that a patient can be in during their healthcare journey, as well as the events or actions that cause the patient to transition between these states. The diagram shows the lifecycle of a patient within the healthcare system, from initial registration to discharge or transfer.

The statechart diagram for patient tracking might start with the patient registration state, which is the initial state for all patients in the system. From there, the patient can transition to various other states based on their healthcare needs and progress.

For example, the diagram might show a "Wait for Appointment" state, where the patient is waiting to be seen by a healthcare professional. The patient can transition from this state to the "In Appointment" state when they are being seen by a healthcare professional.

Other states might include "In Treatment," "Awaiting Test Results," "Discharged," and "Transferred," among others. Each state would have its own set of actions and events that can cause the patient to transition to another state.

For example, in the "In Treatment" state, the patient might have a variety of treatments or procedures that need to be completed. The patient can transition from this state to the "Awaiting Test Results" state when tests have been ordered, and then transition back to the "In Treatment" state once the test results are available and the appropriate treatment plan has been determined.

The statechart diagram would also show the different types of events that can cause a patient to transition between states. These events might include things like test results becoming available, the completion of a treatment or procedure, or the decision to discharge or transfer the patient.

Overall, the statechart diagram for patient tracking provides a visual representation of the different states that a patient can be in throughout their healthcare journey, as well as the events and actions that cause them to transition between these states. It can be used to help healthcare professionals and administrators understand the patient tracking process, and to identify potential areas for improvement or optimization.

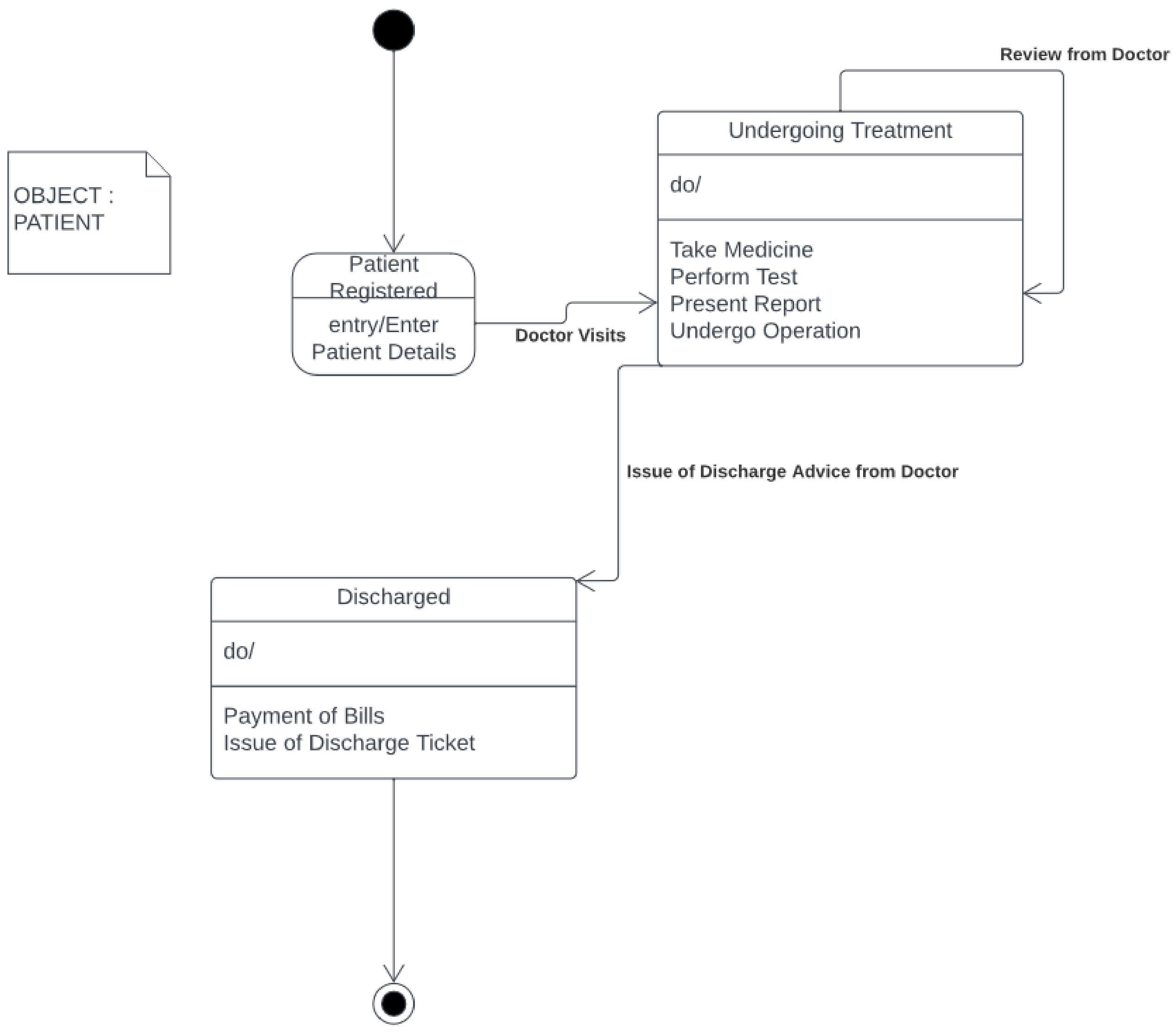


Figure 6: STATECHART DIAGRAM

7.SEQUENCE DIAGRAM

AIM: To draw a Sequence Diagram for Patient Tracking System

DESCRIPTION:

A sequence diagram for a patient tracking system would show the sequence of interactions between the system and its users to track and manage patient information. Here is a brief description of the sequence diagram:

1. Actions
2. Interactions
3. Sequence

A sequence diagram for patient tracking is a UML diagram that illustrates the interactions between different components or actors in the patient tracking system. The diagram shows the sequence of messages or requests that are exchanged between these components, as well as the responses that are received.

The sequence diagram for patient tracking might start with a patient registering in the system. The diagram would show the interaction between the patient and the registration component, with the patient entering their personal information and the registration component storing this information in the system.

From there, the sequence diagram would show the various interactions that occur between the patient and healthcare professionals, such as doctors, nurses, and laboratory technicians. For example, the diagram might show a doctor requesting that a test be performed on a patient, and the laboratory technician responding with the results of the test once it has been completed.

The sequence diagram would also show the interactions between different components within the patient tracking system. For example, the diagram might show a nurse scheduling an appointment for a patient, with the scheduling component confirming the availability of the requested time slot and sending a notification to the patient.

Throughout the diagram, each message or request would be labeled with a number to indicate the sequence in which it occurs. The diagram would also show the response time for each message or request, indicating the amount of time that elapses between sending the request and receiving the response.

Overall, the sequence diagram for patient tracking provides a detailed view of the interactions that occur within the patient tracking system, showing the sequence of messages or requests that are exchanged between different components or actors. It can be used to identify potential bottlenecks or areas for improvement in the system, and to ensure that the system is designed and implemented in a way that meets the needs of healthcare professionals and patients.

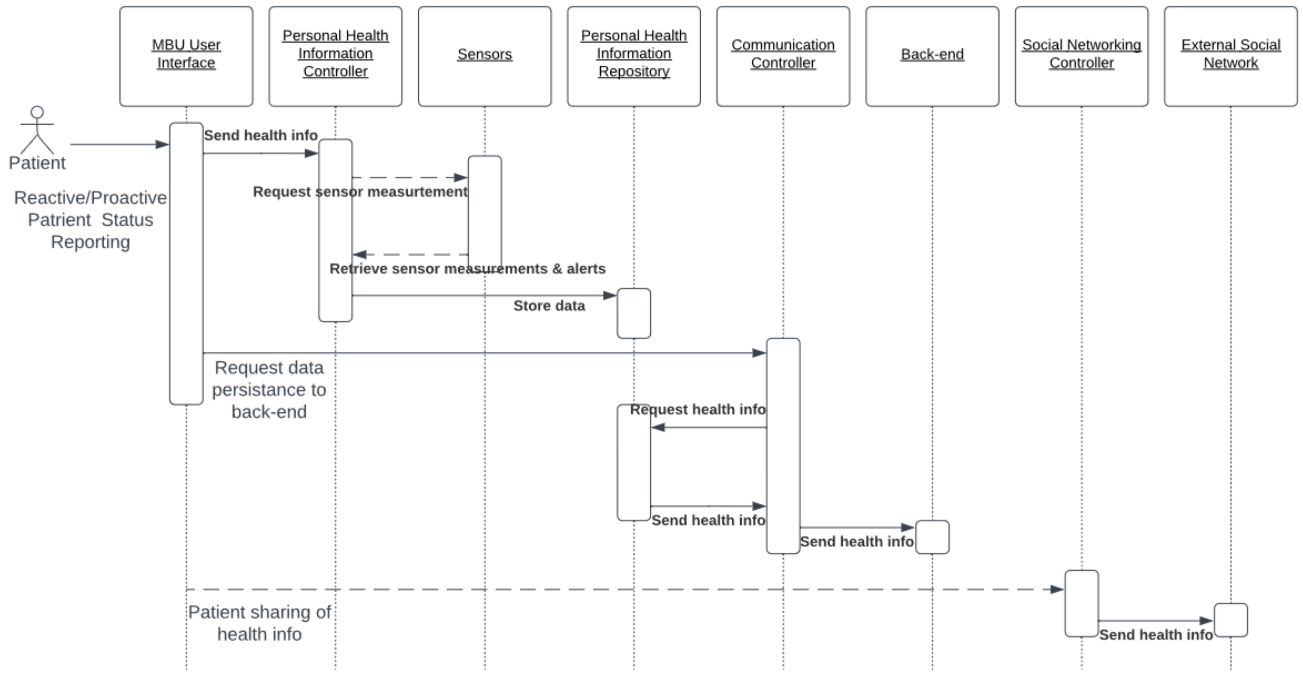

MBU User Interface
Personal Health Information Controller
Sensors
Personal Health Information Repository
Communication Controller
Back-end
Social Networking Controller
External Social Network
Patient
Reactive/Proactive Patrient Status Reporting
Send health info
Request sensor measurtement
Retrieve sensor measurements & alerts
Store data
Request data persistance to back-end
Request health info
Send health info
Send health info
Patient sharing of health info
Send health info

Figure 7: SEQUENCE DIAGRAM

8.COLLABORATION DIAGRAM

AIM: To draw a collaboration Diagram for Patient Tracking System

DESCRIPTION:

A collaboration diagram for a patient tracking system would show the interactions and collaborations between the system components and actors to manage and track patient information. Here is a brief description of the collaboration diagram:

1. Actors
2. Components
3. Interactions
4. collaboration

A collaboration diagram for patient tracking is a UML diagram that illustrates the relationships and interactions between different components or actors in the patient tracking system. Unlike a sequence diagram, which shows the sequence of messages exchanged between components, a collaboration diagram emphasizes the relationships between components and the way they work together to achieve a common goal.

The collaboration diagram for patient tracking might show the different actors involved in the patient tracking system, such as healthcare professionals, administrative staff, and patients. Each actor would be represented by a box, with lines connecting the boxes to indicate the relationships between them.

For example, the diagram might show the relationship between a patient and their primary care physician, with a line connecting the patient box to the physician box. The diagram might also show the relationships between different components within the patient tracking system, such as the registration component, the scheduling component, and the billing component.

Throughout the diagram, each component or actor would be labeled with a brief description of its role in the system. The diagram might also show the different types of interactions that occur between components, such as requests for information or requests for services.

Overall, the collaboration diagram for patient tracking provides a high-level view of the relationships and interactions between different components and actors in the patient tracking system. It can be used to help healthcare professionals and administrators understand the overall structure of the system, and to identify potential areas for improvement or optimization.

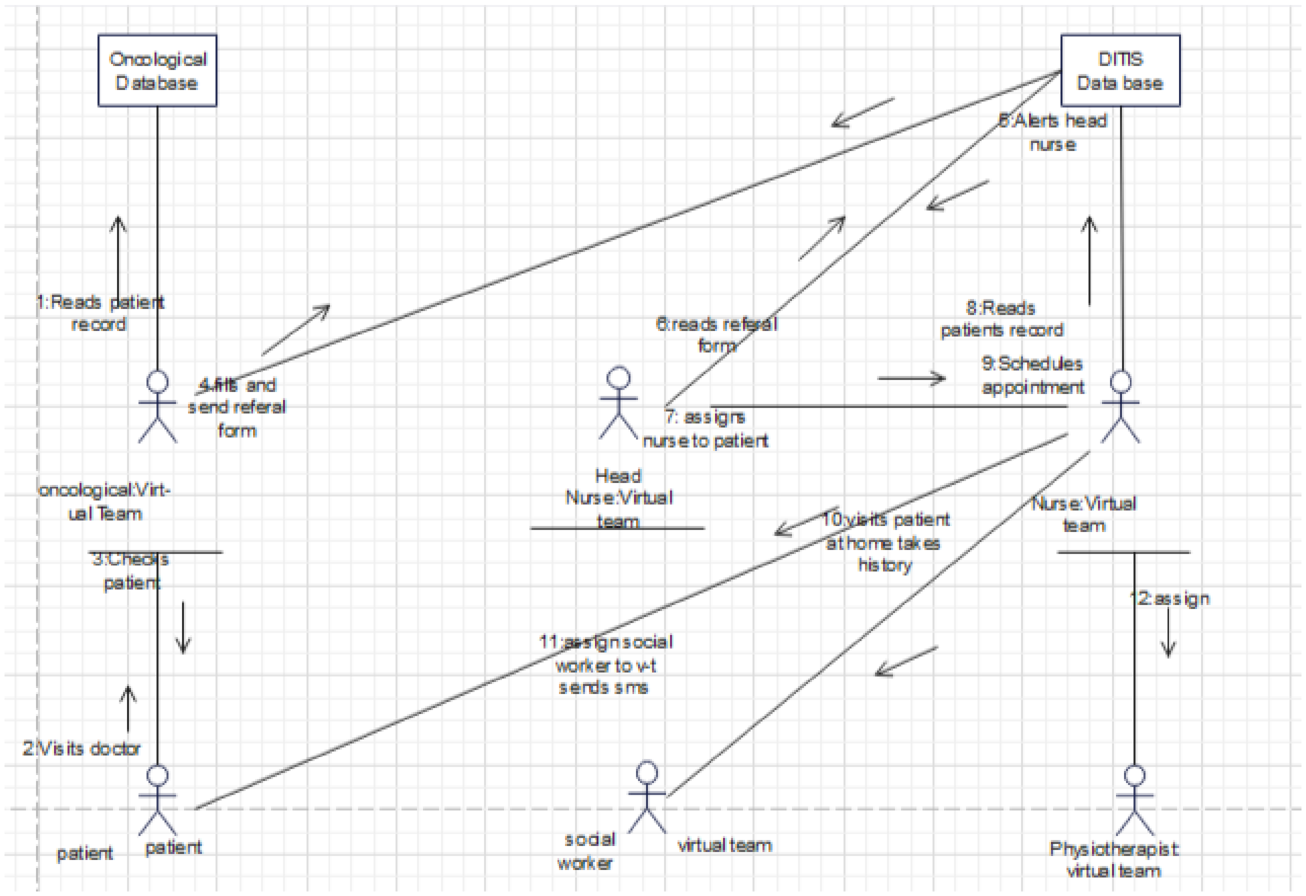


Figure 8: COLLABORATION DIAGRAM

9.TIMING DIAGRAM

AIM: To draw a Timing Diagram for Patient Tracking System

DESCRIPTION:

A timing diagram is a graphical representation of the timing of events or signals in a system. In the context of patient tracking, a timing diagram can be used to show the sequence of events that occur during the tracking process.

The timing diagram for patient tracking typically includes the following elements:

1. Start time: This is the time when the patient tracking process begins, such as when the patient arrives at the healthcare facility.
2. Check-in: This is the time when the patient checks in at the reception desk or with a nurse. This event may be represented as a vertical line or a box with the check-in time.
3. Triage: This is the time when the patient is triaged, meaning that they are assessed for the severity of their condition and prioritized accordingly. This event may also be represented as a vertical line or a box with the triage time.
4. Treatment: This is the time when the patient receives treatment for their condition. This may include medication, procedures, or other interventions. This event may be represented as a horizontal bar spanning the duration of the treatment.
5. Discharge: This is the time when the patient is discharged from the healthcare facility. This event may also be represented as a vertical line or a box with the discharge time.
6. Follow-up: This is the time when the patient returns for follow-up care, if necessary. This event may also be represented as a vertical line or a box with the follow-up time.

The timing diagram for patient tracking can help healthcare providers to identify any delays or bottlenecks in the process, and to optimize the flow of patients through the system. It can also be used to communicate the patient's progress to other members of the healthcare team.

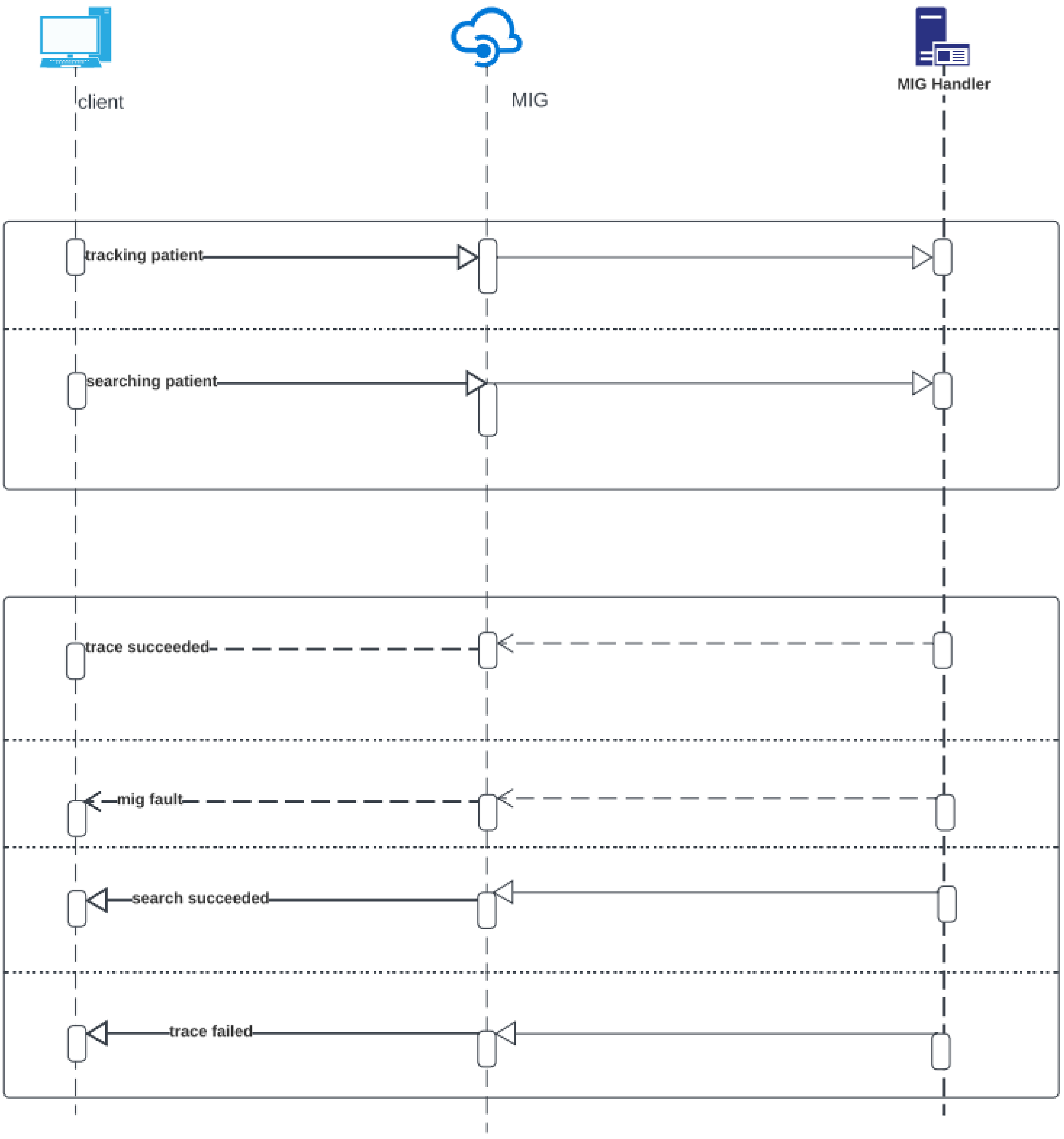


Figure 9: TIMING DIAGRAM

COCOMO MODEL:PATIENT TRACKING SYSTEM

April 16, 2023

AIM:

The COCOMO (Constructive Cost Model) is a software cost estimation model that is used to estimate the effort, cost, and schedule required to develop a software system. The aim of using the COCOMO model for the patient tracking system is to provide a reliable estimate of the resources and time required to complete the development of the system.

DESCRIPTION:

The patient tracking system is a software application that is designed to help healthcare providers track the medical history and treatment progress of patients. The system includes features such as patient registration, appointment scheduling, medical record keeping, and treatment monitoring. The system is expected to be developed using a combination of programming languages, frameworks, and libraries, and deployed on a web server or a cloud platform.

EFFORT:

The effort required to develop the patient tracking system can be estimated using the COCOMO model. The model takes into account factors such as the size of the system, the complexity of the development process, the experience and skill level of the development team, and other project-specific factors. The effort required is typically measured in person-months.

SCHEDULE:

The schedule for the patient tracking system will depend on several factors, such as the development team's size, their experience, and the project's complexity. Assuming a development team of 8 members and a schedule of 12 months, we can estimate the project's schedule as follows:
Schedule = Effort / Team Size = 95.7 / 8 = 11.96 months.

EMBEDDED:

The COCOMO model can be used for embedded software development projects as well, where the software is embedded in hardware devices or systems. The model takes into account factors such as the hardware platform, the software complexity, and the development process used for the embedded system.

TYPES OF MODELS:

The basic model uses a simple formula to estimate effort based on the size of the system. The intermediate model adds factors such as development team experience and software complexity to the basic model.There are three COCOMO types:

- Basic COCOMO model: This model estimates the effort required based on the system size.
- Intermediate COCOMO model: This model includes additional factors such as development team experience, software complexity, and development environment.
- Detailed COCOMO model: This model considers all factors that affect the effort required, including the project's characteristics, the development team's experience, and the software's complexity.

Using these factors, we can estimate the effort required using the following formula:
Effort = A x Size$\hat{B}$ x EAF

DETAILED MODEL:

To apply the detailed COCOMO model to the patient tracking system, we need to consider the following factors:

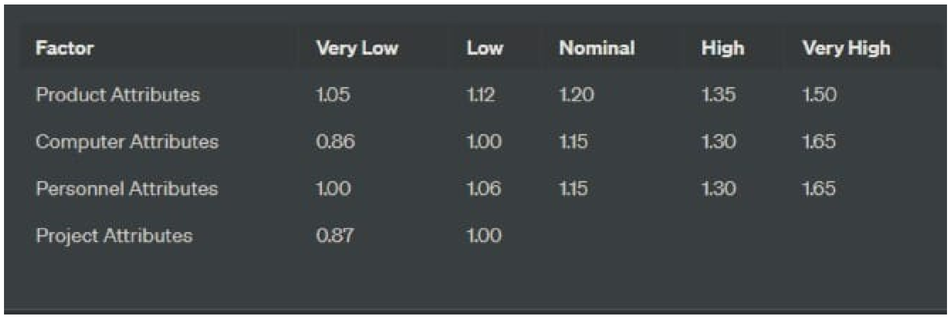
- Product attributes: the size, complexity, and reliability of the system
- Computer attributes: the processing power, memory, and storage capacity of the development environment
- Personnel attributes: the experience and capability of the development team
- Project attributes: the schedule, budget, and other project-specific factors

The Six phases of detailed COCOMO are:

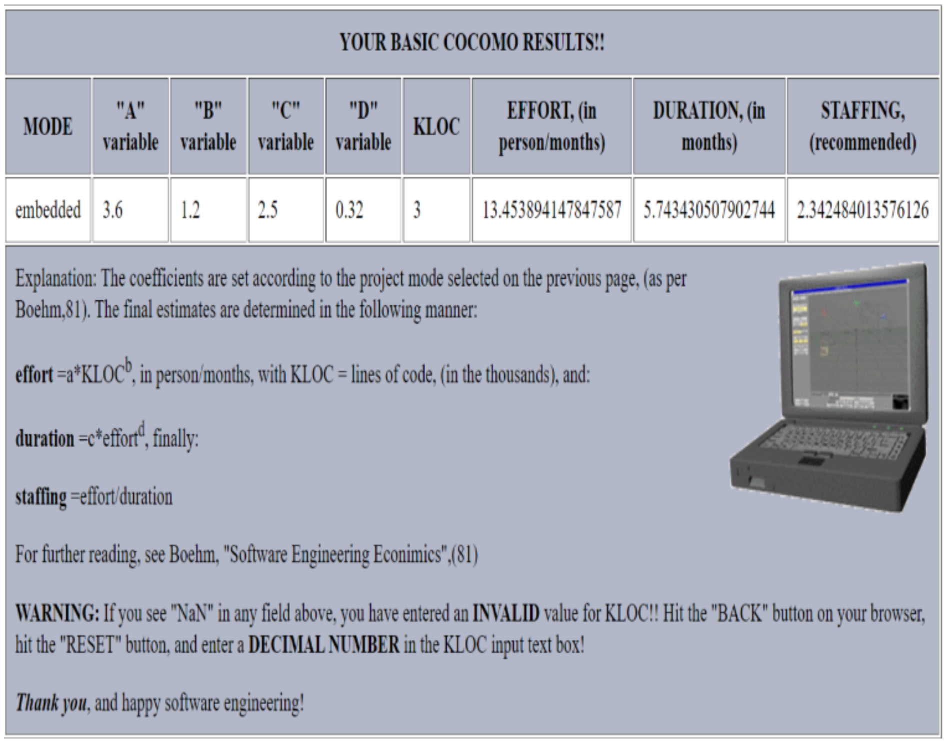
- Planning and requirements
- System structure
- Complete structure
- Module code and test
- Integration and test
- Cost Constructive model

The effort is determined as a function of program estimate, and a set of cost drivers are given according to every phase of the software lifecycle.

To calculate the EAF, we can use a table of multipliers for various project and development team factors. For example:



Factor	Very Low	Low	Nominal	High	Very High
Product Attributes	1.05	1.12	1.20	1.35	1.50
Computer Attributes	0.86	1.00	1.15	1.30	1.65
Personnel Attributes	1.00	1.06	1.15	1.30	1.65
Project Attributes	0.87	1.00			



YOUR BASIC COCOMO RESULTS!!

MODE	"A" variable	"B" variable	"C" variable	"D" variable	KLOC	EFFORT, (in person/months)	DURATION, (in months)	STAFFING, (recommended)
embedded	3.6	1.2	2.5	0.32	3	13.453894147847587	5.743430507902744	2.342484013576126

Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm,81). The final estimates are determined in the following manner:

effort =a*KLOC^{b}, in person/months, with KLOC = lines of code, (in the thousands), and:

duration =c*effort^{d}, finally:

staffing =effort/duration

For further reading, see Boehm, "Software Engineering Econimics",(81)

WARNING: If you see "NaN" in any field above, you have entered an **INVALID** value for KLOC!! Hit the "BACK" button on your browser, hit the "RESET" button, and enter a **DECIMAL NUMBER** in the KLOC input text box!

Thank you, and happy software engineering!

We can assign a score of Very Low, Low, Nominal, High, or Very High to each factor, and then multiply the corresponding multipliers to obtain the EAF. For example, if we assign a score of Nominal to all factors, the EAF will be:

EAF = 1.20 x 1.15 x 1.15 x 1.00 = 1.52

Using the values of A, B, Size, and EAF, we can calculate the effort required as follows:

Effort = A x Size$\hat{B}$ x EAF Effort = 2.94 x (20)$\hat{1}$.08 x 1.52 = 122.5 person-months

SUMMARY

the COCOMO model can be used to estimate the effort, cost, and schedule required to develop a software system such as the patient tracking system. The detailed COCOMO model is the most accurate version of the model, and takes into account a wide range of factors that can influence the development process. The schedule for the patient tracking system can be estimated based on the effort required and the availability of resources, and can be divided into phases to facilitate project management.

ENTITY RELATIONSHIP DIAGRAM ON PATIENT TRACKING SYSTEM

AIM: To Draw an Entity Relationship Diagram for Patient Tracking System

DESCRIPTION:

An entity-relationship diagram (ERD) for a patient tracking system would describe the various entities (objects) involved in the system and the relationships between them. Here is a sample ERD for a patient tracking system:

Entities:

1. Patient: a person who receives medical treatment
2. Doctor: a medical professional who provides treatment to patients
3. Nurse: a healthcare professional who assists doctors in treating patients
4. Treatment: a medical procedure or therapy given to a patient
5. Department: a unit of a hospital or clinic where medical services are provided

Relationships:

1. A patient is assigned to one or more doctors, and a doctor can have many patients (one-to-many relationship)
2. A patient can receive multiple treatments, and a treatment is given to one patient (one-to-many relationship)
3. A doctor works in one or more departments, and a department can have many doctors (many-to-many relationship)
4. A nurse works in one department, and a department can have many nurses (one-to-many relationship)
5. A treatment is provided by one or more doctors and can involve one or more nurses (many-to-many relationship)
6. Additional information can be added to the ERD as required, such as attributes for each entity and relationship, to provide a more detailed view of the patient tracking system.

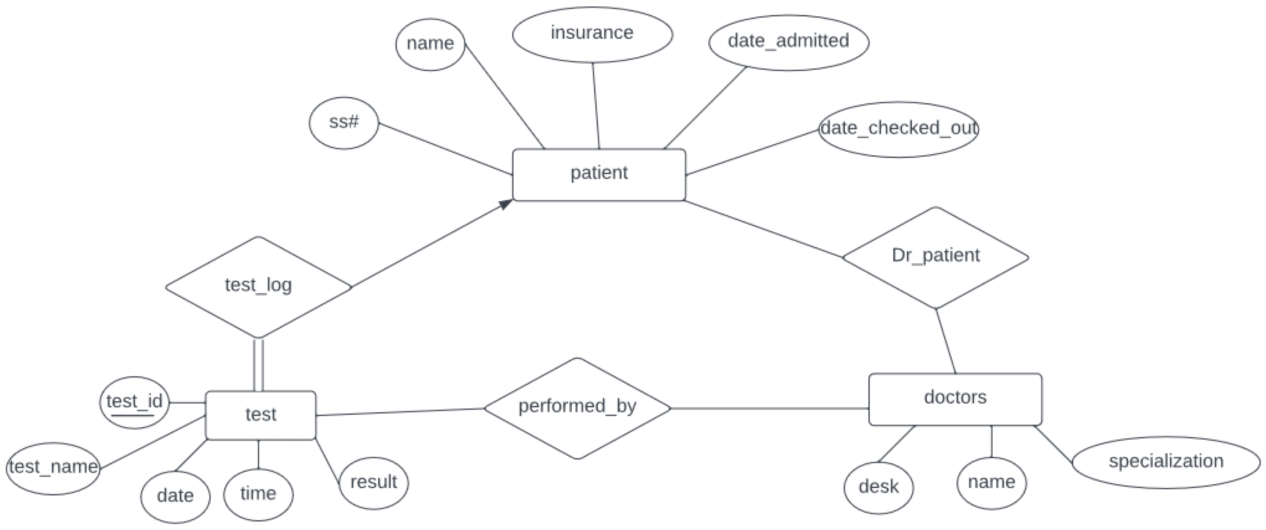


Figure 10: ENTITY RELATIONSHIP DIAGRAM

DATA FLOW DIAGRAM ON PATIENT TRACKING SYSTEM

AIM: To Draw a Data Flow Diagram for Patient Tracking System

DESCRIPTION:

A data flow diagram (DFD) for a patient tracking system would visually represent the flow of data between different components of the system. Here is a possible description for a DFD of a patient tracking system:

The system has three main components: the patient registration module, the tracking module, and the reporting module. The patient registration module allows hospital staff to enter patient information, such as name, age, gender, and medical history, into the system. This information is stored in a database.

The tracking module monitors the progress of patients through the hospital system. When a patient arrives at the hospital, their information is entered into the system and a unique identifier is assigned to them. As the patient moves through the hospital, their status is updated in the system. For example, if the patient is moved from the emergency department to the ICU, this change is recorded in the system. The tracking module also records the dates and times of key events, such as when the patient was admitted, when they had surgery, and when they were discharged.

The reporting module provides reports on patient activity within the hospital. This includes information such as the number of patients currently in the hospital, the average length of stay, and the most common diagnoses. Reports can be generated for specific time periods or for specific types of patients (e.g. pediatric patients).

The DFD would show the flow of data between these components. For example, data would flow from the patient registration module to the tracking module when a new patient is added to the system. Data would flow from the tracking module to the reporting module when reports are generated. The DFD would also show how data is stored in the system's database and how it is accessed by the different modules. Overall, the DFD would provide a high-level overview of how the patient tracking system functions and how data is used to track patient activity within the hospital.

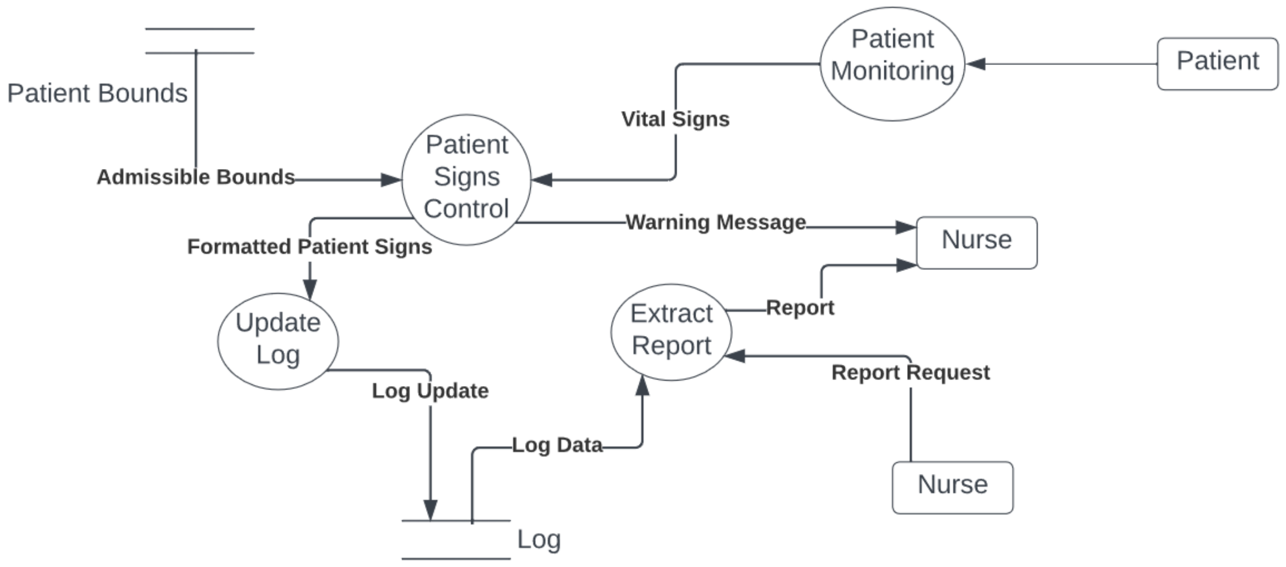


Figure 11: DATA FLOW DIAGRAM

CONTROL FLOW DIAGRAM ON PATIENT TRACKING SYSTEM

AIM: To Draw a Control Flow Diagram for Patient Tracking System

DESCRIPTION:

A control flow diagram (CFD) for a patient tracking system would visually represent the sequence of actions and decisions that occur within the system. Here is a possible description for a CFD of a patient tracking system:

The system begins when a patient arrives at the hospital and their information is entered into the system. The system then assigns the patient a unique identifier and records their status as "admitted." From this point on, the patient's progress through the hospital is tracked using the system.

As the patient moves through the hospital, their status is updated in the system. For example, if the patient is moved from the emergency department to the ICU, this change is recorded in the system. The system also records the dates and times of key events, such as when the patient was admitted, when they had surgery, and when they were discharged.

The patient's status is continuously monitored by the system, and if there are any changes in their condition or if their treatment plan needs to be updated, hospital staff can make the necessary changes in the system.

The system also includes alerts and notifications to ensure that patient care is timely and effective. For example, the system may generate an alert if a patient has been waiting for a certain amount of time without receiving care, or if a patient's vital signs indicate a potential problem.

The CFD would show the sequence of actions and decisions that occur within the system. For example, it would show how a patient's status is updated when they are moved from one department to another, or how an alert is generated when a patient's condition changes. The CFD would also show how hospital staff interact with the system to input and update patient information. Overall, the CFD would provide a high-level overview of how the patient tracking system functions and how it ensures that patients receive timely and effective care.

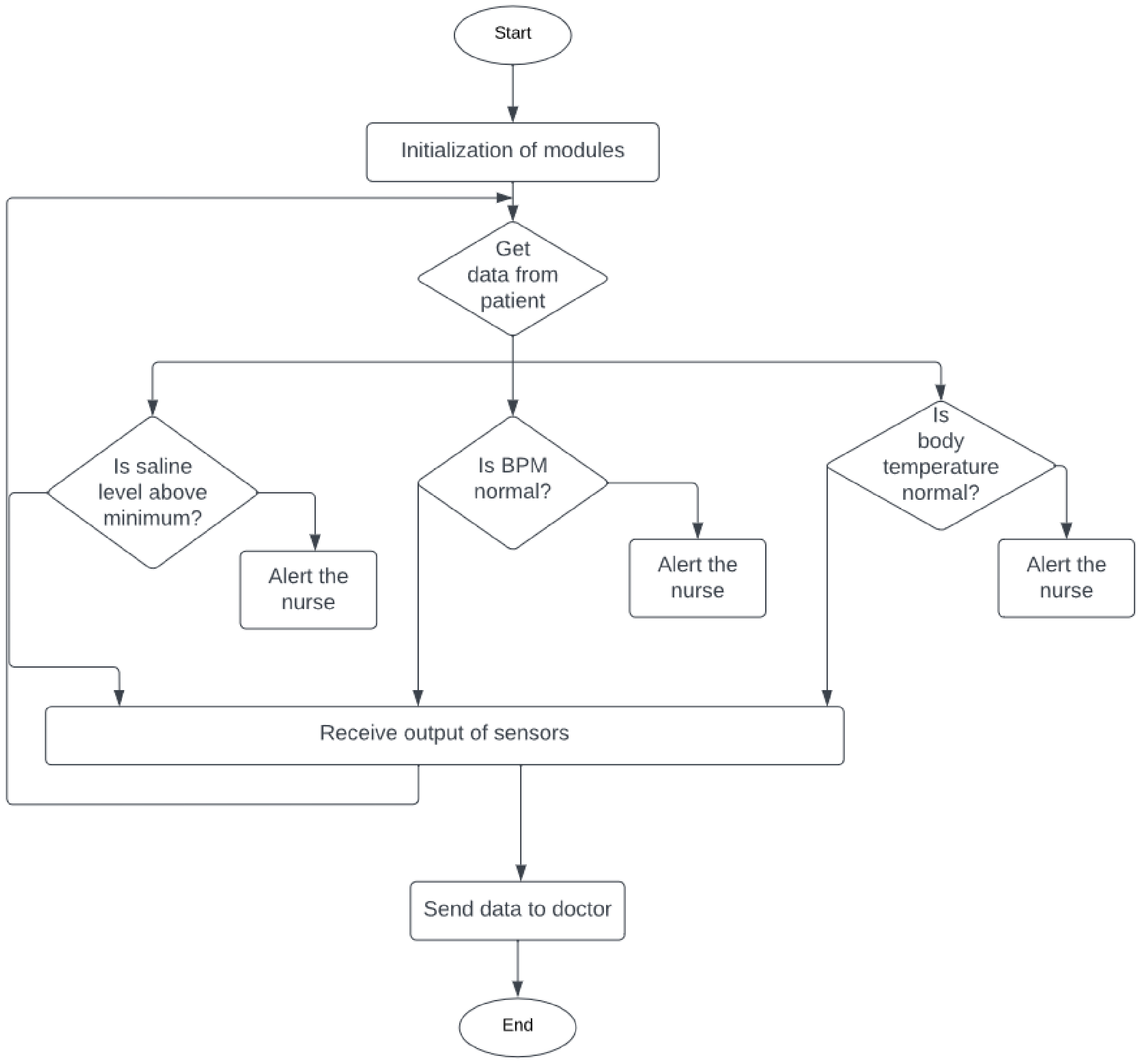

Start
Initialization of modules
Get data from patient
Is saline level above minimum?
Alert the nurse
Is BPM normal?
Alert the nurse
Is body temperature normal?
Alert the nurse
Receive output of sensors
Send data to doctor
End

Figure 12: CONTROL FLOW DIAGRAM

STRUCTURED CHART DIAGRAM ON PATIENT TRACKING SYSTEM

AIM: To Draw a Structured Chart Diagram for Patient Tracking System

DESCRIPTION:

A structured chart of a patient tracking system would typically include the following components:

1. Title: The title of the chart should clearly indicate that it represents a patient tracking system.
2. Patient Information: The chart should include columns or rows for patient information, including patient name, age, sex, medical history, and any other relevant information.
3. Admittance Information: The chart should include columns or rows for admittance information, such as date of admission, reason for admission, referring physician, and any additional notes.
4. Care Team: The chart should include columns or rows for the care team, including the primary physician, nurse, and any other healthcare professionals involved in the patient's care.
5. Diagnosis and Treatment: The chart should include columns or rows for diagnosis and treatment information, including diagnostic tests, medications prescribed, procedures performed, and any other relevant information.
6. Vital Signs: The chart should include columns or rows for the patient's vital signs, such as blood pressure, heart rate, respiratory rate, temperature, and oxygen saturation levels.
7. Progress Notes: The chart should include columns or rows for progress notes, including updates on the patient's condition, any changes in treatment or medication, and any issues or concerns that arise during the patient's stay.
8. Discharge Planning: The chart should include columns or rows for discharge planning, including the anticipated discharge date, any home health services needed, and any follow-up appointments or tests required.
9. Billing and Insurance Information: The chart should include columns or rows for billing and insurance information, including the patient's insurance coverage, any co-pays or deductibles, and any financial assistance programs available.
10. HIPAA Compliance: The chart should include columns or rows to ensure that all patient information is kept private and confidential in accordance with the Health Insurance Portability and Accountability Act (HIPAA).
11. Status Updates: The chart may also include columns or rows to track the status of the patient's insurance approval, any pending tests or procedures, and any other outstanding issues related to the patient's care.
12. Additional Information: The chart may include additional information, such as patient preferences, family contact information, and any other relevant details that may impact the patient's care.

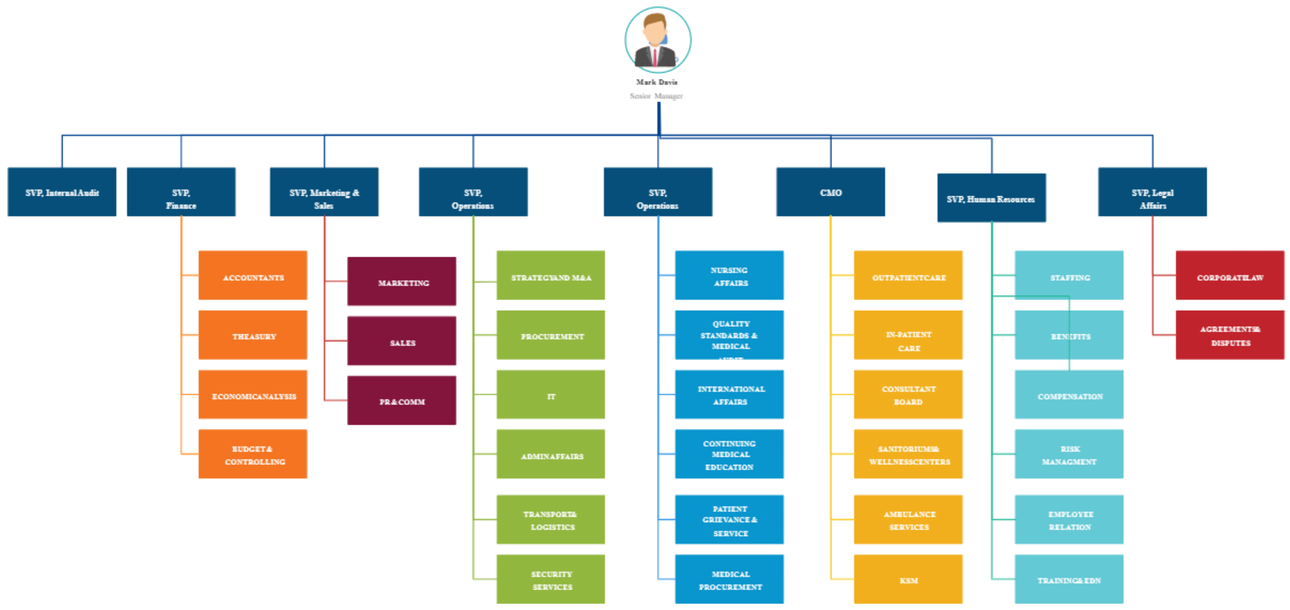


Figure 13: STRUCTURED CHART DIAGRAM

UNIT TESTING FOR PATIENT TRACKING SYSTEM

AIM: Design of Test cases based on requirements and design for Patient Tracking System using using unit testing.

DESCRIPTION:

A Patient Tracking System is a software application designed to help healthcare professionals track and manage patient information, appointments, medical conditions, and treatments. In the context of unit testing, the Patient Tracking System would be broken down into its individual functions and methods, which would be tested in isolation from the rest of the system to ensure that they work correctly and meet their design and functional specifications.

Some examples of functions and methods that might be tested in a Patient Tracking System include:

- Adding a new patient record to the system
- Updating an existing patient record with new information
- Scheduling an appointment for a patient
- Generating a report of patient appointments or medical conditions
- Verifying patient insurance information
- Checking for potential drug interactions between medications prescribed to a patient

Each of these functions and methods would be tested using unit test cases that cover all possible scenarios and test the functionality of the function or method in isolation from the rest of the system. For example, a unit test case for adding a new patient record might test that the patient's name, date of birth, and contact information are correctly stored in the system and can be retrieved later.

By testing each function and method in isolation using unit testing, the Patient Tracking System can be thoroughly tested for functionality and reliability, and any issues or defects can be identified and resolved early in the development process. This can help to ensure that the final system meets its design and functional requirements, and is reliable and easy to use for healthcare professionals who rely on it to manage patient information and care.

Objectives of Unit Testing:

The main objectives of unit testing in software development are:

1. To verify that individual functions and methods of a software application work correctly:
2. To identify defects and issues early in the development process:
3. To provide confidence in the quality of the software application:
4. To ensure that changes and updates to the software application do not cause unintended consequences
5. To facilitate code maintenance and refactoring:

When it comes to designing test cases for a patient tracking system using unit testing, the process is similar to the general steps described earlier. However, there are a few additional considerations to keep in mind:

- Understand the unit under test: When designing unit tests, it's important to understand the specific unit of code that is being tested. For example, if you're testing a registration function, you'll want to understand the inputs and outputs of that function.
- Use test-driven development: Test-driven development (TDD) is a development approach that emphasizes writing tests before writing the actual code. This approach can help to ensure that the code is tested thoroughly and that the tests cover all the necessary requirements.
- Test individual functions and methods: Unit tests should focus on testing individual functions and methods in isolation. This means that any dependencies or external systems should be mocked or stubbed out.
- Test for edge cases: Unit tests should test for edge cases, such as boundary values, invalid inputs, and unexpected scenarios. This helps to ensure that the code is robust and can handle all possible scenarios.
- Use assertions to check outputs: Unit tests should use assertions to check the outputs of the unit under test. This helps to ensure that the output is what is expected.
- Use test frameworks: Test frameworks, such as JUnit, can help to automate the testing process and make it easier to manage and run tests.
- Prioritize tests based on importance: Just as with general testing, unit tests should be prioritized based on their importance and impact on the system.
- Continuously test and refactor: As changes are made to the code, unit tests should be continuously run to ensure that the changes have not introduced any issues. Refactoring the code can also help to improve testability and maintainability.

By following these steps, you can design effective unit tests for a patient tracking system that ensure the code is reliable and meets the requirements and design specifications.

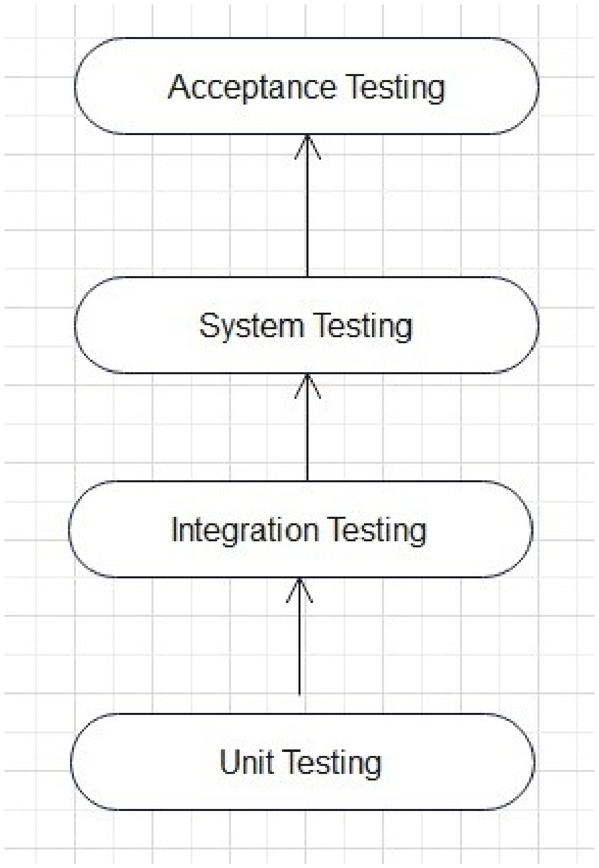

Acceptance Testing
System Testing
Integration Testing
Unit Testing

Figure 14:

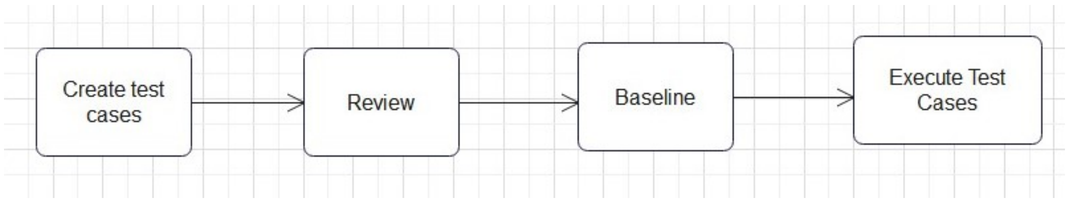


Figure 15: Workflow of Unit Testing

Types of Unit Testing:

There are 2 types of Unit Testin:

1. Manual
2. Automated

Unit Testing Techniques :

There are 3 types of unit technquies .They are:

1. **Black box testing:** This technique involves testing the functionality of a software application without any knowledge of its internal workings.

2. **White box testing:** This technique involves testing the internal workings of a software application, including its code and data structures.

3. **Gray box testing:** This technique involves testing a software application with partial knowledge of its internal workings.

Unit Testing Tools :

1. **JUnit:** JUnit is a popular open-source unit testing framework for Java applications. It provides a simple and flexible API for writing and running unit tests.

2. **NUnit:** NUnit is an open-source unit testing framework for .NET applications, similar to JUnit. It provides a wide range of assertions and is highly extensible.
3. **pytest:** pytest is a popular open-source testing framework for Python applications. It supports the testing of both functional and unit components, and is highly configurable and extensible.
4. **Mocha:** Mocha is a feature-rich JavaScript testing framework that can be used for both unit testing and functional testing. It provides a wide range of features, including test reporting and coverage analysis.
5. **xUnit.net:** xUnit.net is an open-source unit testing framework for .NET applications. It provides a simple and intuitive API for writing and running unit tests, and is highly configurable.

Advantages of Unit Testing :

1. **Catching defects early:**

   Unit testing allows developers to catch defects and errors in the software code early in the development process. This can prevent larger, more complex issues from arising later on in the development cycle, when they can be much more difficult and expensive to fix.

2. **Ensuring code quality:**

   Unit testing helps to ensure that the code being developed meets the required quality standards. By testing each individual unit of code in isolation, developers can be confident that the code is functioning correctly and meeting its intended purpose.

3. **Facilitating code maintenance:**

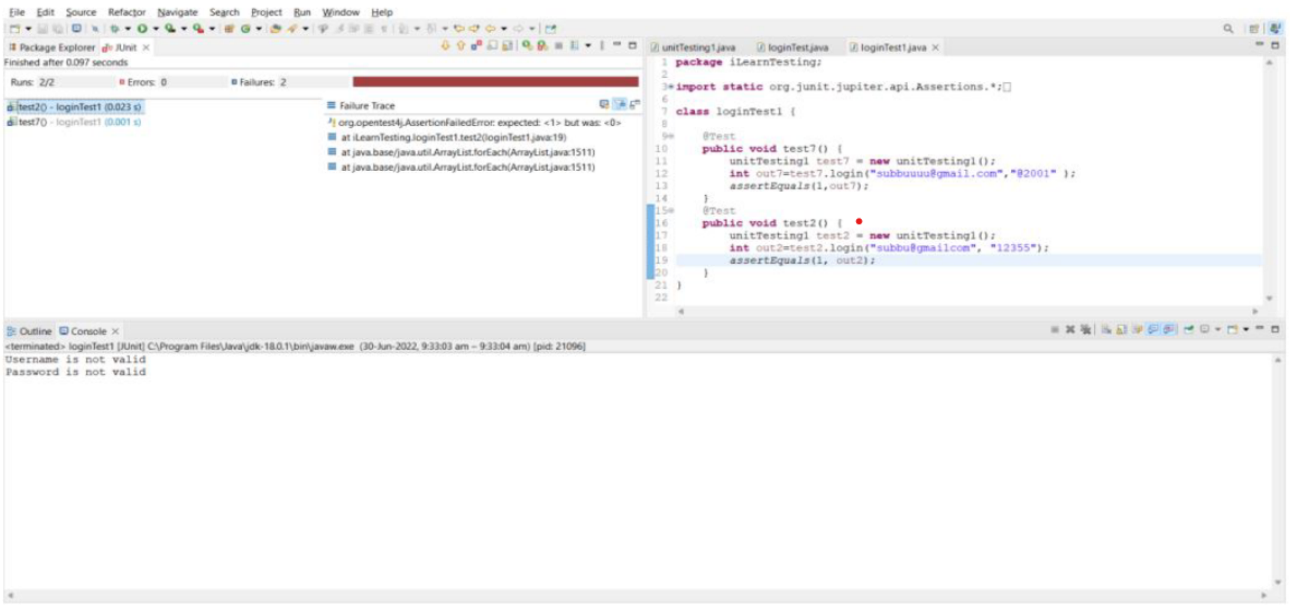
   Unit testing can make code maintenance and updates much easier. By having a suite of unit tests in place, developers can quickly and easily test any changes or updates to the code to ensure that they do not introduce new defects or issues.

Disadvantages of Unit Testing :

1. **Time-consuming:** Writing and running unit tests can be time-consuming, particularly for larger and more complex software systems. This can slow down the development process, particularly if the tests need to be updated frequently.

2. **Limited scope:** Unit tests only test individual units of code in isolation, and do not test how the various components of the system work together. This means that unit tests may not catch defects or issues that only arise when different units are combined.

3. **Limited coverage:** Unit tests can only test a small subset of the possible inputs and outputs of a system, which means that they may not uncover all potential defects or issues. Additional testing, such as integration testing or system testing, may be needed to ensure complete coverage.

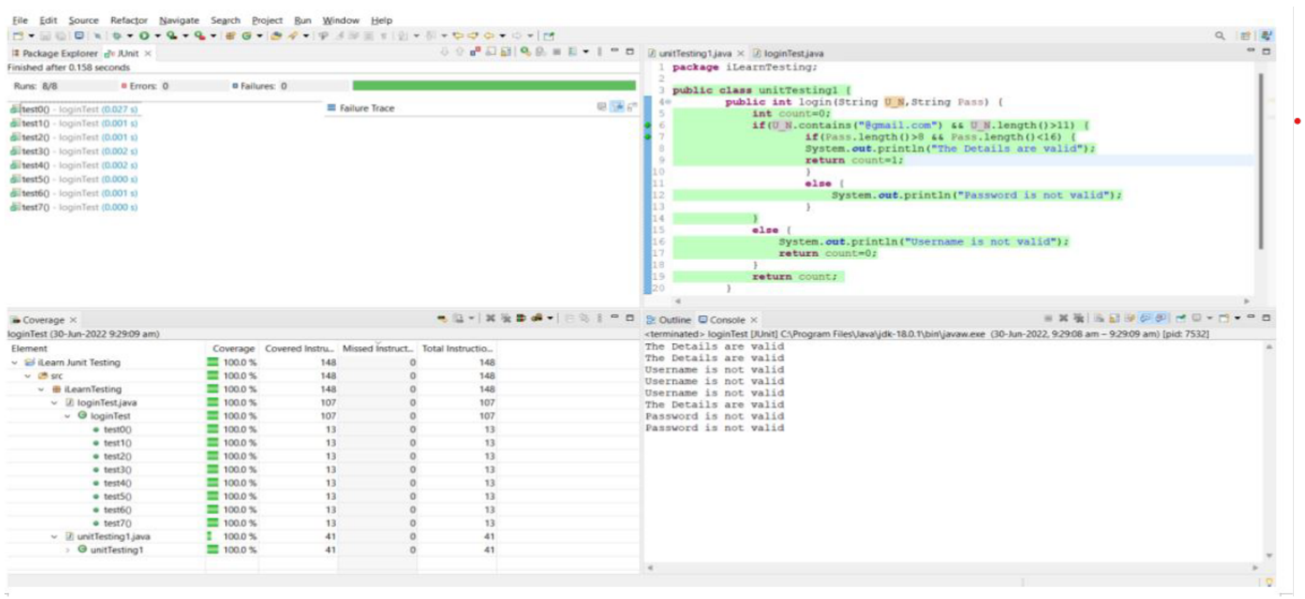


```
Finished after 0.097 seconds
Runs: 2/2   Errors: 0   Failures: 2

test2() - loginTest1 (0.023 s)
test7() - loginTest1 (0.001 s)

Failure Trace
org.opentest4j.AssertionFailedError: expected: <1> but was: <0>
at iLearnTesting.loginTest1.test2(loginTest1.java:19)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

```java
package iLearnTesting;

import static org.junit.jupiter.api.Assertions.*;

class loginTest1 {

    @Test
    public void test7() {
        unitTesting1 test7 = new unitTesting1();
        int out7=test7.login("subbuuuu@gmail.com","@2001" );
        assertEquals(1,out7);
    }
    @Test
    public void test2() {
        unitTesting1 test2 = new unitTesting1();
        int out2=test2.login("subbu@gmailcom", "12355");
        assertEquals(1, out2);
    }
}
```

```
<terminated> loginTest1 [JUnit] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe  (30-Jun-2022, 9:33:03 am – 9:33:04 am) [pid: 21096]
Username is not valid
Password is not valid
```

Figure 16:



```
Finished after 0.158 seconds
Runs: 8/8     Errors: 0     Failures: 0

test0() - loginTest (0.027 s)
test1() - loginTest (0.001 s)
test2() - loginTest (0.001 s)
test3() - loginTest (0.002 s)
test4() - loginTest (0.002 s)
test5() - loginTest (0.000 s)
test6() - loginTest (0.001 s)
test7() - loginTest (0.000 s)
```

```java
package iLearnTesting;

public class unitTesting1 {
        public int login(String U_N,String Pass) {
                int count=0;
                if(U_N.contains("@gmail.com") && U_N.length()>11) {
                        if(Pass.length()>8 && Pass.length()<16) {
                        System.out.println("The Details are valid");
                        return count=1;
                        }
                        else {
                            System.out.println("Password is not valid");
                        }
                }
                else {
                    System.out.println("Username is not valid");
                    return count=0;
                }
                return count;
        }
```

loginTest (30-Jun-2022 9:29:09 am)

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
iLearn Junit Testing	100.0 %	148	0	148
src	100.0 %	148	0	148
iLearnTesting	100.0 %	148	0	148
loginTest.java	100.0 %	107	0	107
loginTest	100.0 %	107	0	107
test0()	100.0 %	13	0	13
test1()	100.0 %	13	0	13
test2()	100.0 %	13	0	13
test3()	100.0 %	13	0	13
test4()	100.0 %	13	0	13
test5()	100.0 %	13	0	13
test6()	100.0 %	13	0	13
test7()	100.0 %	13	0	13
unitTesting1.java	100.0 %	41	0	41
unitTesting1	100.0 %	41	0	41

```
<terminated> loginTest [JUnit] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe  (30-Jun-2022, 9:29:08 am – 9:29:09 am) [pid: 7532]
The Details are valid
The Details are valid
Username is not valid
Username is not valid
Username is not valid
The Details are valid
Password is not valid
Password is not valid
```

Figure 17:

VERSION CONTROL AND CHANGE CONTROL FOR SOFTWARE CONFIGURATION ITEMS

AIM : To Prepare Version control and change control for software configuration items for Patient Tracking System

DESCRIPTION :

Software configuration management is a critical aspect of managing a patient tracking system. It involves the identification, version control, and management of all software components, including code, documentation, and related files. By implementing a robust software configuration management plan, the patient tracking system can ensure that all changes are properly controlled and tracked, and that each release is based on a stable and approved baseline. This helps to ensure the system's reliability, stability, and maintainability over time, which is crucial for providing high-quality patient care.

VERSION CONTROL :

1. **Version numbering:** The patient tracking system will use a three-digit version numbering system, where the first digit represents major changes, the second digit represents minor changes, and the third digit represents bug fixes or patches.

2. **Repository:** The system's source code will be stored in a central repository using Git as the version control system.

3. **Branching strategy:** The repository will have two main branches, a development branch for ongoing development work, and a release branch for stable releases.

4. **Tagging:** Each release will be tagged in the repository with the version number.

5. **Release notes:** Release notes will be created for each release, outlining the changes made and any known issues.

6. **Code review process:** All code changes must go through a code review process before being merged into the development branch. Code reviews will be conducted by other developers on the team to ensure code quality, maintainability, and adherence to coding standards.

7. **Continuous integration and delivery:** The patient tracking system will use a continuous integration and delivery (CI/CD) pipeline to automate the build, test, and deployment process. The CI/CD pipeline will be triggered automatically when changes are pushed to the development branch.

8. **Release schedule:** The patient tracking system will follow a release schedule based on the urgency and impact of the changes. Minor bug fixes and patches may be released immediately, while major changes may be scheduled for a specific release cycle.

CHANGE CONTROL :

1. **Change request process:** All changes to the patient tracking system must be submitted through a formal change request process.

2. **Change request form:** The change request form must include a description of the change, the impact on the system, and the justification for the change.

3. **Change approval process:** Changes must be approved by the system's designated change control board (CCB) before they can be implemented.

4. **Testing and validation:** Changes must be thoroughly tested and validated before they can be implemented in the production environment.

5. **Rollback plan:** A rollback plan must be in place in case any changes cause unexpected issues in the system.

6. **Change documentation:** All changes must be documented, including the date of the change, who made the change, and the reason for the change.

7. **Change review and prioritization:** Change requests will be reviewed by the CCB to determine their impact on the system and their priority level. Urgent changes will be prioritized over less critical changes.

8. **Change approval authority:** The CCB will have the authority to approve or reject change requests based on their impact and priority level.

9. **Change implementation process:** Changes will be implemented in a test environment first to ensure that they do not have any unintended consequences. Once the changes have been validated in the test environment, they will be implemented in the production environment during a scheduled maintenance window.

10. **Change communication:** All stakeholders, including users, administrators, and other relevant parties, will be notified of any changes to the system and their impact on the system.

11. **Change tracking and reporting:** All changes will be tracked and documented, including their status, approval, and implementation details. Regular change reports will be provided to stakeholders to ensure transparency and accountability.

This software configuration management plan, the patient tracking system can ensure that all configuration items are properly controlled and tracked, and that each release is based on a stable and approved baseline. This helps to ensure the system's reliability, stability, and maintainability over time.

By implementing this version control and change control plan, the patient tracking system can maintain a consistent and stable codebase while ensuring that changes are properly tested and approved before being implemented in the production environment.