PRACTICAL-1

AIM: Project Definition and Objective of the specified module and perform Requirement Engineering Process.

Hospital Management System (HMS) Project:

A **Hospital Management System (HMS)** is a comprehensive software solution designed to manage and automate the day-to-day operations of a hospital or healthcare facility. It helps healthcare organizations efficiently handle their administrative, financial, and clinical processes, improving both patient care and operational efficiency.

1. Definition of Hospital Management System (HMS):

A Hospital Management System (**HMS**) is a platform or suite of applications that integrates multiple hospital functions such as patient management, staff management, billing, inventory control, appointment scheduling, medical record management, and other hospital-related activities into a single unified system.

It can be implemented either as an on-premise solution or a cloud-based solution, allowing for better coordination between departments, data accuracy, and ease of access for healthcare providers.

2. Information Processed in a Hospital Management System:

The HMS handles various types of information and data flows within a hospital, including:

- **Patient Information:** Personal details, medical history, visit records, prescriptions, and discharge summaries.
- **Staff Information:** Employee details, schedules, payroll, and performance records.
- **Medical Records:** Patient medical history, diagnoses, prescriptions, lab results, and radiology reports.

ENROLLMENT NO:2303031240064

- **Financial Data:** Billing, insurance claims, patient payments, and salary management.
- **Inventory Information:** Stock management of medical equipment, drugs, and consumables.
- **Appointment Information:** Scheduling, cancellations, reminders, and appointment history.
- **Emergency and Admission Data:** Emergency response, bed management, patient admission, and discharge management.

3. Functions of a Hospital Management System

3.1. Patient Management

- Registration of new patients.
- Managing existing patient details.
- Tracking patient appointments, treatments, and progress.

3.2. Doctor/Staff Management

- Scheduling doctor shifts and appointments.
- Tracking staff performance, schedules, and availability.
- Managing payroll and benefits.

3.3. Appointment Scheduling

- Patients can book, reschedule, or cancel appointments online or through the system.
- Automated appointment reminders for patients and doctors.

3.4. Medical Record Management

- Storing and maintaining electronic health records (EHR) for all patients.
- Tracking patient history, diagnoses, treatments, and medications.
- Generating prescriptions and managing lab results.

3.5. Billing & Invoicing

- Generating bills for patients based on services provided.
- Managing insurance claims and payments.
- Generating reports for financial tracking and auditing.

ENROLLMENT NO:2303031240064

3.6. Inventory Management

- Keeping track of stock levels of medicines, medical equipment, and consumables.
- Automated stock reordering when thresholds are reached.
- Expiry management for pharmaceuticals.

3.7. Hospital Administration

- Managing overall hospital resources like beds, rooms, and equipment.
- Coordinating between different hospital departments (e.g., pharmacy, lab, surgery).
- Managing the hospital's finances, budget, and reports.

3.8. Emergency & Bed Management

- Handling emergency cases and triaging patients.
- Managing bed availability and patient admissions in real-time.

3.9. Reporting & Analytics

- Generating performance and financial reports.
- Analyzing patient data for trends in health issues, medication, and hospital performance.
- Providing actionable insights for better decision-making.

4. Features of a Hospital Management System

4.1. Patient Registration and Management

- Registration of new and returning patients.
- Updating patient profiles and history.
- Managing appointments and consultations with doctors.

4.2. Billing and Financial Management

- Real-time billing generation and processing.
- Payment tracking and insurance claim management.
- Accounting reports and audits.

4.3. Appointment Scheduling

ENROLLMENT NO:2303031240064

- Patients can book appointments online or through the system.
- Real-time availability of doctors and specialists.
- Automated SMS/Email reminders to patients and doctors about scheduled appointments.

4.4. Electronic Health Record (EHR) Management

- A comprehensive record of patient's medical history, allergies, treatments, and diagnoses.
- Integration with diagnostic tools like labs and imaging systems.
- Easy sharing of medical data across healthcare departments.

4.5. Laboratory and Radiology Integration

- Integration with laboratory equipment and radiology systems for seamless reporting.
- Automatic updates for diagnostic results and tests.
- Easy access to lab reports by doctors and patients.

4.6. Prescription Management

- Automated prescription generation with detailed medication schedules.
- Tracking medication history and dosage.
- Alerts for drug interactions or allergies.

4.7. Inventory and Pharmacy Management

- Stock tracking for medications, medical supplies, and equipment.
- Alerts for low stock levels and expiry of medicines.
- Efficient inventory ordering and management.

4.8. HR & Payroll Management

- Managing the hospital staff roster, shifts, and attendance.
- Payroll calculation, salary generation, and leave management.

4.9. Real-Time Bed and Room Management

- Tracking room availability and patient bed allocation.
- Managing patient transfers, discharges, and room assignments.

4.10. Emergency Management

- Handling emergency situations and fast-tracking patient admission.
- Integration with ambulance services and emergency medical teams.

4.11. **Data Security and Backup**

- Ensuring patient data privacy and complying with medical data protection laws (like HIPAA).
- Regular data backup and disaster recovery solutions.

4.12. User-Friendly Interface

- Easy-to-navigate dashboard for hospital staff, doctors, and administrators.
- Accessible mobile and web-based versions for convenience.

5. **Benefits of a Hospital Management System**

- **Increased Efficiency**: Automates routine tasks like billing, record-keeping, and appointment scheduling, leading to faster and more accurate operations.
- **Improved Patient Care**: With quick access to patient records, diagnosis, and treatment history, doctors can make informed decisions.
- **Cost Reduction**: Reduces the operational costs by streamlining processes and minimizing errors.
- **Better Resource Utilization**: The system helps in better managing hospital resources, such as rooms, staff, and medical equipment.
- **Regulatory Compliance**: HMS ensures compliance with medical regulations and standards, such as patient data protection laws.
- **Data-driven Insights**: The system provides actionable data for hospital administrators to improve operational performance and patient outcomes.

---

6. **Types of Hospital Management Systems**

1. **Standalone HMS**: A basic, often smaller, system used by smaller healthcare providers or individual clinics.

2. **Integrated HMS**: A comprehensive system that integrates all departments of a hospital, including clinical, administrative, financial, and operational aspects.
3. **Cloud-based HMS**: An HMS hosted on cloud servers, offering scalability, remote access, and cost-effectiveness.
4. **On-premise HMS**: An HMS deployed on the hospital's own servers, providing more control but requiring higher upfront costs.

PRACTICAL-2

AIM:Identify suitable design and implementation model from the different software engineering models.

PROTOTYPE MODEL:

Selecting the Prototype Model for software development, such as for a system like PULMS (Public University Learning Management System), offers several advantages.

Below are the key reasons to choose the Prototype Model:

1. Evolving Requirements:

Reason: The Prototype Model is ideal when the system requirements are unclear or evolving. In many projects, especially large-scale systems like PULMS, users may not have a precise idea of what they want initially. Prototyping allows requirements to evolve over time as users interact with early versions of the system and provide feedback.

Benefit: You can quickly adapt to changing needs, helping to prevent miscommunication or misunderstanding of user requirements.

2. Early User Feedback:

Reason: A key advantage of the Prototype Model is its ability to gather early feedback from users. Users get to interact with a working version of the system early in the process, which helps them clarify their needs and expectations.

Benefit: The system can be refined based on real-world user feedback, reducing the risk of developing features that are not useful or desired by the users.

3. Improved Communication with Stakeholders:

Reason: Since prototypes are tangible and interactive, stakeholders (students, instructors, administrators) can provide more specific and actionable feedback. It fosters better communication between developers and users.

Benefit: By showing stakeholders a working prototype, you ensure that everyone involved has a clearer understanding of the system's capabilities, and any misunderstandings about the functionality can be addressed early.

Reasons not to select other models:

1. Waterfall Model:

The Waterfall Model is not flexible and doesn't allow for iterative changes based on user input, making it less suitable for PULMS, where ongoing user engagement and requirement evolution are essential.

2. Spiral Model:

While the Incremental Model works well for certain types of systems, its reliance on pre-determined increments may be less adaptable for PULMS, where user feedback can lead to changes in feature requirements throughout development.

3. Agile Model:

While Agile can be beneficial in environments where flexibility and user feedback are paramount, the constant iterations and demands for frequent releases could be challenging for larger systems like PULMS that may need a more structured approach to ensure all features integrate smoothly.

PUMLS

Software engineering model

1. Requirements Gathering:

Objective: In this initial phase, you gather high-level requirements that are essential for building the prototype. The goal is to identify core functionalities and features needed in PULMS but without focusing on detailed specifications at this point.

Process:

Engage with key stakeholders (e.g., students, faculty, administrators) to understand the essential features of the system.

Identify major functionalities like course management, student registration, assignment submission, grading system, discussion boards, and user profiles.

Focus on the basic requirements that will form the foundation of the prototype. Detailed features, performance expectations, and non-functional requirements (like security) may be deferred for later stages.

Example Requirements for PULMS:

Users can log in and view a dashboard of courses.

Students can submit assignments and view grades.

Instructors can upload course content, create assignments, and grade submissions.

2. Quick Design:

Objective: After gathering the basic requirements, a quick design of the prototype is created. This design will focus on just the most important functionalities that need to be demonstrated in the prototype.

Process:

Sketch out basic user interfaces (UI) and define user flows.

Use simple design tools (e.g., wireframes or mockups) to visualize how the PULMS system will look and work.

ENROLLMENT NO:2303031240064

Ensure the design represents core functionality (e.g., user login, dashboard view, assignment submission interface) without building a complete, fully-detailed system.

At this point, the design doesn't focus on technical complexities or backend features but aims to demonstrate key workflows.

Example Quick Design for PULMS:

UI Elements: Basic wireframes for student and instructor dashboards.

User Flow: Steps for a student to log in, view courses, submit assignments, and check grades.

Functional Flow: How instructors upload content and grade assignments.

3. Build Prototype:

Objective: Build a working prototype based on the quick design. This prototype will have basic functionality but won't be fully developed or scalable.

Process:

Develop the prototype focusing on the core functionality identified in the requirements phase.

Use rapid development tools to create a basic version of the system that allows users to interact with it (e.g., using simple front-end and back-end tools).

The prototype will include basic operations such as user login, course enrollment, assignment submission, and basic grading, but may not have complex backend systems, data security, or scalability.

The prototype is built quickly to demonstrate the intended features to users.

Example Prototype Features for PULMS:

A student login screen that allows students to access their courses.

A simple "Upload Assignment" feature for instructors.

Basic notification functionality when assignments are graded.

4. Evaluate Prototype:

Objective: After the prototype is developed, evaluation is conducted with real users (students, instructors, administrators) to gather feedback on how the prototype meets their needs and expectations.

Process:

Present the prototype to stakeholders (students, faculty, and administrators) to interact with and provide feedback.

Gather insights on the usability, functionality, and design. Ask questions about whether the features are intuitive, whether the system meets their needs, and where improvements can be made.

Evaluate user interactions with the system to identify issues in navigation, design, or workflow.

Document feedback on both the positive aspects of the system and areas that require improvement.

Example Evaluation for PULMS:

Students may comment on whether the assignment submission process is clear and easy to follow.

Instructors might provide feedback about how easy it is to upload course materials and grade assignments.

Administrators may highlight concerns related to security and data privacy.

5. Refine Prototype:

Objective: Based on the feedback from the evaluation phase, the prototype is refined. This may involve adding new features, improving usability, or fixing problems identified in the evaluation stage.

Process:

Modify the prototype to incorporate the feedback from users. This can include enhancing UI/UX, adding new features (e.g., notifications for assignment deadlines), or removing unnecessary elements.

Refine the design and functionality to address the issues discovered during evaluation. This may involve iterative cycles of development and feedback until the system reaches an acceptable state.

The system may undergo design iterations (UI redesign, improved navigation) or feature enhancements (e.g., adding automated grading).

Example Refinements for PULMS:

Improve the UI for assignment submission based on user feedback (e.g., adding drag-and-drop for file uploads).

Add a notification system to alert students when new content is uploaded or when assignments are graded.

Implement a basic grading system that automatically calculates grades based on student submissions.

6. Testing:

Objective: The refined prototype undergoes testing to ensure that the features work as expected and that the system meets both functional and non-functional requirements.

Process:

Conduct functional testing to ensure that each feature of the PULMS prototype works correctly. This includes user authentication, assignment submission, and grading workflows.

Perform usability testing with real users to evaluate the system's ease of use and identify any user experience (UX) problems.

Test for bugs and ensure that there are no major issues in the prototype’s functionality.

Depending on the phase of the project, security testing (e.g., ensuring that user data is protected) and performance testing (e.g., checking system response times) might also be conducted.

Example Testing for PULMS:

Functional Testing: Ensure that students can submit assignments correctly and that instructors can grade them.

Usability Testing: Evaluate if users can easily navigate the system and complete tasks without confusion.

Security Testing: Test whether the student and instructor data are protected, particularly in assignment submission and grading.

7. Final Refinement and Iteration:

After testing, more refinements are made to address any issues found during testing, and the prototype continues to evolve until it meets the desired level of functionality and user satisfaction.

The process may continue through several cycles of prototyping, testing, and refinement, ultimately leading to a fully functional and stable PULMS system ready for deployment.

ENROLLMENT NO:2303031240064