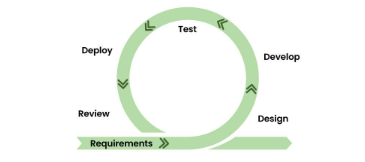
**CHAPTER 27** PROJECT SCHEDULING **727**



In this, two programmers work together due to which the code is error free and there are very few mistakes in it.

* In this the software project is completed in a very short time.
* In this the customer representative has an idea of ​​each iteration so that he can easily change the requirement.
* This is a very realistic approach to software development.
* In this, focus is given on teamwork.
* There are very few rules in this and documentation is also negligible.
* There is no need for planning in this.
* It can be managed easily.
* It provides flexibility to developers.

that lead to high-quality software engineering work (typical values for *P* range between 2000 and 12,000), and *t* is the project duration in calendar months.

Rearranging this software equation, we can arrive [SE\_Unit 5 83.docx?web=1](SE_Unit%205%2083.docx?web=1)at an expression for develop- ment effort *E*:

# As the project deadline becomes tighter and

3

*E* = *P*3*t*4

*L*

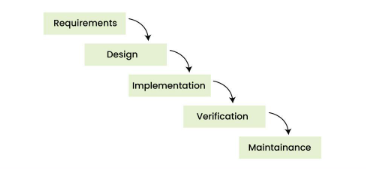
(27.1)

# tighter, you reach a point at which the work cannot be completed on schedule, regardless of the number of people doing the work. Face reality and define a new delivery date.

where *E* is the effort expended (in person-years) over the entire life cycle for software development and maintenance and *t* is the development time in years. The equation for development effort can be related to development cost by the inclusion of a burdened labor rate factor ($/person-year).

This leads to some interesting results. Consider a complex, real-time software project estimated at 33,000 LOC, 12 person-years of effort. If eight people are assigned to the project team, the project can be completed in approximately 1.3 years. If, how- ever, we extend the end date to 1.75 years, the highly nonlinear nature of the model described in Equation (27.1) yields:

**CHAPTER 27** PROJECT SCHEDULING **728**



1. **Requirement phase:-** Requirement phase is the first phase of the [waterfall model](https://www.geeksforgeeks.org/waterfall-vs-agile-software-development-model/). In this phase the requirements of the system are collected and documented. This phase is very crucial because the next phases are based on this phase.
2. **Design phase:-** Design phase is based on the fact how the software will be built. The main objective of the design phase is to prepare the blueprint of the software system so that no problems are faced in the coming phases and solutions to all the requirements in the requirement phase are found.
3. **Implementation phase:-** In this phase, hardware, software and application programs are installed and the database design is implemented. Before the database design can be implemented, the software has to go through a testing, coding, and debugging process. This is the longest lasting phase in waterfall.
4. **Verification phase:-** In this phase the software is verified and it is evaluated that we have created the right product. In this phase, various types of testing are done and every area of ​​the software is checked. It is believed that if we do not verify the software properly and there is any defect in it then no one will use it, hence verification is very important. One advantage of verification is that it reduces the risk of software failure.
5. **Maintenance phase:-** This is the last phase of waterfall. When the system is ready and users start using it, then the problems that arise have to be solved time-to-time. Taking care of the finished software and maintaining it as per time is called maintenance.
6. **Advantages of Waterfall Model**

This model is simple and easy to understand.

This is very useful for small projects.

This model is easy to manage.

The end goal is determined early.

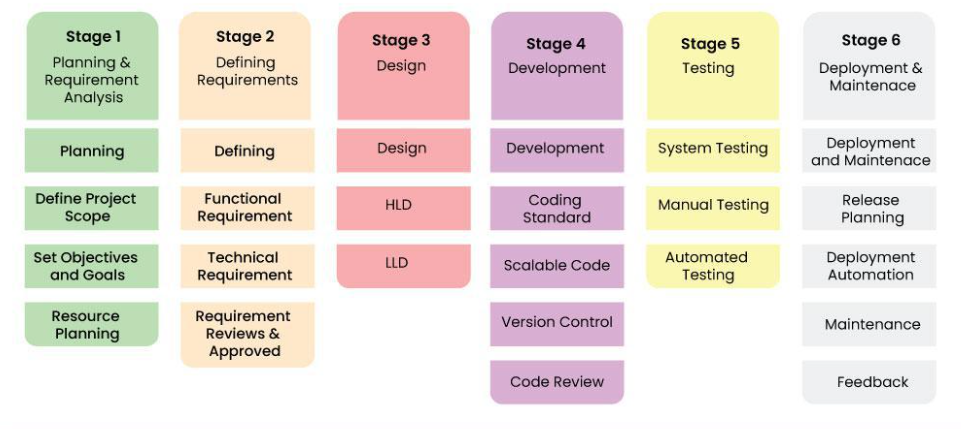
Each phase of this model is well explained.

It provides a structured way to do things.

This is a base model, all the SDLC models that came after this were created keeping this in mind, although they worked to remove its shortcomings.

In this model, we can move to the next phase only after the first phase is successfully completed so that there is no overlapping between the phases.

**CHAPTER 27** PROJECT SCHEDULING **729**



**SDLC (Software Development Life Cycle)** is a structured approach to software development that outlines the stages involved in building, deploying, and maintaining software applications. It helps ensure that the software meets the required quality standards and is delivered on time.

**Phases of SDLC:**

**Requirement Analysis:**

In this phase, the project’s objectives, requirements, and scope are defined. The goal is to understand what the users need from the software, including functional and non-functional requirements.

**System Design:**

This phase involves creating the architecture and design for the software. The system is divided into smaller modules and defined in detail. Designers specify the system’s components, their interactions, and how the system will meet the requirements.

**Implementation (Coding):**

In this phase, developers start coding the system as per the design specifications. This phase often takes the most time, and the code is tested for correctness.

**Testing:**

After development, the software is rigorously tested to ensure it meets the defined requirements and functions as expected. Testing may include unit testing, integration testing, system testing, and user acceptance testing (UAT).

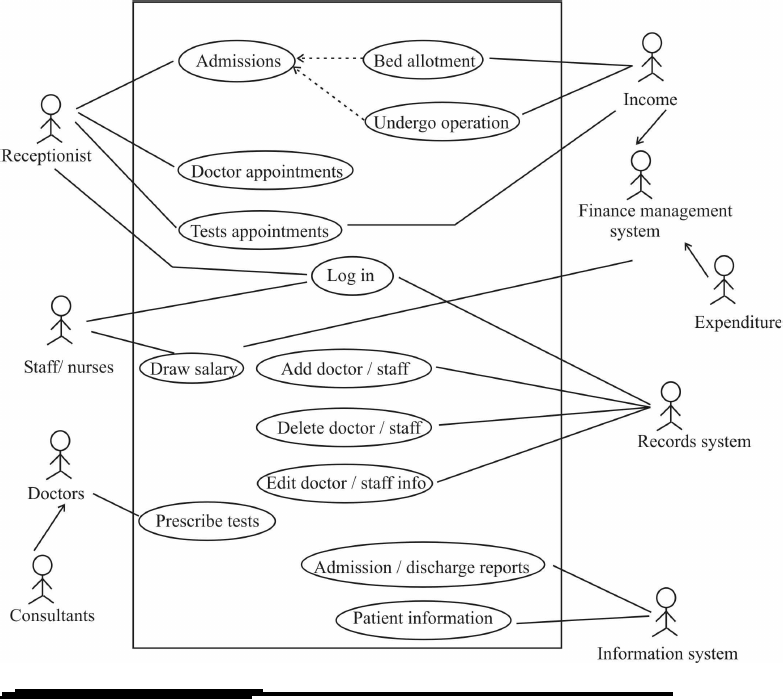
**Deployment:**

* + Once the software passes all tests, it is deployed for use. This could be a full rollout or a staged deployment depending on the organization’s needs.

**Maintenance:**

* + After deployment, the software enters the maintenance phase, where it is monitored for issues, bugs are fixed, and updates or enhancements are made based on user feedback.

**CHAPTER 27** PROJECT SCHEDULING **730**



The Software Development Lifecycle (SDLC) for a Hospital Management System (HMS) involves structured phases to guide development while ensuring customer needs are met. **1. Requirements Gathering and Analysis:** Stakeholders like doctors, patients, and administrators are interviewed to gather and document requirements, including features like patient records, appointment scheduling, and reminders. Regular consultations with stakeholders and requirement validation ensure customer needs are captured. **2. System Design:** The architecture is designed (e.g., client-server or cloud-based), and user interfaces and database schemas are developed for managing records and appointments. Security mechanisms are planned. Designs are reviewed with stakeholders, and prototypes are provided for feedback. **3. Implementation (Coding):** Developers use the MERN stack to write code for modules like appointment booking and prescription management. Agile development ensures iterative feedback from users, and working modules are tested early. **4. Testing:** Unit, integration, system, and user acceptance testing (UAT) are performed to ensure functionality and usability. Feedback from UAT refines the system, addressing issues raised by users. **5. Deployment:** The HMS is deployed in a production environment, and users are provided with training and documentation. A gradual rollout minimizes disruptions and ensures smooth adoption. **6. Maintenance and Updates:** System performance is monitored, bugs are resolved, and features are updated to adapt to changing needs. Regular feedback sessions ensure user satisfaction. At each phase, customer involvement through reviews, testing, and feedback ensures the system evolves to meet user needs and remains effective over time.

**CHAPTER 27** PROJECT SCHEDULING **731**

Critically analyze the limitations of using the Waterfall model for the Hospital ManagementSystem. Justify the adoption of an evolutionary process model by explaining how it accommodates dynamic healthcare requirements, ensures adaptability to feedback from medical staff, and supports iterative improvements. Your analysis should consider constraints such as the need for real-time updates, compliance withhealthcare regulations, and integration with existing hospital systems?

.The Waterfall model has significant limitations when applied to a Hospital Management System (HMS) due to its rigid structure, delayed feedback, limited adaptability, and challenges with integration. It struggles to accommodate evolving healthcare requirements, such as changes in regulations or stakeholder feedback, and delays adjustments until later stages, potentially misaligning the final product with user needs. In contrast, an evolutionary process model like Agile or Spiral is better suited for HMS as it supports dynamic requirement handling, iterative improvements, and frequent feedback cycles. It enables early delivery of critical features, allows for real-time updates, and ensures compliance with new healthcare regulations through regular testing and iterations. Additionally, this approach facilitates seamless integration with existing hospital systems, addressing complexities incrementally while ensuring adaptability and responsiveness. Thus, the evolutionary model is ideal for developing an HMS that meets the dynamic, high-stakes needs of the healthcare sector.it is monitored for issues, bugs are fixed, and updates or enhancements are made based on user feedback.

Design a process flow using the incremental process model for developing the core modules of a Hospital Management App, such as patient registration, appointment

The incremental process model for developing a Hospital Management App begins with the **Patient Registration Module**, where a basic registration system is created, including fields for name, age, contact, and medical history, supported by a secure database schema. This module is integrated with a user-friendly interface for data entry and tested for functionality, followed by refinements based on feedback from administrative staff, such as adding input validation and multi-language support. Next, the **Appointment Scheduling Module** is developed to allow patients to book, reschedule, and cancel appointments while providing doctors with a calendar to manage availability. This module is integrated with the registration system, ensuring only registered patients can schedule appointments, and synced with the database for storing appointment details. Feedback from healthcare professionals ensures usability, leading to iterative enhancements like automated reminders and conflict resolution features. Finally, the **Medical Records Management Module** is built to store patient medical histories, diagnoses, and prescriptions. This module integrates with the previous ones, enabling doctors to access records during appointments. Iterative testing and feedback from both doctors and patients ensure continuous improvement, addressing usability, security, and compliance with healthcare regulations.

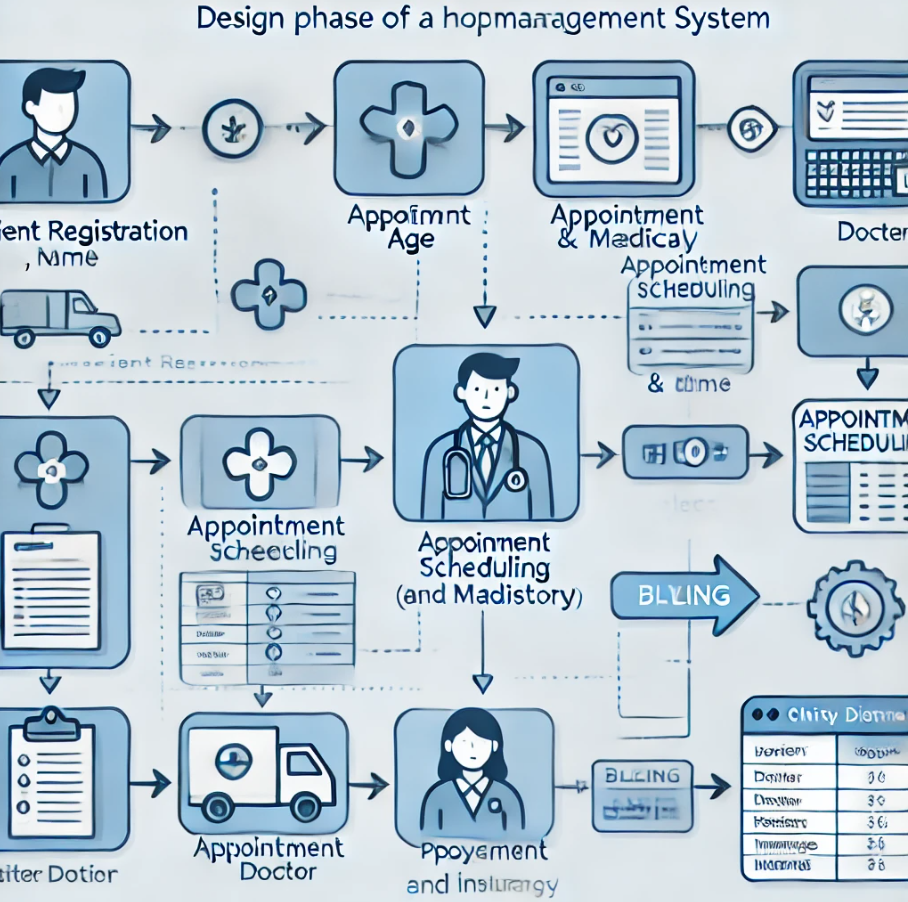
**CHAPTER 27** PROJECT SCHEDULING **732**

Waterfall Model Nature: Linear and sequential. Phases: Requirements → Design → Implementation → Testing → Deployment → Maintenance. Used in: Well-defined projects with clear, unchanging requirements (e.g., government or large-scale enterprise systems). Incremental Model Nature: Iterative and modular. Phases: Requirements → Incremental Design and Development → Integration → Testing → Deployment (per increment). Used in: Projects where functionality can be developed in stages (e.g., mobile apps or web applications). Spiral Model Nature: Risk-driven and iterative. Phases: Planning → Risk Analysis → Engineering (Development & Testing) → Evaluation (per spiral). Used in: High-risk projects that need frequent risk analysis (e.g., large software systems or embedded systems). Agile Model Nature: Iterative, incremental, and flexible. Phases: Concept → Iterative Development → Testing → Feedback → Deployment (per sprint). Used in: Dynamic projects with evolving requirements (e.g., startups, software for rapidly changing markets). Prototype Model Nature: Exploratory and iterative. Phases: Requirements → Prototype Development → Feedback → Refinement → Final Development. Used in: Projects with unclear requirements or when user feedback is critical early on (e.g., user interface design). V-Model Nature: Verification and validation-focused. Phases: Requirements ↔ Acceptance Testing → Design ↔ System Testing → Implementation ↔ Unit Testing. Used in: Projects requiring high reliability and strict validation (e.g., safety-critical systems like medical devices). Big Bang Model Nature: Unstructured and experimental. Phases: Minimal Planning → Development → Testing → Delivery. Used in: Small or experimental projects with flexible requirements (e.g., proof of concepts or small prototypes). RAD (Rapid Application Development) Model Nature: Fast, user-focused, and iterative. Phases: Requirements → Rapid Prototyping → Feedback → Refinement → Finalization. Used in: Projects requiring quick development and user feedback (e.g., business applications, small-scale systems).

**CHAPTER 27** PROJECT SCHEDULING **732**

List the primary functional and nonfunctional requirements for the Hospital Management System project. What are thé pros and cons of each requirement, and how do they contribute to the success of the system in managing patient data, appointments, billing, and overall healthcare operations?

The **Hospital Management System (HMS)** requires both functional and non-functional requirements to ensure smooth operation and effectiveness. Functional requirements include **Patient Registration**, which facilitates accurate data entry for patients but requires careful validation to avoid errors. **Appointment Scheduling** optimizes doctor-patient interactions but requires real-time synchronization for accuracy. **Medical Records Management** enables easy access to patient history, although it demands robust security protocols to protect sensitive information. **Billing and Payment Processing** streamlines financial transactions but can be complex when integrating with multiple payment systems. **Inventory Management** ensures efficient resource availability but requires accurate tracking. Non-functional requirements include **Performance and Scalability**, ensuring the system can handle increased users and transactions without slowdown, although it can be resource-intensive. **Security** is essential for protecting patient data and complying with regulations, but implementing strong security measures can be costly. Finally, **Usability** ensures ease of use for healthcare providers, though it requires user-centered design to minimize errors. These requirements collectively contribute to a secure, efficient, and user-friendly system that supports patient care, billing, and hospital operations effectively.



**CHAPTER 27** PROJECT SCHEDULING **731**

List the key stakeholders in a Hospital Management System project and explain their roles.

Key stakeholders in a **Hospital Management System (HMS)** project include **hospital administrators**, who oversee operations, set system goals, and make high-level decisions; **doctors and medical staff**, who use the system to access patient records, schedule appointments, and prescribe medications; **patients**, who interact with the system to book appointments, view medical records, and receive reminders; **IT team/developers**, responsible for designing, developing, and implementing the system, ensuring its security and functionality; and **nurses and support staff**, who assist in patient care, manage daily operations, and interact with the system for patient data management. Each stakeholder plays a critical role in ensuring the system meets the hospital’s operational and patient care needs

Explain the significance of risk management in ensuring the success of a Hospital Management System project under tight deadlines. Analyze the risks associated with delays in the project and propose proactive strategies to mitigate them.

Risk management is crucial in ensuring the success of a **Hospital Management System (HMS)** project under tight deadlines as it helps identify, assess, and mitigate potential obstacles that could impact project delivery. Delays in the project can lead to increased costs, missed deadlines, reduced quality, and unsatisfied stakeholders. Risks include insufficient resources, technical challenges, inadequate testing, and scope creep. Proactive strategies to mitigate these risks include setting realistic milestones, regular progress reviews, allocating sufficient resources, maintaining clear communication among stakeholders, and implementing an agile development approach to adapt to changes quickly. These strategies ensure the project stays on track and meets the required deadlines without compromising quality.

Calculate size-oriented metrics using the following data: The Hospital ManagementSystem has 10 modules, with an average of 300 lines of code per module.

o calculate size-oriented metrics for the **Hospital Management System**, we can use the Lines of Code (LOC) as the primary metric. The system has 10 modules, with an average of 300 lines of code per module. Therefore, the total lines of code (LOC) for the entire system would be:

**Total LOC = 10 modules × 300 LOC/module = 3,000 LOC.**

This metric helps in estimating the system's size and serves as a foundation for further analysis, such as estimating development effort, testing, and maintenance.