MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE REPUBLIC OF KAZAKHSTAN
SDU UNIVERSITY

Imankulova B.B.

PROJECT MANAGEMENT PRACTICAL WORKBOOK

Study guide

Kaskelen, 2026

UDC 005.8
LBC 65.290-2
I-10

Reviewers:

D. Zholdybai - Assist. Professor of the Department of Information Systems, SDU University

G. Alin – Assist. Professor of the Department of Cybersecurity, IITU

T. Nurlybayev - Executive Director of the Public Association “Internet Society”

B.B. Imankulova. Project management practical workbook: Study guide / B.B. Imankulova. – Kaskelen: SDU University, 2026. – 115 p.

ISBN 978-601-7647-46-9

This study guide provides a structured overview of practical project management for university students. It explains key concepts and shows how they connect through real project documents, from project definition and scope planning to scheduling, budgeting, quality control, communication, risk management, and project closure. The purpose of the guide is to help students understand why each tool is used, apply it step by step to one case project, and build consistent documentation that supports planning and control. It serves as a concise reference and a practical companion to the “Project Management Practical Workbook” for developing skills needed to manage real projects effectively.

Recommended by the AC of the University (protocol No. 9_dated "23" February 2026) for publication in the open press under the stamp of the Ministry of Science and Higher Education of the Republic of Kazakhstan and use in the educational process as a guide for students in the undergraduate specialty 6B06101 – Computer Science and 6B06101 - Information Systems.

ISBN 978-601-7647-46-9

CONTENTS

LIST OF ABBREVIATIONS

AI – Artificial Intelligence
BA – Business Analyst
BOM – Bill of Materials
CI – Continuous Integration
COCOMO – Constructive Cost Model
CPA – Critical Path Analysis
CPM – Critical Path Method
CR – Change Request
DevOps – Development and Operations
ERP – Enterprise Resource Planning
ETC – Estimate to Complete
EV – Earned Value
EVM – Earned Value Management
Gantt – Gantt Chart (a project scheduling tool)
HR – Human Resources
IRR – Internal Rate of Return
IT – Information Technology
KPI – Key Performance Indicator
MVP – Minimum Viable Product
OBS – Organizational Breakdown Structure
PERT – Program Evaluation and Review Technique
PM – Project Management / Project Manager
PMBOK – Project Management Body of Knowledge
PMO – Project Management Office
PO – Product Owner
PRINCE2 – Projects IN Controlled Environments (Version 2)
QA – Quality Assurance
QC – Quality Control
RACI – Responsible, Accountable, Consulted, Informed
RAID – Risks, Assumptions, Issues, Dependencies
RAM – Responsibility Assignment Matrix
RFP – Request for Proposal
ROI – Return on Investment
RTM – Requirements Traceability Matrix
SDLC – Software Development Life Cycle
Scrum – Agile Project Management Framework
SOW – Statement of Work
SME – Subject Matter Expert
SPI – Schedule Performance Index
SLA – Service Level Agreement
TCO – Total Cost of Ownership
TQM – Total Quality Management
WBS – Work Breakdown Structure

INTRODUCTION

This Practical Workbook supports a hands-on approach to project management. It helps you learn how to plan, organize, control, and close a project through real project documents and simple management tools. The workbook follows the full project life cycle and uses consistent logic, each topic leads to a practical output that a project manager can use in real work.

The main goal of the workbook is to build applied skills, not only theoretical knowledge. For this reason, every part includes short explanations and a clear task. As you progress through the sections, you will create a complete project documentation package: from the project charter and stakeholder register to the schedule baseline, risk register, status reports, and final report. At the end, you will have a structured set of materials that reflect professional project management practice.

This workbook can support individual work or teamwork. It fits academic courses, training programs, and independent study. It also suits different project types, because the templates and steps remain the same even when the project topic changes.

HOW TO USE THIS WORKBOOK

Start by selecting one project case for all tasks. Use the same project throughout the workbook to keep all documents consistent. Your project may be academic, business, or technical, but it must include clear objectives, a realistic scope, key stakeholders, and a defined time frame.

Complete the workbook in sequence. Each section builds on previous outputs. Do not skip early documents, because later tasks depend on them. For example, the WBS supports the activity list, the activity list supports the schedule, and the schedule supports cost planning and progress control.

For each chapter, follow the same workflow: review the key concepts, apply the instructions to your project case, complete the practical task, and fill in the template. Use the checklist to verify quality and correct errors before submission.

Keep all project documentation in one folder (digital or paper). Save each completed template with a clear file name and version number. For each revision, record the date and the reason for the update. This practice improves traceability and supports project control.

Use clear and measurable wording. Write objectives as outcomes, not activities. Define scope through deliverables. State assumptions as clear conditions. Provide realistic schedule and cost estimates and add a short basis for each estimate. In risk management, focus on the most important risks and select responses that match the project context.

During execution and control, update status reports regularly. Record issues as soon as they appear. Use the change request form when scope, schedule, or cost requires an official decision. Track decisions in the decision log to prevent confusion and repeated discussions.

At the end of the workbook, complete closure documents: acceptance checklist, handover plan, final project report, and lessons learned. These steps support professional practice and improve future projects.

Assessment considers both the quality of each document and the consistency between them. A strong submission shows alignment across scope, schedule, cost, quality, resources, communication, risks, and stakeholder expectations.

This workbook uses a running case example: Service Desk Web Portal v1.0.

1 FOUNDATION OF PROJECT MANAGEMENT

1.1 Project management basics

A project is a unique set of processes that includes coordinated and controlled activities with defined start and end dates, carried out to meet project objectives. To achieve these objectives, the project must deliver outputs that match specific requirements. A project may also face multiple constraints [1].

Project management is a way to plan, organize, and control resources to meet specific goals on time and within budget. It is especially important in Information Technology (IT), where fast changes in technology and complex project needs require effective management to ensure success.

Project management is the structured use of knowledge, skills, methods, tools, and techniques to achieve a project's defined objectives. It covers all phases of the project life cycle and relies on organized processes. Project success depends on balancing key constraints such as scope, time, cost, quality, resources, and risk [1].

Project management focuses on balancing three core constraints: scope, time, and cost. This concept, often called the project management triangle, requires effective control so the project meets client expectations and business objectives.

- Scope – the work required to deliver the product or service
- Time – the schedule and deadlines for completion
- Cost – the budget and available financial resources

The triangle illustrates (see Figure 1.1, Figure 1.2) the interdependence of the three variables. A change in one requires an adjustment in the other two to maintain balance. If one-point shifts without corresponding changes to the others, the integrity of the triangle is lost and the project's quality declines.

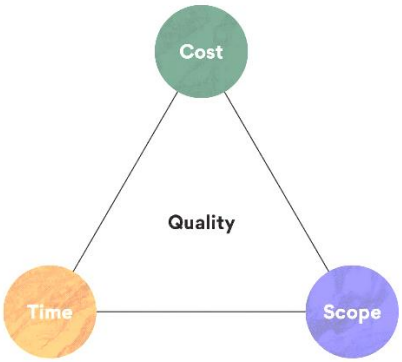


Figure 1.1 - The project management triangle [3]

The project manager must keep the three sides of the triangle in balance to deliver the required quality, while remaining within the approved budget, meeting deadlines, and following project requirements.

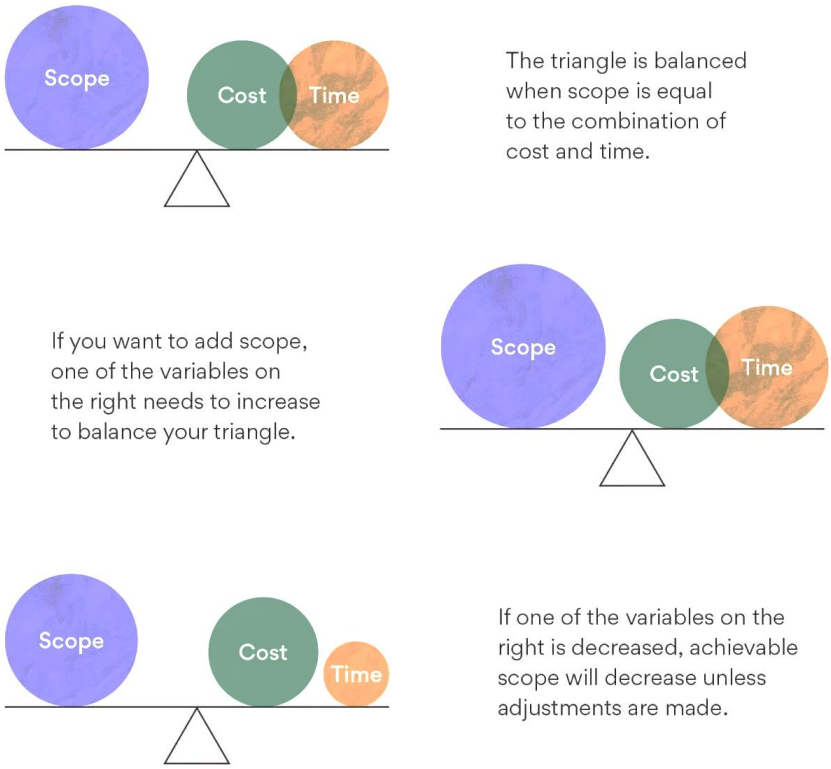


Figure 1.2 – Balancing the triple constraints [3]

In practice, project management also considers additional factors:

- Quality – the degree to which outcomes meet defined standards
- Resources – the personnel, equipment, tools, and materials required
- Risk – potential events that may affect project success

These constraints are interconnected. A change in one often influences the others. For instance, expanding the scope usually demands more time and budget. The role of a project manager is to achieve project goals while maintaining balance across all constraints.

Project managers must maintain a careful balance between these constraints. A change in one area often affects the others. For example, expanding the scope of a project may require more time and funding. Effective project management aims to achieve goals while remaining within these constraints.

Table 1.2 - Example of Service Desk Web Portal v1.0

Item	Sample answer
1) Project goal (1 sentence)	Deliver a web-based Service Desk portal that allows university users to submit and track support tickets within 10 weeks.
2) Key deliverables (5–7)	1) Ticket submission form 2) User dashboard (ticket status) 3) Agent panel (assignment and updates) 4) Admin settings (categories/priorities) 5) Email notifications 6) Basic reports

3) Constraints (3)	1) Time: 10-week deadline 2) Budget: fixed limit approved by sponsor 3) Scope: no mobile app in v1.0
4) Assumptions (3)	1) Users have university accounts for login 2) IT department provides feedback on time 3) Hosting is available in university infrastructure
5) Success criteria (3 measurable)	1) Go-live by Week 10 2) ≥ 90% acceptance test cases pass 3) Average ticket creation time ≤ 2 minutes

Learning objectives

By the end of this section, you can:

- define a project;
- write a clear goal;
- identify deliverables, constraints, assumptions, and success criteria.

Practical tasks:

1. Write the project goal in one sentence for the chosen topic of the project.
2. List 5-7 key deliverables that prove project completion.
3. Write three constraints (time, cost, or scope).
4. Write three assumptions that must stay true for the plan to work.
5. Define three measurable success criteria.

Table 1.2 - Template of project summary

Project title	
Project goal (one sentence)	
Key users	
Key stakeholders (3 - 5)	
Key deliverables (5 - 7)	
Constraints (3)	
Assumptions (3)	
Success criteria (3 measurable)	
Notes / comments	

1.2 Project life cycle and process groups

Every project follows a life cycle. The life cycle describes the path of a project from the first idea to the final product. It helps the team understand what happens first, what comes next, and what must be completed before the project ends.

Many organizations describe project work through five process groups. The groups can overlap, but each group has a distinct purpose (see Figure 1.3).

1. Initiation

The project begins with an idea or need. At this stage, the sponsor authorizes the project and confirms the business purpose. The team defines high-level goals, scope boundaries, and key stakeholders. The organization often assigns a project manager and approves initial resources. The key question is: Should we start this project?

2. Planning

Planning creates the foundation for successful delivery. The team defines scope, schedule, cost, quality, resources, risks, and communications. The project manager prepares the project plan, and the team develops key planning documents such as the WBS, activity list, and risk register. The key question is: How will we complete the project?

3. Execution

During execution, the team performs the work and produces deliverables. The project manager coordinates tasks, supports the team, and manages stakeholders and vendors. The project delivers products or services according to the plan and quality requirements. The key question is: Are we producing the planned results?

4. Monitoring and Controlling

This process group runs throughout the project. The project manager tracks progress, compares actual results with the baselines, and identifies deviations. The team manages changes, controls risks and issues, and takes corrective action when needed. The key question is: Are we on track, and what must change?

5. Closure

Closing completes the project in a formal way. The sponsor or client confirms acceptance, and the team hands over deliverables to users or operations. The project closes contracts, releases resources, and documents lessons learned for future improvement. The key question is: What did we deliver and what did we learn?

The five process groups can be summarized as follows:

Initiating: authorize the project and define high-level goals.

Planning: define scope, schedule, cost, quality, resources, risks, and communications.

Executing: build deliverables and manage the team and vendors.

Monitoring and Controlling: track performance, manage changes, and take corrective action.

Closing: confirm acceptance, hand over deliverables, and capture lessons learned.

Figure 1.3 - Project Life Cycle

Learning objectives

By the end of this section, you can:

– describe the project life cycle and the five process groups;
– match each process group with a key output (document or result);
– choose simple metrics to monitor progress and quality during execution.

Practical tasks:

1. For the project, list one key output for each process group.
2. Write two monitoring metrics you will track during execution (example: defects, schedule variance).

Table 1.3 - Template of a process groups map

Process group	Purpose	Key outputs for this case
Initiating		
Planning		
Executing		
Monitoring and controlling		
Closing		

Table 1.4 – Template of a process groups map (sample answer for Service Desk Web Portal v1.0)

Process group	Purpose	Key outputs for this case
Initiating	Authorize and align	Approved project charter; named project manager; initial stakeholder list

Planning	Define baselines and plans	Scope statement; WBS; activity list; schedule baseline; cost estimate; risk register; communication plan
Executing	Build deliverables	Developed portal features (ticket submission, dashboards); configured categories and priorities; test builds; training draft
Monitoring and controlling	Track and correct	Weekly status reports; issue log; change requests and change log; updated risk register; quality check results
Closing	Accept and hand over	Acceptance checklist; handover plan; final project report; lessons learned document

Project management frameworks and methodologies.

Project management frameworks provide structured approaches that guide managers and teams in planning, executing, and controlling projects. A framework typically defines roles, processes, governance rules, and supporting tools to help deliver project outcomes within time, cost, scope, and quality constraints. The choice of framework depends on how stable the requirements are, how complex the work is, and how much flexibility is needed during delivery.

It is useful to distinguish related terms. A *framework* describes the overall structure and governance (roles, ceremonies, artifacts, decision points). A *methodology* describes the practical way of working day to day (planning, execution, feedback, and improvement cycles). In addition, *standards and guidance* (for example, PMBOK-based practices or ISO-style guidance) provide common terminology and recommendations, but they do not prescribe a single delivery approach.

Several methodologies are commonly used in project management, and each is suited to different project conditions (Figure 1.4):

- **Waterfall** is a linear, sequential approach that works well when requirements are stable and fully defined early, and when changes are expected to be limited.
- **Agile** is an iterative approach designed for evolving requirements and frequent stakeholder feedback, making it common in software and digital product development.
- **Lean** focuses on efficiency by reducing waste, streamlining processes, and improving value delivery.
- **Scrum** is a framework within Agile that organizes work into short iterations (sprints), supports frequent feedback loops, and is effective for complex work that benefits from rapid learning and adjustment.

Other well-known frameworks and approaches include:

- **PRINCE2 (Projects in Controlled Environments)**, widely used in Europe, which emphasizes defined processes, clear governance, and structured documentation.
- **Kanban**, an Agile-oriented approach that uses visual workflow boards and work-in-progress limits to improve flow, predictability, and efficiency.
- **Hybrid models**, which combine elements of traditional approaches (such as Waterfall stage gates and formal documentation) with Agile practices (such as iterative

delivery and continuous feedback). Hybrid delivery is common in large organizations where compliance and governance must coexist with flexibility.

Selecting an appropriate framework is typically based on practical criteria. Key factors include the stability of requirements, risk and uncertainty level, regulatory or documentation needs, stakeholder availability for regular feedback, team size and distribution, and the expected delivery cadence (single release versus continuous delivery). Good fit improves decision-making, makes progress visible, supports risk control, and increases the likelihood of successful delivery.

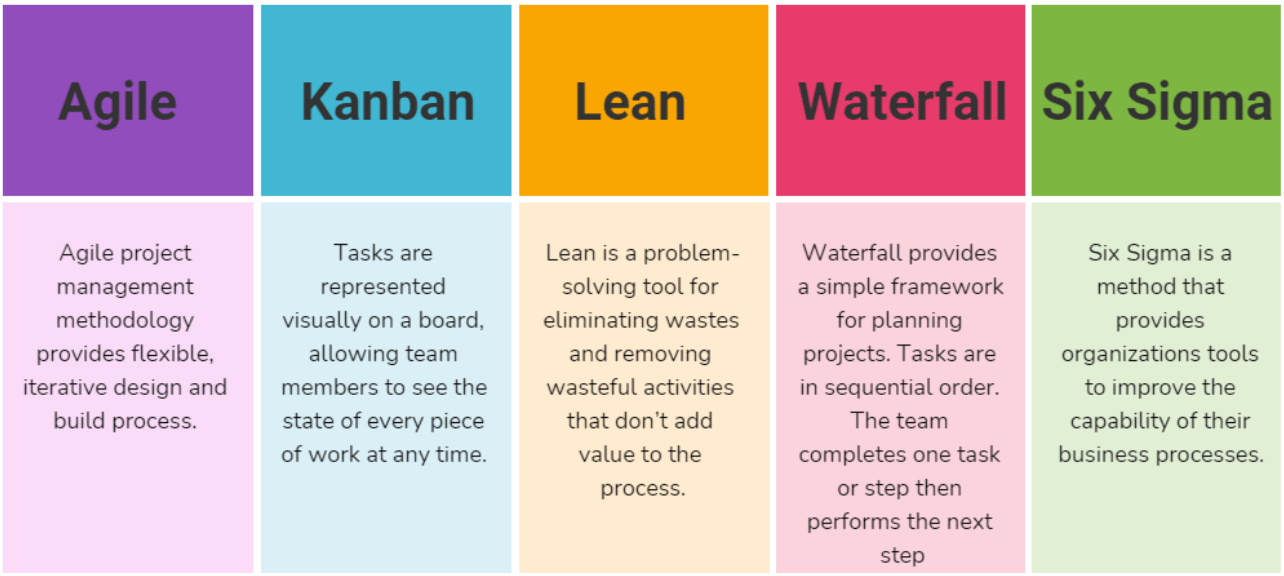


Figure 1.4 - PM methodologies

1.3 Roles and Responsibilities

Different roles define responsibility and authority in the project. Clear roles prevent confusion and improve teamwork.

Project Sponsor is a senior leader who authorizes the project. The sponsor secures funding, approves major changes, and represents the project at the organizational level. Without sponsor support, a project usually cannot succeed.

Project manager is a professional who leads a project from initiation to closure. This person sets project objectives, develops plans, allocates resources, and directs the team to complete the work successfully. The project manager makes sure the project follows the approved scope, stays within budget, meets deadlines, and delivers results that meet quality requirements.

The project manager also acts as the main communication link among stakeholders, the project team, and senior management. The role includes risk oversight, decision-making, and team coordination to keep all work aligned with project goals.

An organization appoints the project manager to achieve the project objectives. Depending on the organizational structure, the project manager may report to a functional manager or an operations manager. In some cases, the project manager reports

to a program or portfolio manager who supervises multiple projects and programs across the organization.

In IT projects, the project manager works with specialists from different areas, such as software development, cybersecurity, and systems integration.

Key responsibilities:

- Define project goals and deliverables
- Develop and manage the schedule and budget
- Assign tasks and coordinate team efforts
- Report on progress and address stakeholder concerns
- Identify and resolve project risks
- Oversee documentation and quality compliance

Team Members are specialists who perform tasks. They bring technical knowledge, solve problems, and contribute to results. Team members may come from different departments or work only for the project.

Stakeholders are people or groups affected by the project. They may include customers, end users, suppliers, or community members. Their feedback helps the project stay relevant and useful.

Example of Stakeholder list for Service Desk Web Portal v1.0 is shown in Table 1.5.

Table 1.5 – Stakeholder list (example: Service Desk Web Portal v1.0)

Stakeholder	Category	Interest / need	Influence (H/M/L)	Notes
Executive sponsor (CIO)	Sponsor	Visibility of service performance, cost control, strategic alignment	H	Provides funding and final approvals
IT operations manager	Sponsor	Efficient incident handling, SLA compliance	H	Owns operational KPIs
Service desk manager	Users	Better ticket tracking, reporting, workload management	H	Key business owner of the portal
Service desk agents	Users	Simple UI, fast ticket creation and updates	M	Daily users of the system
End users (employees)	Users	Easy issue reporting, status tracking, self-service	M	Large user group, indirect influence
Product owner	Team	Clear backlog, prioritization, value delivery	H	Represents business needs

UX/UI designer	Team	Usable and consistent interface	M	Influences usability and adoption
Backend developer	Team	Stable APIs, integration with ITSM tools	M	Responsible for core functionality
QA / test engineer	Team	System quality, defect prevention	M	Controls release readiness
IT security officer	Other	Access control, data protection, compliance	H	Can block release if requirements are not met
Infrastructure / DevOps engineer	Other	Reliable deployment, monitoring, availability	M	Supports deployment and operations
Training / HR representative	Other	User onboarding and training materials	L	Supports adoption phase

Learning objectives

By the end of this section, you can:

- identify key project stakeholders and group them by role;
- explain basic responsibilities of sponsors, users, and the project team;
- assign responsibility and accountability using a simple RACI matrix.

Practical tasks:

1. List 8-12 stakeholders for the chosen topic and group them as Sponsor, Users, Team, and Other.

2. Create a simple RACI for five activities: requirements approval, UX review, deployment approval, user training, and acceptance.

Table 1.6 – Stakeholder list

Stakeholder	Category	Interest / need	Influence (H/M/L)	Notes

1.4 Project documents overview

Project documents support a project from initiation to closure. They help the team plan the work, track progress, control changes, and communicate with stakeholders. Each document has a clear purpose: it either defines what the project must achieve, explains how the team will achieve it, or records what happened during execution.

At the start, the most important document is the project charter. It formally authorizes the project and names the project manager. The charter summarizes the project

purpose, objectives, high-level scope, key stakeholders, main risks, and major milestones. After approval, the team can develop detailed plans.

During planning, the project manager prepares the project management plan. This plan combines the main baselines and supporting plans. The scope baseline describes project deliverables and includes the scope statement, the work breakdown structure (WBS), and the WBS dictionary. The schedule baseline defines the planned timeline, usually shown as a network diagram and a Gantt chart. The cost baseline sets the approved budget and becomes the reference for cost control. Along with these baselines, the plan often includes sections for quality, resources, communications, risk, and procurement.

As the project moves into execution and control, the team uses working documents to monitor performance and manage issues. A register list contains key stakeholders and their expectations. Requirements list records what the client and users need. A risk register describes identified risks, their priority, and response actions. An issue log tracks problems that require attention and resolution. A change log records change requests and decisions, while a decision log captures major decisions and their reasons. Regular status reports summarize progress, current results, and next steps. If the project uses contracts, procurement documents such as vendor criteria, requests for proposals, and contract agreements also become part of the documentation set.

At closure, the project manager prepares documents that confirm completion and capture experience. An acceptance form verifies that the client approves deliverables. A handover document transfers the product or service to operations. A final project report summarizes outcomes against scope, schedule, cost, and quality goals. A lesson learned report records what worked well, what did not, and recommendations for future projects.

Strong documentation does not mean more paperwork. It means the project has the right documents, updated at the right time, and used by the team as practical tools for control and communication.

Project documents support planning, control, and communication. A small project can use a lightweight set of documents, but each document must have a clear purpose. The charter authorizes the project. The plan defines baselines and rules. Reports and logs support execution and control.

- Initiation: charter, stakeholder list, high-level risks.
- Planning: scope statement, WBS, schedule baseline, cost baseline, quality plan, communications plan, risk register.
- Execution and control: status report, issue log, change log, decision log, test reports, dashboards.
- Closure: acceptance checklist, handover plan, final report, lessons learned.

CHAPTER 2. PROJECT INITIATION

2.1 Project charter

A project charter is the formal document that starts the project. It gives the project official approval and confirms that the organization is ready to invest time, budget, and resources. The charter also assigns the project manager and defines the project manager's authority level. Without an approved charter, detailed planning often becomes unclear, because the team has no confirmed objectives, scope boundaries, or decision-making structure.

The main purpose of the charter is to explain why the project exists and what success looks like at a high level. It summarizes the business need or problem, sets measurable objectives, and defines the expected deliverables. The charter does not replace the project management plan. Instead, it creates a clear starting point for planning by establishing scope limits, major milestones, key stakeholders, and the main constraints.

A good charter uses simple and specific language. Objectives should describe outcomes, not actions. Scope must clearly separate what the project will deliver from what it will not deliver. Constraints must reflect real limits such as time, budget, technology, and available staff. Assumptions must describe conditions the team expects to be true. High-level risks must capture the main uncertainties that may affect delivery.

For the Service Desk Portal v1.0, the charter should explain the need for a unified web portal that supports ticket submission and tracking for university users. It should define the expected value for students, staff, and the IT Service Desk, and it should confirm the 10-week timeframe. After the sponsor approves the charter, the team can begin detailed planning activities such as stakeholder analysis, scope planning, schedule development, and cost estimation.

Table 2.1 – Example of project charter (Service Desk Web Portal v1.0)

Field	Sample content
Project title	Service Desk Web Portal v1.0
Sponsor	Head of IT Department (University)
Project manager	Assigned PM (part-time)
Business need / problem	The university currently receives support requests through phone calls, emails, and informal messages. This causes lost requests, slow response time, and limited reporting. A unified portal will standardize ticket submission and tracking for students and staff and will improve service transparency.
Project objectives (3)	1) Launch a working portal that supports ticket creation and tracking within 10 weeks. 2) Improve request handling by providing a clear ticket status and assignment process.

	3) Provide basic reports for the IT Service Desk (open/closed tickets, response time).
High-level scope (in /out)	In scope: user login, ticket submission form, user dashboard, agent panel (assign/update tickets), admin settings (categories, priorities), email notifications, basic reports. Out of scope: mobile app, external system integrations (ERP/LMS), AI/ML routing, multi-language support (optional for v1.0).
Major deliverables	Web portal (front-end + back-end), database for ticket data, admin panel, notification module, basic reporting page, user guide (short), deployment package.
High-level milestones (weeks)	Week 1: charter approved. Week 2: requirements approved. Week 4: UI prototype approved. Week 6: MVP demo. Week 8: system testing complete. Week 9: user acceptance testing (UAT). Week 10: go-live and handover.
Constraints and assumptions	Constraints: 10-week duration, fixed budget limit, small team capacity. Assumptions: university accounts are available for login, stakeholders provide feedback within 2 days, hosting and domain access are available.
High-level risks	Stakeholder feedback delays, scope growth (extra features), technical issues with authentication/hosting, limited testing time, low user adoption after launch.
Approval (name, date, signature)	Sponsor name: _____ Date: _____ Signature: _____

Learning objectives

By the end of this section, you can:

– explain the purpose of a project charter;
– draft a short charter that defines goal, scope, stakeholders, and constraints;
– set a clear baseline for planning and approvals.

Practical tasks:

1. Draft a charter for the chosen project. Keep it short (one to two pages).

Table 2.2 - Template of project charter

Project title	
Sponsor	
Project Manager	
Business need / problem	
Project objectives (3)	

High-level scope (in / out)	
Major deliverables	
High-level milestones (dates or weeks)	
Constraints and assumptions	
High-level risks	
Approval (name, date, signature)	

Checklist:

- Objectives are measurable and aligned with the business need.
- Scope includes out-of-scope items.
- Milestones are realistic for a 15-week timeline.
- Sponsor and PM are clearly named.

2.2 Stakeholder register and analysis

Every project involves people and groups who can affect the project or feel its results. These people are called stakeholders. In an IT project, stakeholders often include the sponsor, users, the project team, IT support staff, and university management. Some stakeholders make decisions, some provide requirements, and some use the final product. If the project team ignores key stakeholders, the project may face delays, conflicts, or rejection during acceptance.

A stakeholder register is the first document that helps organize this information. It is a structured list of stakeholders with basic details such as role, expectations, influence level, and preferred communication method. The register helps the project manager understand who must be informed, who must approve key decisions, and who needs regular updates.

Stakeholder analysis goes one step further. It explains how each stakeholder can influence the project and what engagement approach is required. A simple method uses two factors: power and interest. Power shows how much authority or influence the stakeholder has. Interest shows how much the stakeholder cares about the project results. High-power stakeholders often require frequent communication and early involvement in decisions. High-interest stakeholders often provide valuable feedback and should stay engaged to support adoption.

For the Service Desk Portal v1.0, stakeholder analysis is important because the portal affects both users and IT support staff. Students and staff want a simple way to submit requests and track progress. Support agents want a clear workflow and stable system performance. The sponsor wants better control, transparency, and reports. By documenting stakeholders early, the team can plan communication, reduce misunderstandings, and gain faster approval during the project.

Table 2.3 - stakeholder register example for Service Desk Web Portal v1.0

Name/Role	Group	Needs/ expectations	Power (H/M/L)	Interest (H/M/L)	Preferred communication	Engagement action
Head of IT Department (Sponsor)	Sponsor	On-time delivery, visibility, reports	H	H	Weekly meeting and email	Approve charter, major changes, final acceptance
Service Desk Manager	Business owner	Clear workflow, faster ticket handling	H	H	Weekly meeting	Approve requirements and process rules
Support agent (Level 1)	End user	Simple interface, fast updates	M	H	Team chat and demo	Give feedback during demos and testing
Student representative	End user	Easy ticket submission, status tracking	L	H	Survey and demo	Join usability testing (UAT)
Staff representative	End user	Notifications, transparency	L	H	Email and survey	Confirm expectations and acceptance needs
IT System Administrator	Technical	Secure hosting, backups, stability	M	M	Direct meeting	Confirm deployment and infrastructure needs
Information security officer	Compliance	Access control, data protection	H	M	Formal review	Review security requirements and risks

Project team (PM/Dev/QA/UX)	Project team	Clear scope, priorities, quality	M	H	Daily sync and tracker	Deliver work and update status weekly

How to rate stakeholders (Power and Interest)

1) Power (H/M/L) shows how much authority or influence a stakeholder has over project decisions, budget, scope, and approvals.

- H (High): can approve, reject, or stop the project (sponsor, IT leadership).
- M (Medium): influences key work and decisions within the team (service desk manager, system admin).
- L (Low): has limited decision power but may still be affected by results (students, staff users).

2) Interest (H/M/L) shows how strongly a stakeholder cares about project outcomes and how actively they will follow progress.

- H (High): actively involved and strongly affected by results (agents, key users).
- M (Medium): interested in results but not involved daily (support functions).
- L (Low): minimal involvement and limited impact on daily work (some external stakeholders).

Learning objectives

By the end of this section, you can:

- identify and document project stakeholders and their interests;
- assess stakeholder influence and priority;
- plan simple engagement actions for key stakeholders.

Practical tasks:

1. Complete the stakeholder register for the chosen project.
2. Select high-power stakeholders and write one engagement action for each.

Table 2.4 - Template of stakeholder register

Name/ Role	Group	Needs/ expectations	Power (H/M/L)	Interest (H/M/L)	Preferred communication	Engagement action

Checklist:

- Registers include users, agents, IT leadership, and security/operations.
- Engagement actions are specific (meeting, demo, approval step).
- Communication channels match the stakeholder type.

2.3 Project objectives, success criteria, and assumptions

This section helps you define what the project must achieve and how success will be measured. Clear objectives and success criteria make the project easier to plan, control, and evaluate. Assumptions help the team understand which conditions must remain true for the plan to work.

Project objectives describe the expected outcomes of the project. They explain what the project will deliver and what value it will provide. Objectives must be clear, realistic, and measurable. Avoid writing objectives as tasks or activities. Instead of "create a website," write "deliver a portal that supports ticket submission and tracking."

Success criteria show how you will measure whether each objective is achieved. Good success criteria include numbers, deadlines, or acceptance conditions. They help the team confirm results during testing and final approval. Success criteria also reduce misunderstandings between the sponsor, users, and the project team.

Assumptions describe conditions that the team expects to be true during the project. They are important because they affect planning decisions. For example, a project may assume that stakeholders will provide feedback on time, or that hosting infrastructure is available. When an assumption becomes false, it can cause delays, extra costs, or scope changes. For this reason, assumptions should be written clearly and reviewed regularly. If an assumption looks uncertain, treat it as a potential risk and plan a response.

By completing this section, you will create a clear foundation for project planning and for evaluating project success at closure.

Table 2.5 - Example of objectives and success criteria for Service Desk Web Portal v1.0

Objective	Success criterion (measurable)	How to measure (source)
Deliver a working Service Desk portal for students and staff	Portal goes live by Week 10	Deployment record and sponsor approval sign-off
Improve ticket processing transparency and response control	≥ 90% of tickets have correct status and assigned agent during UAT	UAT checklist and ticket system logs
Ensure quality and usability for end users	≥ 90% of acceptance test cases pass and average ticket creation time is ≤ 2 minutes	Test report and usability test results (time measurement)

Learning objectives

By the end of this section, you can:

- write clear project objectives and link them to success criteria;
- identify key project assumptions;
- transform assumptions into potential risks.

Practical tasks:

1. Write three project objectives and match one success criterion to each objective.

2. Write five assumptions and convert two assumptions into risks (write what can go wrong).

Table 2.6 - Template of objectives and success criteria

Objective	Success criterion (measurable)	How to measure (source)

Checklist:

- Objectives are outcomes, not activities.
- Success criteria have a clear unit (%, days, number).
- Assumptions are realistic and linked to risks.

CHAPTER 3. SCOPE PLANNING

3.1 Scope statement and requirements list

The project scope defines the boundaries, objectives, and deliverables of a project. It is a comprehensive description of what the project will accomplish, including the specific tasks, features, and functions required to achieve the desired outcomes. Clearly defining the project scope is essential to avoid scope creep, which occurs when uncontrolled changes or continuous growth in a project's scope led to delays, budget overruns, or resource strain.

Key components of project scope:

1. Project Objectives
    - High-level goals that the project aims to achieve, aligned with the organization's overall strategy.
    - Example: develop an e-commerce website with secure payment integration.
2. Deliverables
    - Tangible outputs or products resulting from the project. Deliverables must be specific, measurable, and verifiable.
    - Example: a functioning web application, user manuals, and a completed testing report.
3. Inclusions and Exclusions
    - Defines what is explicitly included in the project and, equally importantly, what is not.
    - Example: inclusion - website design and development; Exclusion: Ongoing site maintenance post-launch.
4. Constraints
    - Factors that limit the project, such as budget, time, resources, or technical specifications.
    - Example: the project must be completed within six months and not exceed a $50,000 budget.
5. Assumptions
    - Conditions or factors assumed to be true for planning purposes but may change during execution.
    - Example: assuming all team members will be available full-time for the duration of the project.
6. Stakeholder Requirements
    - Needs and expectations of stakeholders, documented during the initiation phase.
    - Example: the client requires a mobile-friendly interface and integration with existing databases.

Importance of project scope:

- Provides clarity because the project team and stakeholders understand what the project will deliver and how.

- Aligns expectations because all stakeholders share the same understanding of project objectives and deliverables.
- Guides decisions because the scope serves as a reference for evaluation of change requests or resolution of disputes.
- Controls scope creep because the scope prevents unauthorized additions or changes to the project.

Scope planning defines what the project will deliver and what it will not deliver. It helps the team avoid misunderstandings and prevents uncontrolled changes later. A clear scope also supports realistic schedule and cost planning, because the team can estimate work based on defined deliverables.

A scope statement describes the project boundaries at a high level. It explains the project purpose, the main deliverables, and the acceptance conditions. It also lists what is in scope and what is out of scope. This distinction is important in IT projects, because stakeholders often request extra features during development. A well-written scope statement protects the team from scope creep and supports clear decisions when change requests appear.

A requirements list translates stakeholder needs into specific, testable statements. Each requirement should describe one need in clear wording and avoid vague terms such as “fast” or “easy” without a measurable definition. Requirements can be functional (what the system must do) or non-functional (how the system must perform, such as security, reliability, usability, and response time). A good requirements list includes priorities, a source (who requested it), and an idea for acceptance testing. This structure helps the team confirm completion during testing and final acceptance.

For the Service Desk Web Portal v1.0, the scope statement will define the portal features for users, agents, and administrators, and it will confirm what the first release will not include. The requirements list will then capture detailed needs such as ticket submission, status tracking, notifications, role-based access, and basic reporting.

Table 3.1 - Example of scope statement for Service Desk Web Portal v1.0

Field	Sample content
Project purpose	Provide a unified web portal for students and staff to submit and track IT support tickets and improve Service Desk transparency.
In scope	User login (university accounts), ticket submission form, ticket status tracking, user dashboard, agent panel (assign/update tickets), admin settings (categories, priorities), email notifications, basic reports.
Out of scope	Mobile application, integrations with ERP/LMS, AI/ML-based ticket routing, multi-language interface (optional), advanced analytics dashboards.
Deliverables	Web portal (UI and backend), database, admin panel, notification module, basic reporting page, short user guide, deployment package.
Acceptance criteria (high-level)	Portal is deployed and accessible; core features work as expected; ≥ 90% acceptance tests pass; sponsor confirms final acceptance.

Constraints	10-week schedule, limited team capacity, fixed budget limit approved by sponsor.

Table 3.2 - Example of requirements list for Service Desk Web Portal v1.0

ID	Requirement	Type (F/NF)	Priority (H/M/L)	Source	Acceptance test idea
R1	The system allow users to log in using university credentials.	F	H	Sponsor / IT admin	Log in with a valid account and verify access.
R2	The system shall allow users to create a new ticket with category, subject, and description.	F	H	End users	Create a ticket and verify it appears in “My tickets”.
R3	The system shall assign a unique ticket ID after submission.	F	H	Service Desk manager	Submit a ticket and verify a unique ID is generated.
R4	The system shall allow users to view ticket status (New/In progress/Resolved/Closed).	F	H	End users	Change status as agent and verify user view updates.
R5	The system allow users to add comments to an existing ticket.	F	M	End users	Add comment and verify it appears in ticket history.
R6	The system shall allow agents to assign tickets to themselves or other agents.	F	H	Support agents	Reassign a ticket and verify assigned agent changes.
R7	The system shall allow agents to set ticket priority (Low/Medium/High).	F	H	Service Desk manager	Update priority and verify it appears in ticket details.
R8	The system shall allow admins to manage ticket categories (add/edit/remove).	F	M	Admin / Sponsor	Add a category and verify it appears in the user form.
R9	The system should send an email notification when a ticket is created or updated.	F	M	End users	Update ticket and confirm notification email is received.
R10	The system shall provide basic reports (open tickets,	F	M	Sponsor	Open report page and verify

	closed tickets, response time).				correct counts and values.
R11	The system shall restrict agent and admin functions based on user role.	NF	H	Security officer	Log in as user and confirm agent/admin pages are blocked.
R12	The system shall load the user dashboard within 3 seconds under normal conditions.	NF	M	Sponsor	Measure dashboard load time during test session.
R13	The system shall store ticket data securely and support regular backups.	NF	H	IT admin	Verify backup procedure and restore a sample ticket record.
R14	The system shall log key actions (ticket created, status change, assignment).	NF	M	Service Desk manager	Check log records after performing key actions.
R15	The system should support at least 200 active users without system failure.	NF	M	Sponsor / IT admin	Run a basic load test and confirm stable operation.

1) Type (F/NF)

- F (Functional): what the system must do (features and actions). Example: “Users can create a ticket.”
- NF (Non-functional): how the system must work (quality, performance, security, usability). Example: “Dashboard loads within 3 seconds.”

2) Priority (H/M/L)

- H (High): mandatory for the first release (project cannot succeed without it).
- M (Medium): important, but the project can still launch without it if time is limited.
- L (Low): optional or “nice to have” (can move to a future version).

Learning objectives

By the end of this section, you can:

– define project scope using in-scope and out-of-scope statements;
– distinguish between functional and non-functional requirements;
– document clear requirements that support scope control.

Practical tasks:

1. Write a statement for the portal (in-scope and out-of-scope).

2. Create requirements list with at least 15 requirements (mix of functional and non-functional).

Table 3.3 - Template of scope statement

Project purpose	
In scope	
Out of scope	
Deliverables	
Acceptance criteria (high-level)	
Constraints	

Table 3.4 - Template of requirements list

ID	Require-ment	Type (F/NF)	Priority (H/M/L)	Source	Acceptance test idea

Checklist:

- In-scope and out-of-scope are clear.
- Requirements use clear language and support testing.
- Each requirement has priority and a source.

3.2 Work Breakdown Structure (WBS)

A work breakdown structure (WBS) is a visual diagram that organizes project tasks, showing their connections to one another and their place within the overall project.

The Work Breakdown Structure (WBS) is a visual tool that maps out all project activities and their hierarchical relationships to the overall goal (see Figure 3.1). By presenting tasks in a result-oriented, tree-like format or a structured list, it allows managers to model different scenarios and forecast outcomes. In essence, teams use the WBS to deconstruct a complex project into smaller, more manageable units. This decomposition continues until an individual team member can take responsibility for each distinct component.

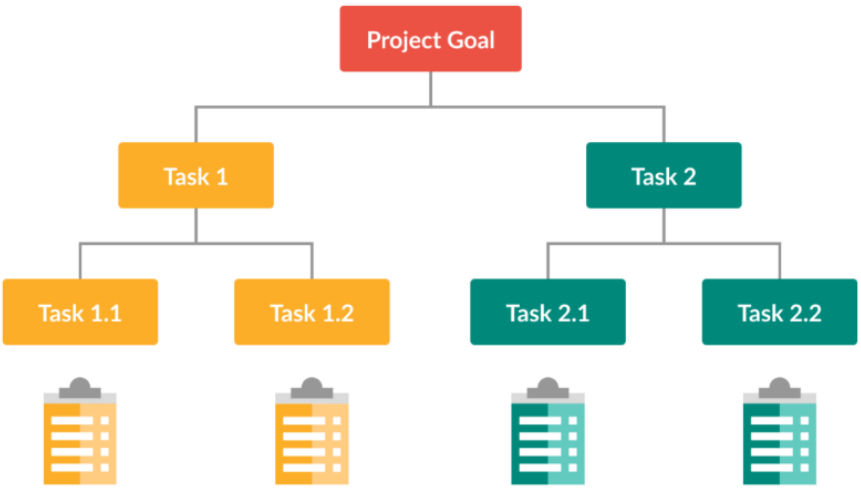


Figure 3.1 – WBS illustration

Consider a scenario in which a team develops a new mobile app. The primary level one project objective is to "launch a new mobile app." This main objective depends on three core level two elements: the creation of a database, the development of back-end logic, and the design of the user interface. For the user-interface element, level three tasks include a creative brainstorming session, the creation of a mood board, and the construction of wireframes.

Simple projects with only a few activities can often be managed by a single person who understands the overall structure with ease. However, projects that require formal planning usually involve dozens, hundreds, or even thousands of activities. As project size increases, so do the number of activities and the level of detail that managers must address.

a) Work Breakdown Structure (WBS) - "What"

When projects involve many tasks, it is important to organize them so that the plan clearly connects with the information needed by executors. A Work Breakdown Structure (WBS) provides a systematic way to decompose complex projects into smaller work packages and tasks. Some organizations develop standard work packages for similar projects and assign codes to track costs and schedules. These codes may also indicate factors such as material type or location.

The deeper the WBS, the more detailed the decomposition becomes. The lowest level typically represents individual tasks, which provide the detail needed for effective control and execution. The level of detail depends on the needs of managers and executors: managers focus on overall budget and deadlines, while executors require task-specific instructions.

Three main types of WBS are commonly used: Project WBS, Standard WBS, and Contract WBS [7]. Among them, Project WBS is the most practical tool for monitoring and controlling project work (see Figure 3.2)

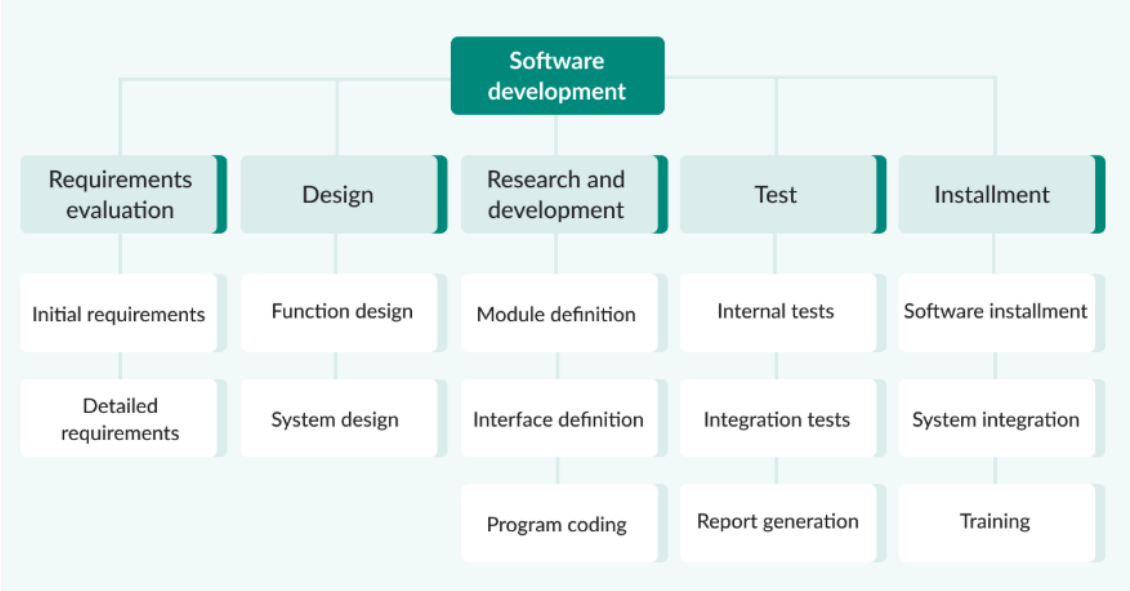


Figure 3.2 - Example of a WBS for a Software Development Project.

Typically, a Work Breakdown Structure (WBS) is derived from activities performed in past projects of a similar nature [10]. A WBS from a previous project can serve as a template for a new one. Figure 3.3 presents the top levels of a sample WBS template designed for the development of a client website.

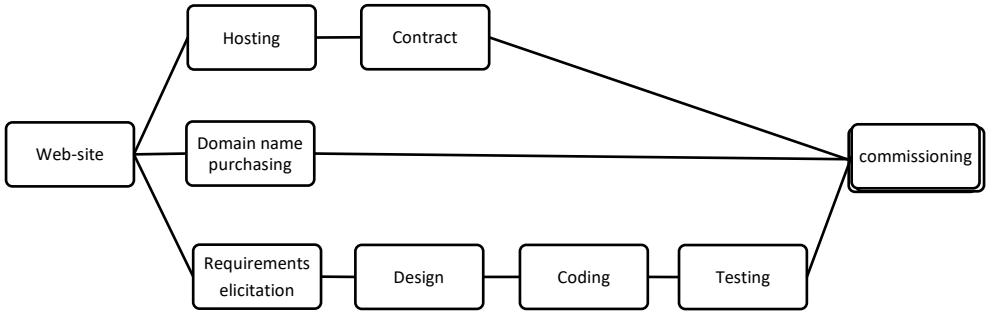


Figure 3.3 - Example of a WBS for a Client Website Development Project

The contract is then finalized between the client and the contractor. It must include a breakdown of the work scope into key elements that will serve to measure progress, control the project, and determine the contract price. This breakdown may be less detailed than the project WBS.

In summary, a WBS represents a result-oriented decomposition of the project scope into sufficient detail to ensure clear identification of all information required for executing and managing tasks. A common guideline for defining the appropriate level of detail is the "8-80 rule," which recommends assigning between 8 and 80 working hours to a lower-level task. Tasks requiring less than 8 hours risk excessive micromanagement, while those exceeding 80 hours should be decomposed further.

b) Organizational Breakdown Structure (OBS) - "Who"

After determining what needs to be done, it is necessary to identify all human resources required to complete the project. Depending on the scope of work, the project may require engineering skills, procurement capabilities, developer (executor) labor,

managerial staff, and more. The Organizational Breakdown Structure (OBS) is a practical method for decomposing the human resource requirements for all areas of expertise and then into project roles, regardless of the number of participants. It assigns specific roles to individuals [11].

Project Resources

Internal Resources:

- Project Management Team
  - Project Manager
  - Development Department Manager
  - Task Scheduler
- Design Engineering
  - Design Engineer
  - Network Architect
  - Security Engineer
- Development Engineering
  - Lead Specialist
  - Developers
  - Testers

External Resources:

- Subcontractors
  - Hosting
  - Developers
  - Etc.
- Material Suppliers
  - Active Equipment
  - Software and Licensing Providers

c) Cost Breakdown Structure (CBS) - "How Much"

After defining "what" will be done through the WBS and "who" will perform it through the OBS, the next question is "how much" it will cost. This is addressed through the Cost Breakdown Structure (CBS).

CBS is a hierarchical system that divides project costs into elements, functions, and categories. It classifies resources by cost accounts, typically including labor, materials, and other direct expenses. The CBS provides an economic breakdown of the project into budgets for work packages, allowing managers to track progress and expenses against the planned scope and responsibilities.

Direct costs in the CBS include labor, materials, equipment, project management, design services, licenses, and risk management. Indirect company overheads not tied to the project, such as utilities or taxes, are excluded.

There are two common approaches to structuring the CBS:

- WBS-based CBS (Activity-Based Costing, ABC)

The CBS mirrors the WBS, with each cost account tied to a work package or task. This allows detailed budgeting, accounting, and tracking. Costs at the lowest WBS level may include labor, materials, equipment, and subcontracting.

- Agile-based CBS

Used in corporate Agile projects, this approach estimates costs based on user stories and tasks. Techniques like Scrum Poker or Wideband Delphi are applied to estimate effort in terms of hours, time, and money. The total cost is then calculated, with an added buffer (e.g., 30%) and material or resource costs.

An example of how user stories are presented is shown in Table 3.5 [12].

Table 3.5 - Example of a User Story

Background: The HR department manager intends to create an automated testing system for conducting interviews with candidates applying for open positions in the company.
As a manager, I want to review my existing tests so that I can recall what I currently have available and determine whether I can simply reuse or update an existing test for the position I currently need to fill.
As an HR manager, I want to match the required skills for an open position with test topics so that I can create a test suitable for screening candidates.
As an HR manager, I want to send a draft test to the functional manager to ensure that I have covered the correct topics in the review.
As a functional manager, I want to send feedback on the test draft to the HR manager to ensure that the candidate evaluation process is as effective as possible.
As an HR manager, I need to purchase an upgraded service level for the system so that I can add additional topics to my tests.
As an HR manager, I want to add additional questions to the test so that we can cover extra topics that are important to the functional manager.
As an HR manager, I want to trial the screening test to ensure it works as expected and that I am ready to administer it to candidates and share the results with the functional manager.

Despite the differences in approach, when applying Agile, the final CBS just as in the traditional method is ultimately derived by multiplying the cost of an individual module by the number of complete modules. In practice, the executor's internal CBS, often used for cost accounting, usually becomes the basis for the contract. The key challenge lies in ensuring accurate cost accounting when functional modules overlap.

In software development projects, the contract WBS is typically prepared by the executor and therefore matches the project WBS. The total value of the contract equals the execution costs plus the agreed price, which is paid progressively as the project advances. Since revenue directly depends on cost, the project WBS must align with the CBS whenever corporate cost control is required. In such cases, it is advisable to begin with a higher-level CBS and then decompose it further according to the project's work requirements.

For the Service Desk Web Portal v1.0, the WBS can include deliverables such as authentication, ticket submission, user dashboard, agent panel, admin settings, notifications, reporting, testing, and deployment. The WBS dictionary then defines what "done" means for each work package.

WBS outline for Service Desk Web Portal v1.0:

1.1 Project management and control
1.1.1 Project charter and baseline documents
1.1.2 Status reporting and stakeholder communication
1.1.3 Issue, risk, and change control logs

1.2 Requirements and design
1.2.1 Requirements list and scope baseline
1.2.2 UX wireframes and UI prototype
1.2.3 System architecture and database schema

1.3 Core user functions (student/staff)
1.3.1 User authentication and role access
1.3.2 Ticket creation form
1.3.3 User dashboard (ticket list and status)
1.3.4 Ticket details page (view history and comments)

1.4 Service desk agent functions
1.4.1 Agent login and permissions
1.4.2 Ticket assignment and queue view
1.4.3 Status update and resolution workflow
1.4.4 Internal notes and communication (agent-side)

1.5 Admin configuration
1.5.1 Category management (add/edit/remove)
1.5.2 Priority and SLA rules (basic)
1.5.3 User and role management (basic admin control)

1.6 Notifications and reporting
1.6.1 Email notifications (ticket created/updated)
1.6.2 Basic reporting dashboard (open/closed tickets)
1.6.3 Response time summary report (basic)

1.7 Quality assurance and release
1.7.1 Test plan and acceptance test cases
1.7.2 System testing results and defect log
1.7.3 User acceptance testing (UAT) package

1.8 Deployment and handover

1.8.1 Production deployment package
1.8.2 User guide (short) and admin guide (short)
1.8.3 Project closure and lessons learned

Table 3.6 - WBS dictionary for Service Desk Web Portal v1.0

WBS ID	Work package	Description	Owner	Acceptance criteria	Notes/assumptions
1.3.1	User authentication and role access	Login for students/staff using university credentials and basic role-based access (user/agent/admin).	Developer and IT admin	Users can log in successfully; role restrictions work; unauthorized pages are blocked.	University accounts and access method are available.
1.3.2	Ticket creation form	Form to submit a new ticket with category, subject, description, and optional attachment.	Developer and UX	Ticket can be submitted; ticket ID is created; data is stored in the database; confirmation message appears.	Attachment size limit defined (example: 5 MB).
1.4.2	Ticket assignment and queue view	Agent view to see new tickets and assign them to an agent (self or other).	Developer and Service Desk manager	Agent can see ticket queue; assignment updates correctly; assigned agent appears in ticket details.	Ticket status categories are approved during planning.
1.6.1	Email notifications	Automatic email alerts for ticket creation and key updates (status change, comment, resolution).	Developer	Emails are sent for defined events; message content includes ticket ID and status; delivery tested successfully.	Email server/service is available for integration.
1.8.1	Production deployment package	Deployment-ready version with configuration files and basic setup instructions.	IT admin and PM	Portal is deployed in production; users can access it; basic functions pass smoke testing.	Hosting environment and domain access are provided by IT.

More examples of WBS:

1. Qadam app – core gameplay mechanics for territory capture, resource management, and player interaction (Figure 3.4). Presented by SDU students during the PM class.

2. Accounting service (Figure 3.5).

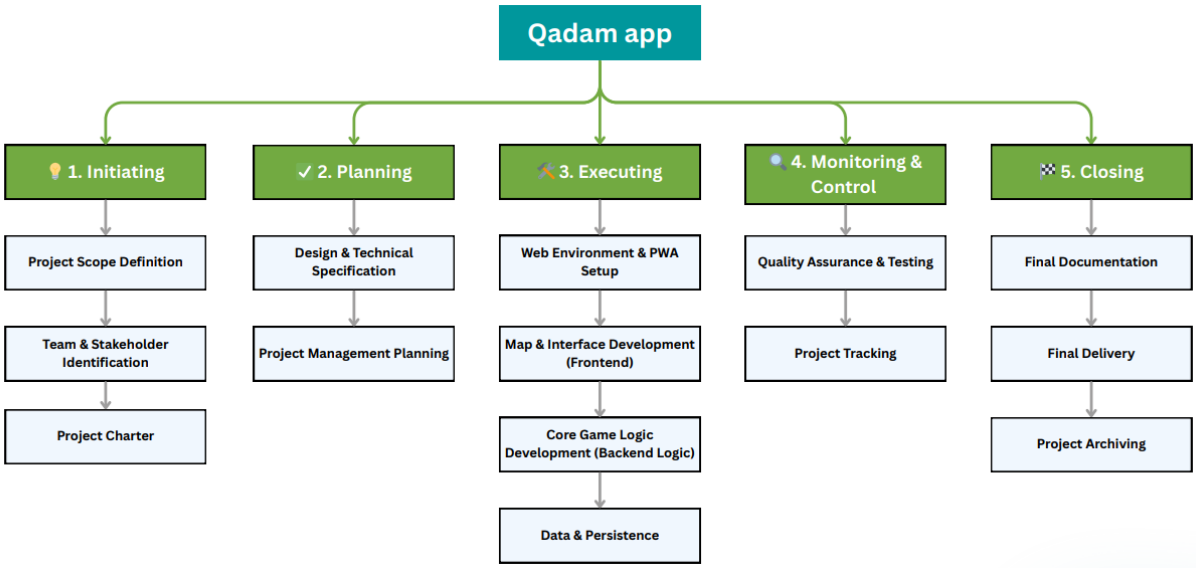


Figure 3.4 – Qadam app's WBS

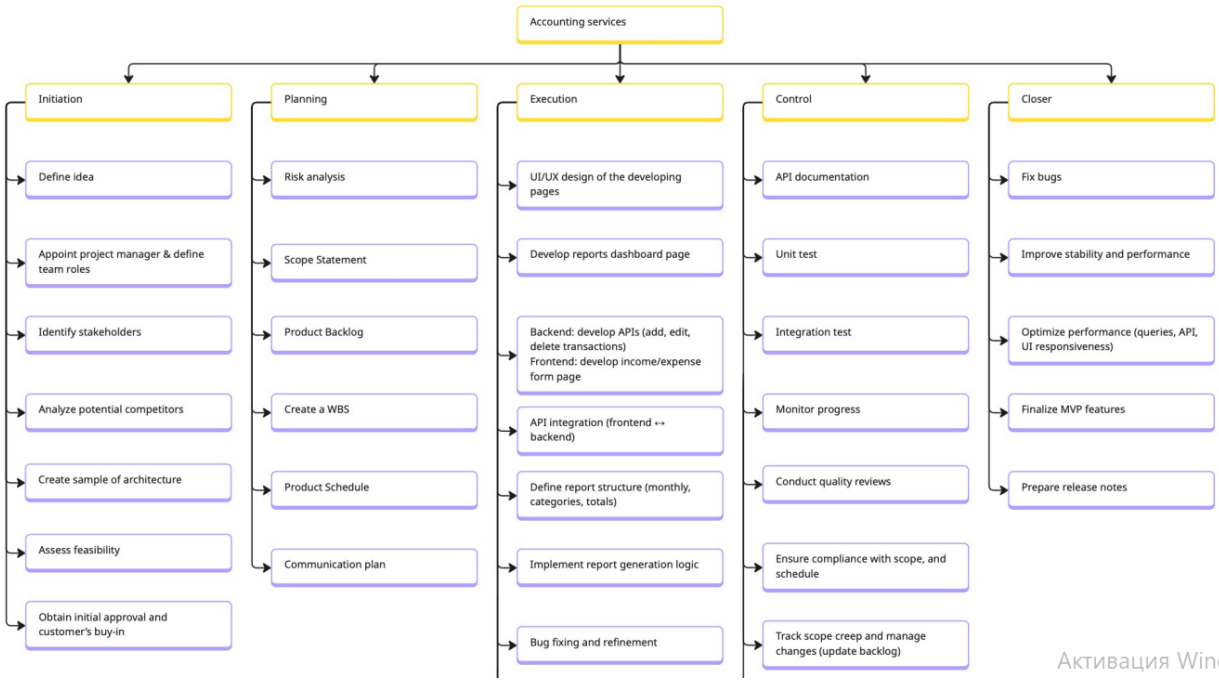


Figure 3.5 – Accounting service's WBS

Learning objectives

By the end of this section, you can:

– decompose project scope into a structured Work Breakdown Structure (WBS);
– define work packages at an appropriate level of detail;
– document work package details using a WBS dictionary.

Practical tasks:

1. Create a WBS with three levels (at least 20 work packages).
2. Fill out a WBS dictionary for five work packages.

Table 3.7 - Template of WBS dictionary

WBS ID	Work package	Description	Owner	Acceptance criteria	Notes/assumptions

3.3 Scope baseline and change rules

After defining scope, the project must protect it. The scope baseline is the official reference that describes what the project will deliver. It is used to control project work and to prevent scope creep. The scope baseline includes three documents: the scope statement, the WBS, and the WBS dictionary. Together, they create a clear and measurable definition of the approved project scope (Figure 3.6).

The scope statement explains the boundaries of the project and lists what is included and excluded. The WBS breaks the approved scope into deliverable-based components, and the WBS dictionary adds detailed descriptions and acceptance criteria for work packages. Once these documents are approved, they become the baseline. This means the team must compare all future work against these documents to confirm that the project stays aligned.

In real projects, stakeholders often request changes after planning starts. A new feature request, an extra report, or a change in workflow may affect scope, schedule, or cost. For this reason, the project needs change rules. Change rules define how the team submits, reviews, approves, and implements change requests. A clear change process protects the project from uncontrolled work and supports fair decision-making.

A typical change process includes four steps. First, a stakeholder submits a change request with a clear description and reason. Second, the project manager evaluates the impact on scope, schedule, cost, resources, and risk. Third, the sponsor or approval authority reviews the analysis and makes a decision to approve, reject, or defer the request. Finally, if the change is approved, the team updates the project documents, implements the change, and tracks it in the change log. If the change is not approved, the project continues with the baseline scope.

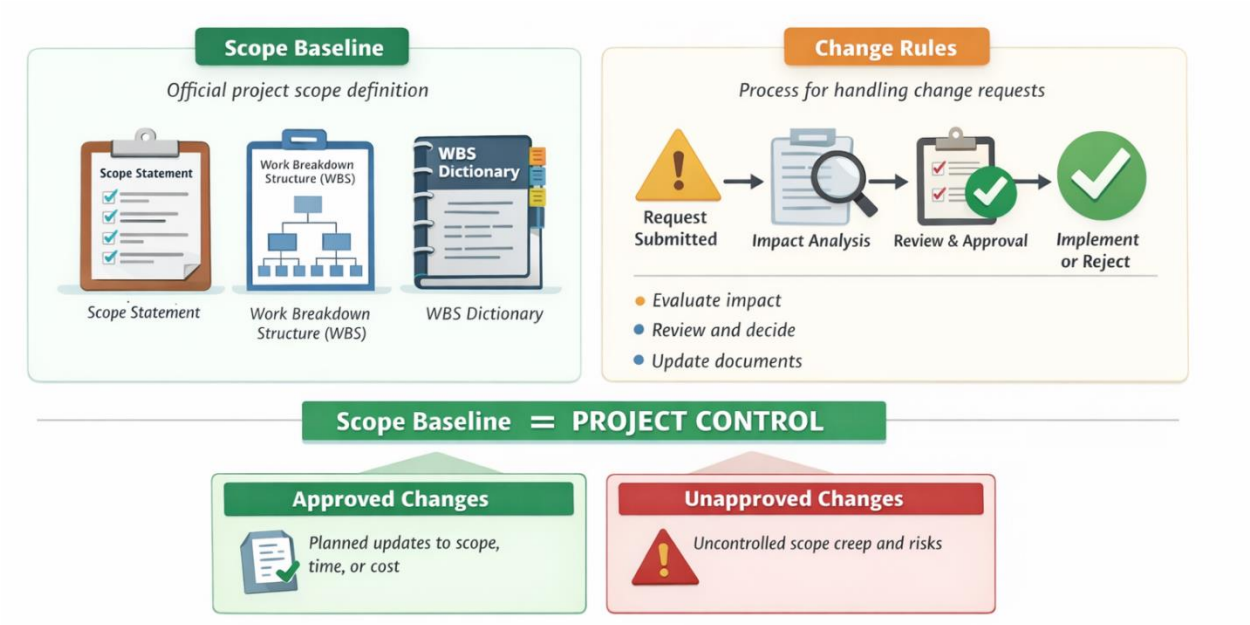


Figure 3.6 - Scope baseline and change control flow

For the Service Desk Web Portal v1.0, change rules are especially important because stakeholders may request additional features such as mobile access, integrations, or advanced reporting (example shown in Table 3.8). The change process allows the sponsor to decide whether the project should include these items now, postpone them to a future release, or reject them to protect the 10-week timeline.

Table 3.8 – Change rules for the Service Desk Web Portal v1.0 example

Field	Sample content
Change request channel	Change request form (shared project folder) and submission via email to the project manager
Required information	Requester name/role; date; change description; reason; priority; affected deliverable/WBS ID; deadline (if any)
Impact analysis items	Scope impact; schedule impact (days/weeks); cost impact; resource impact; risk impact; quality/testing impact
Approval authority	Sponsor approves changes that affect scope, schedule baseline, or budget; PM may approve minor changes with no baseline impact
Decision time limit	3 working days after submission (urgent changes: 1 working day)
Versioning rule	Update affected documents with version number and date (v1.0 to v1.1); record the decision in the change log; keep old versions in the archive folder

Learning objectives

By the end of this section, you can:

– identify key project baselines that require control;
– explain why formal change control is necessary;
– define clear rules for requesting and approving changes.

Practical tasks:

1. List the baseline components you will control for the chosen project.

2. Write changes rules: who can request a change, who approves, and what information is required.

Table 3.9 - Change Rules

Change request channel	
Required information	
Impact analysis items	
Approval authority	
Decision time limit	
Versioning rule	

Checklist:

- Rules specify an approver and a time limit.
- Impact analysis includes scope, schedule, cost, and risk.
- Baseline documents have clear version names.

CHAPTER 4. SCHEDULE PLANNING

4.1 Activity list and dependencies

Successful time management requires a clear definition of project activities. Each activity must have a start, a finish, and measurable outputs. In IT projects, typical activities include requirement analysis, database design, module development, system testing, and deployment.

Dependencies establish the logical order of activities. Four primary dependency types exist (see Figure 4.1):

- Finish to Start (FS): Activity B starts only after Activity A finishes. Example: system design must finish before coding starts.
- Start to Start (SS): Both activities start at the same time. Example: technical documentation starts at the same moment as coding.
- Finish to Finish (FF): Two activities finish together. Example: module testing and bug fixing finish simultaneously.
- Start to Finish (SF): The start of one activity controls the finish of another. This type appears rarely.

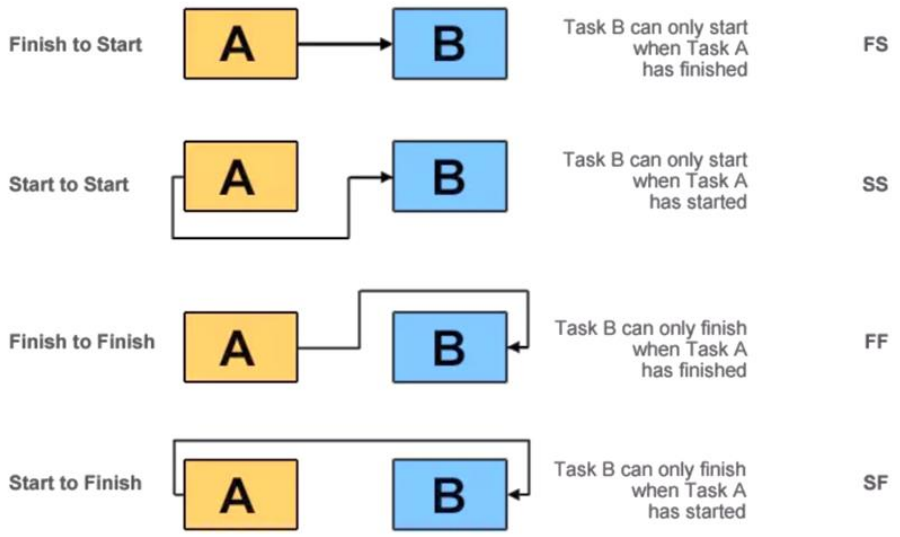


Figure 4.1 - The four types of relationship-based task dependencies

To determine task dependencies in project management, the manager must analyze the logical relations between tasks and establish which activities require completion before others may start or finish. In other words, both predecessors and successors must be considered. Several approaches support this process:

- Review of project scope and objectives: clear understanding of the overall goals allows identification of essential tasks and their required sequence.
- Development of a Work Breakdown Structure (WBS): a WBS divides the project into smaller, controllable elements. This structure reveals tasks and their connections.
- Analysis of the project schedule: the schedule highlights activities that rely on the completion of prior tasks.

- Consultation with subject matter experts (SMEs): experts contribute practical knowledge that clarifies dependencies among tasks.
- Application of project management software: such tools detect dependencies and generate a visual view of the entire schedule.

Table 4.1 - Activity list example for Service Desk Web Portal v1.0

ID	Activity name	WBS link	Predecessor(s)	Dependency type	Notes
A1	Approve project charter	1.1.1	-	-	Sponsor approval
A2	Approve requirements	1.2.1	A1	FS	Requirements sign-off
A3	Approve UI prototype	1.2.2	A2	FS	UX review completed
A4	Finalize architecture and database schema	1.2.3	A2	FS	Tech design baseline
A5	Build authentication and roles	1.3.1	A4	FS	User/agent/admin roles
A6	Build ticket creation and user dashboard	1.3.2-1.3.3	A3, A5	FS	Core user flow
A7	Build agent workflow (queue, assignment, status)	1.4.2-1.4.3	A6	FS	Service Desk rules
A8	Build admin settings and notifications and reports	1.5.1, 1.6.1-1.6.2	A7	FS	Basic v1.0 functions
A9	System testing and defect fixes	1.7.1-1.7.2	A8	FS	QA and dev fixes
A10	UAT, deployment, and closure	1.7.3, 1.8.1-1.8.3	A9	FS	Go-live and handover

Learning objectives

By the end of this section, you can:

– derive activities from a Work Breakdown Structure;
– identify logical dependencies between activities;
– prepare inputs for schedule development.

Practical tasks:

1. Create an activity list based on your WBS (at least 20 activities).
2. For each activity, define at least one dependency.

Table 4.2 - Activity list

ID	Activity name	WBS link	Dependency (ies)	Depend-ency type	Notes

Checklist:

- Activities reflect work, not deliverables.
- Each activity has at least one predecessor (except the first).
- Dependencies look realistic for software delivery.

4.2 Network Diagram (PDM)

Two main tools provide schedule representation:

- Gantt Chart. This chart places each activity on a timeline with horizontal bars. It allows a manager to see task duration, overlaps, and sequence. In an IT project, a Gantt chart can show phases such as UI design, backend coding, integration testing, and release preparation.

- Network Chart (PERT/CPM). This diagram presents activities as nodes with arrows that indicate dependencies. It supports identification of bottlenecks and critical paths. For IT projects, it shows the logical flow from requirement analysis to deployment.

A Gantt chart is a horizontal bar diagram that presents the planned start and finish dates for each task within a project. It also indicates task dependencies and key milestones. The diagram may display task progress and status with the help of different colors or symbols. A Gantt chart offers a simple and clear view of the project timeline, scope, and resource allocation. It also serves as an effective communication tool for teams and stakeholders, as it highlights expectations and responsibilities for every role.

Figure 4.2 below presents the main components and format of a Gantt chart, together with the Gantt bar symbols that represent different task types.

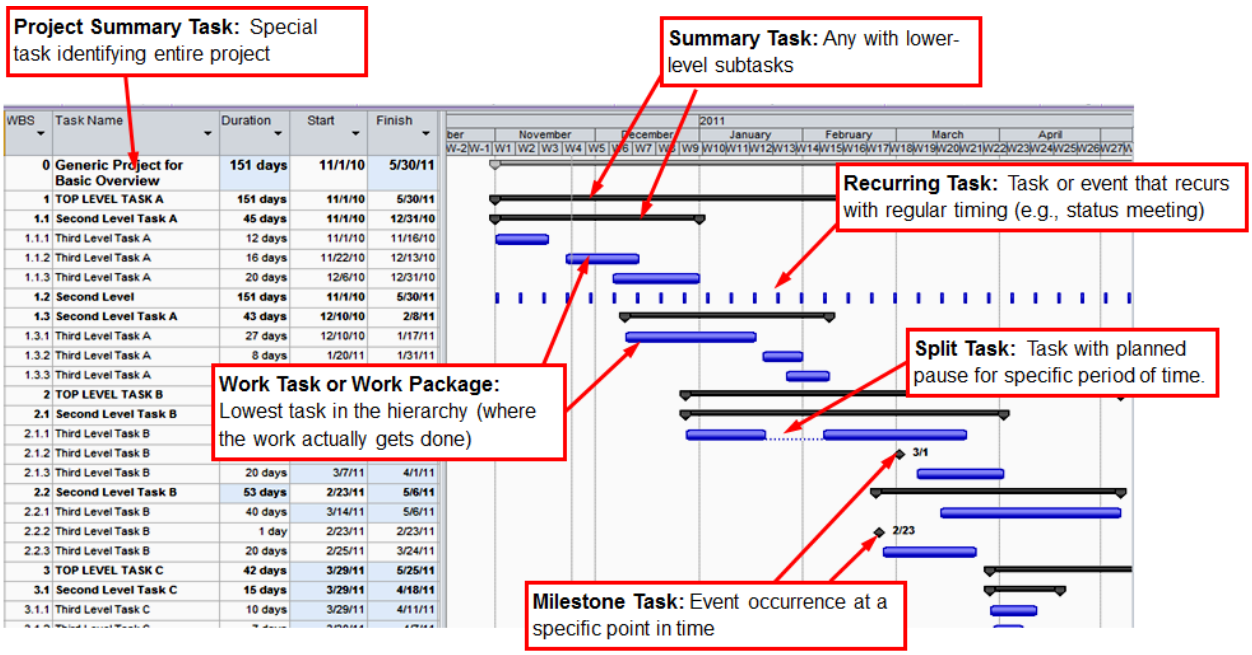


Figure 4.2 - Example of Gantt Chart

This Figure 4.2 presents an example of a project Gantt chart that shows tasks and time information. On the left side, the chart contains five columns: WBS number, task name, duration, start date, and finish date. On the right side, task bars appear along a time scale divided into years, months, and weeks, with vertical grid lines for clarity. Annotation boxes with arrows describe the main task types. The project summary task defines the entire project, while a summary task includes lower-level subtasks. A recurring task represents an event that takes place regularly, such as a status meeting. A work task or work package identifies the lowest level where actual work takes place. A split task shows a planned pause in the middle of its timeline. A milestone task marks a key event at a specific date with a diamond symbol.

A network diagram is a graphical tool that illustrates the logical relationships between project tasks. The diagram contains nodes and arrows, where nodes denote tasks and arrows denote dependencies. It displays the sequence and order of tasks, along with the critical path and the available float or slack for each task. Float or slack refers to the amount of time a task may face delay without an impact on the overall project completion date. A network diagram supports analysis and optimization of the project schedule, since it makes dependencies and task constraints visible.

What's a PERT Chart?

A PERT chart, short for Program Evaluation Review Technique, represents a graphical method that maps the tasks required to complete a project. In this chart, arrows indicate the tasks that lead to an event, while nodes symbolize the completion of project phases. Each node reflects a finished stage, and the chart serves as a tool to estimate the time needed for task completion within the project.

The clearest way to describe a PERT chart and its function is through an example. Figure 4.3 below illustrates a PERT chart designed for a business website development project. Each box in the diagram denotes a project task, while arrows indicate task dependencies and the required sequence of tasks from left to right. These arrows also show the estimated duration of each task, which determines when the subsequent

task may begin. By using a PERT chart, one can sum up the duration of all project tasks and calculate the overall project timeline. The example also highlights the critical path, which appears as the longest task sequence within the PERT chart.

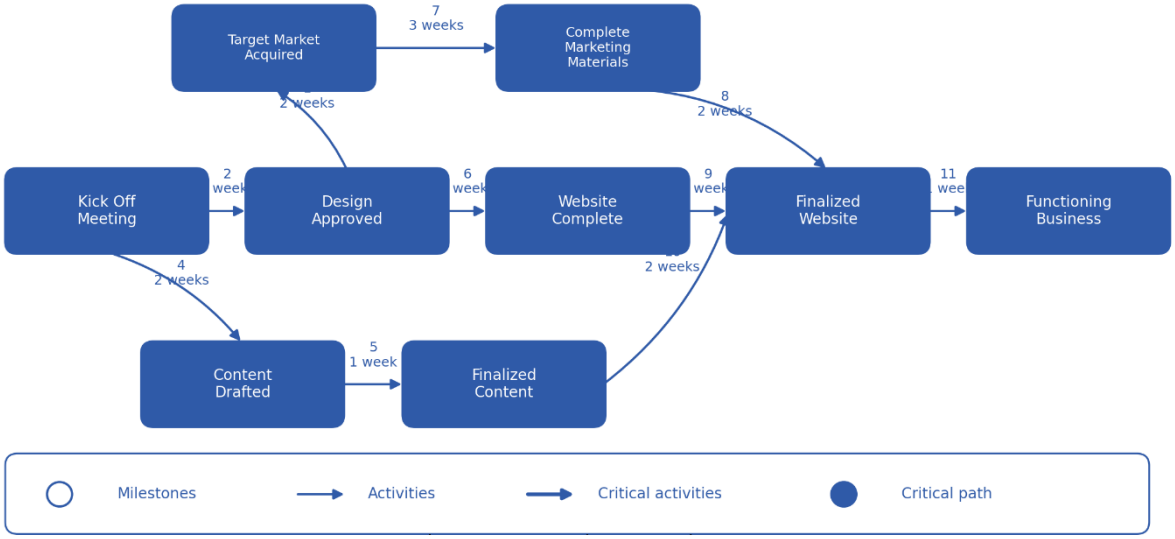
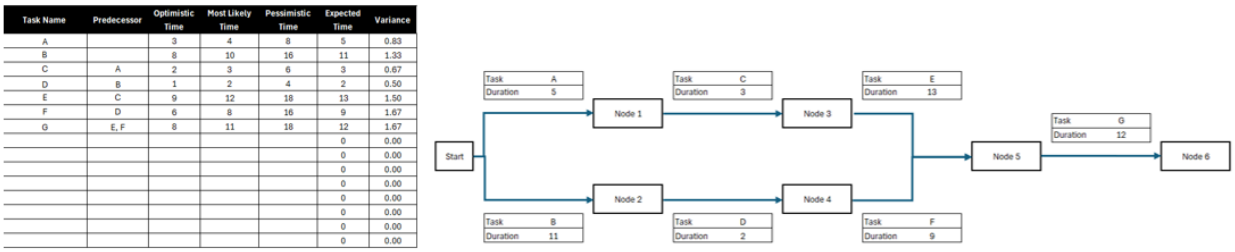


Figure 4.3 - PERT Chart Example

A PERT chart provides a clear view of the entire project timeline in a single, easy-to-read diagram that supports both team members and stakeholders. Beyond project scheduling, a PERT chart also serves as a tool to estimate the resources required for each task and ensures that those resources are allocated to activities on the critical path.



Task Name	Predecessor	Optimistic Time	Most Likely Time	Pessimistic Time	Expected Time	Variance
A		3	4	8	5	0.83
B		8	10	16	11	1.33
C	A	2	3	6	3	0.67
D	B	1	2	4	2	0.50
E	C	9	12	18	13	1.50
F	D	6	8	16	9	1.67
G	E, F	8	11	18	12	1.67
					0	0.00
					0	0.00
					0	0.00
					0	0.00
					0	0.00
					0	0.00
					0	0.00
					0	0.00

Figure 4.4 - PERT Chart Template

Gantt charts and PERT charts both help plan and organize project tasks. The main difference is their appearance: a PERT chart shows a network of tasks with arrows, while a Gantt chart shows tasks as bars on a timeline. A PERT chart makes task connections clear, so you can see which tasks must happen before others. It also shows the critical path, meaning the tasks that control how long the project will take. With this, you can see the earliest and latest times a task can start, which helps you plan better, save time, and make smarter choices. Example of PDM is shown in Figure 4.5.

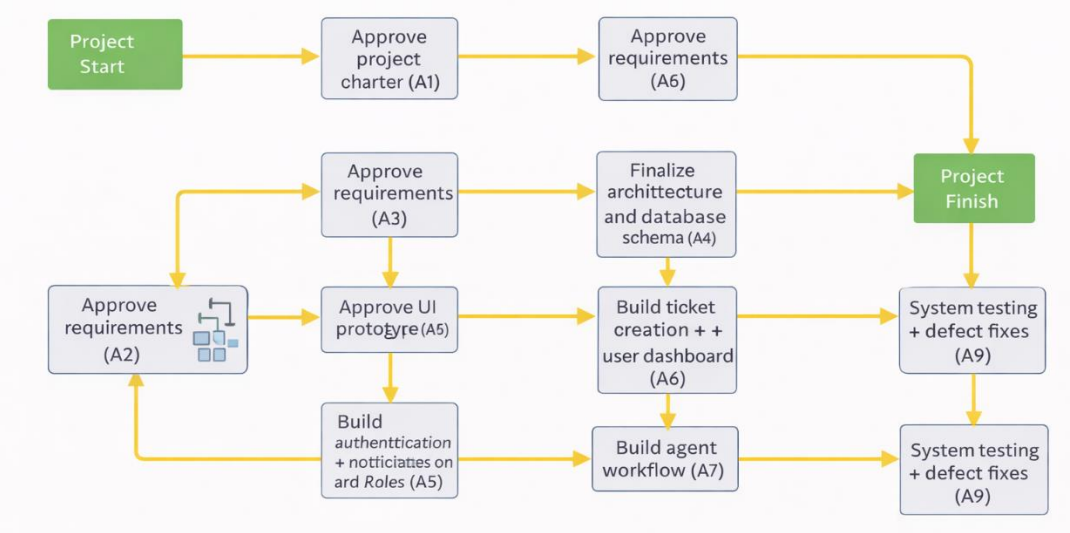


Figure 4.5 – PDM network diagram for Service Desk Portal v1.0

Learning objectives

By the end of this section, you can:

– represent activity dependencies using a Precedence Diagramming Method (PDM) network;
– identify start and finish activities in a project network;
– verify logical consistency by checking for loops and missing links.

Practical tasks:

1. Draw a PDM network for at least 10 key activities.
2. Mark the start and finish activities and verify there are no loops.

Checklist:

– Diagram has a single clear start and finish.
– No circular dependencies exist.
– Key dependencies match the activity list.

4.3 Duration estimates and resource needs

Duration estimates define how long each project activity is expected to take, while resource needs identify the people, equipment, and materials required to complete those activities. Accurate estimates help create realistic schedules, balance workloads, and avoid bottlenecks.

For Internet of Things (IoT) projects, this involves considering:

- Software development tasks – coding, testing, integration.
- Hardware tasks – prototyping, assembly, deployment.
- Team availability – developers, engineers, testers, and support staff.
- Dependencies – tasks that rely on others before starting.

Together, duration and resource planning ensures the project is achievable within time and capacity constraints, reducing risks of delays or overloading team members. Example of duration and resource need is shown in Table 4.3.

Table 4.3 - Duration and resources example for Service Desk Web Portal v1.0

Activity ID	Duration (days)	Role	Effort (hours)	Basis of estimate	Risk note
A1	1	PM	4	Approval step	Sponsor delay
A2	4	PM	16	Similar projects	Late feedback
A3	4	UX	24	Prototype screens	Extra revisions
A4	3	Dev	24	Standard setup	Env/access issues
A5	5	Dev	40	Auth module	Login integration delay
A6	10	Dev	80	Core features	Scope creep
A7	8	Dev	64	Agent workflow	Workflow changes
A8	7	Dev	56	Admin and reports	Email/report issues
A9	7	QA	56	Test cycle	High defects
A10	5	PM	16	UAT and release	UAT delays

Learning objectives

By the end of this section, you can:

- estimate activity durations;
- assign a main resource role to each activity.
- prepare inputs for schedule and resource planning.

Practical tasks:

1. Estimate duration for activities (days).
2. Assign a main resource role to each activity (PM, Dev, QA, UX).

Table 4.4 - Duration and Resources

Activity ID	Duration (days)	Role	Effort (hours)	Basis of estimate	Risk note

Checklist:

- Durations fit inside the 10-week constraint.
- Resource roles match skills needed.
- Large estimates include a short basis note.

4.4 Gantt chart and schedule baseline

A schedule baseline represents the approved version of the project schedule. It functions as a reference point for performance measurement. Without this baseline, no effective control of schedule deviations exists. In IT projects, the baseline often fixes release dates, sprint plans, and integration points.

Creating a Schedule Baseline (Quick Steps)

- List Tasks: Identify all tasks needed to reach the final deliverable using a work breakdown structure.
- Estimate Duration: Determine how long each task will take using analogous or three-point estimating.
- Set Milestones: Mark key points or phase completions in the project. Identify Dependencies: Determine which tasks rely on the completion of others.
- Assign Resources: Allocate people, materials, and tools to each task with associated costs and time.
- Establish Baseline: Freeze the finalized schedule to use as a reference for tracking progress.

Time control involves techniques and strategies to keep the project on schedule. One of the primary methods is performance measurement, which includes metrics such as schedule variance and schedule performance index. These metrics allow managers to quantify how far ahead or behind the project is relative to the baseline. The Critical Path Method (CPM) is another key approach, focusing on the sequence of dependent tasks that determine the project's minimum duration. Tasks on the critical path demand close attention, as delays directly affect the project completion date.

In addition, tracking milestones throughout the project provides reference points to monitor progress and detect potential delays early. Regular progress reporting and schedule updates are also crucial. By collecting actual performance data and updating the schedule accordingly, managers can compare results against the baseline and implement corrective measures promptly. Altogether, these time control methods, when combined with a robust schedule baseline, enable project managers to maintain control over timelines and improve the likelihood of completing projects on schedule.

Examples:

1. Application for cleaning service (Figure 4.6)
2. Application for Urban Road analytics (Figure 4.7)

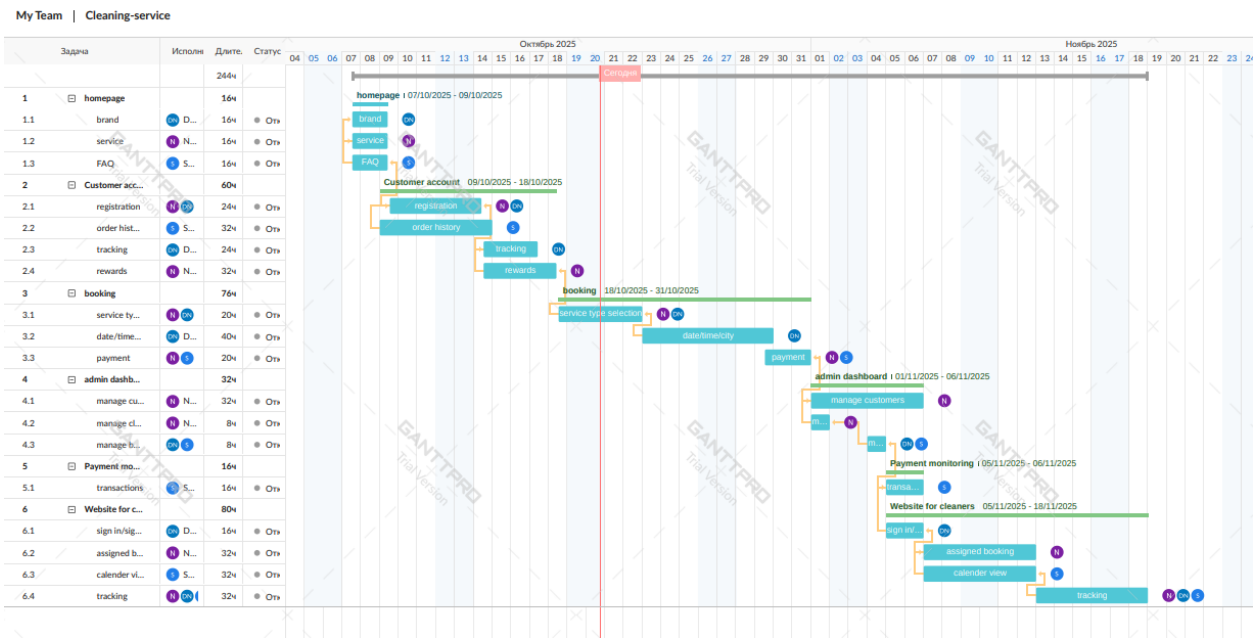


Figure 4.6 – Schedule planning in GanttPRO of a Cleaning service

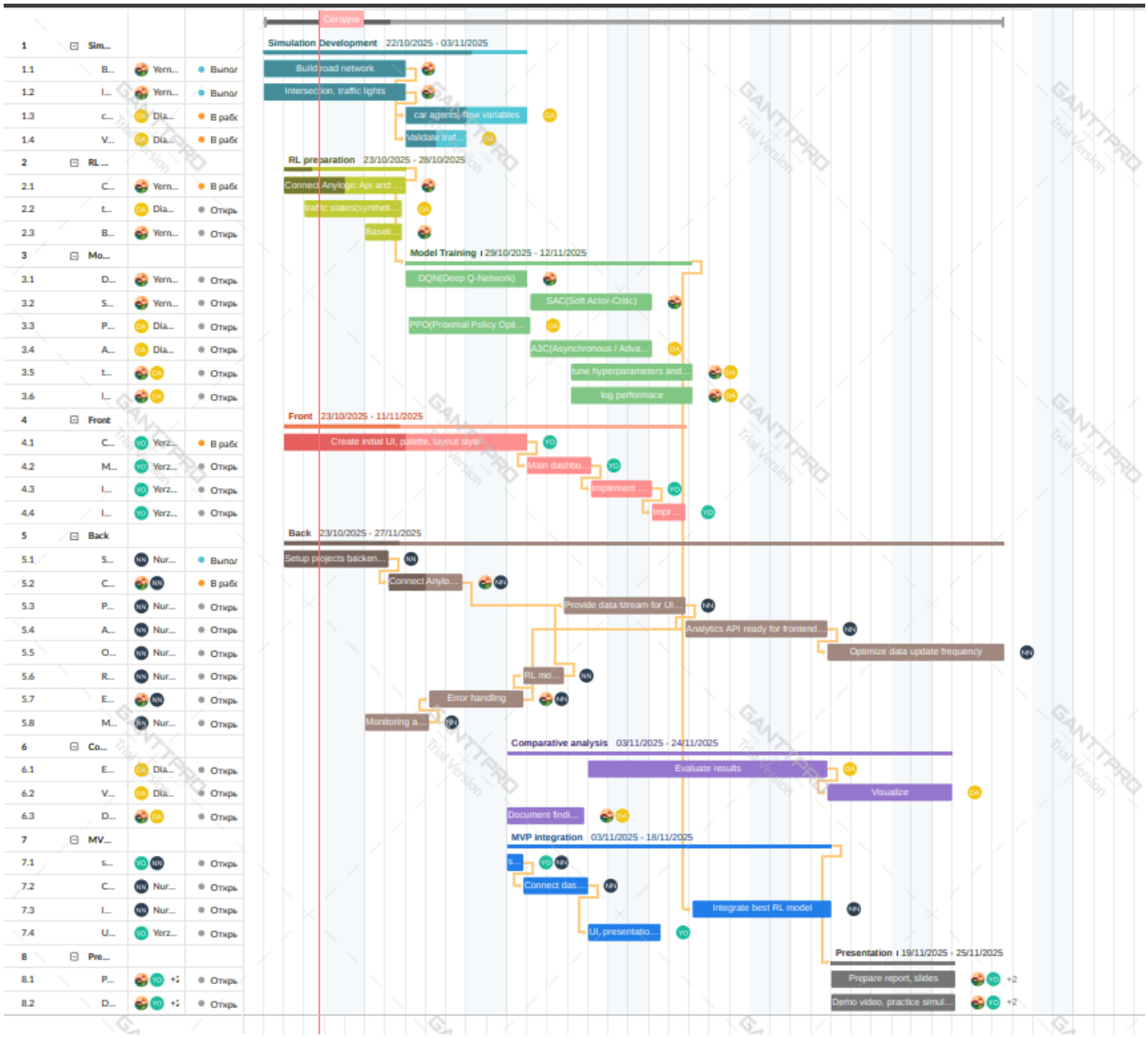


Figure 4.7 - Schedule planning in GanttPRO of an application for Urban road analytics

Learning objectives

By the end of this section, you can:

– develop a project schedule using a Gantt chart;
– include milestones and key dependencies in the schedule;
– establish and record a schedule baseline for control.

Practical tasks:

1. Create a Gantt chart for your project (use Excel, MS Project, or another tools such as GanttPRO (https://ganttpro.com/), Jira (https://www.atlassian.com/software/jira), Asana (https://asana.com).
2. Record the schedule baseline date and version.

Checklist**:**

– Chart includes milestones and key dependencies.
– Baseline version and date are recorded.
– Schedule matches your activity list.

4.5 Milestones and Critical Path

The Critical Path Method (CPM) is a technique used to identify the tasks required to complete a project and to plan its schedule flexibly. In project management, the critical path is the longest sequence of tasks that must be carried out to finish the project on time. Any delay in these tasks will directly affect the entire project timeline.

CPM involves identifying the most important tasks in the project timeline, determining their dependencies, and calculating the time required for their completion.

The method was developed in the late 1950s to address the issue of rising costs caused by poor planning. Since then, CPM has gained popularity to schedule projects and set priorities. It helps break down complex projects into smaller tasks and ensures greater flexibility in managing the project.

Why use the Critical Path Method?

CPM provides valuable insights for planning projects, allocating resources, and setting task deadlines.

Here are a few reasons to apply this method:

- It supports planning for future projects. CPM allows comparison between expected and actual progress. Data from current projects can also be used to improve the planning of upcoming ones.
- It improves resource management. CPM helps project managers identify priority tasks, giving them a clearer view of where and how resources should be used.
- It helps overcome obstacles. Delays in projects often result in wasted time. By mapping dependent tasks in a network diagram, you gain a clearer understanding of which tasks can run in parallel and which cannot. This makes it easier to create an efficient project schedule.

How to find the critical path?

We can find the critical path by analyzing the duration of tasks with different levels of importance. Below are the step-by-step actions with examples.

Step 1. Create a task list. Use a work breakdown structure to list all the activities or tasks in the project that must be completed to achieve the expected results. This task list in the work breakdown structure becomes the foundation for the rest of the CPM process (see Figure 4.8).

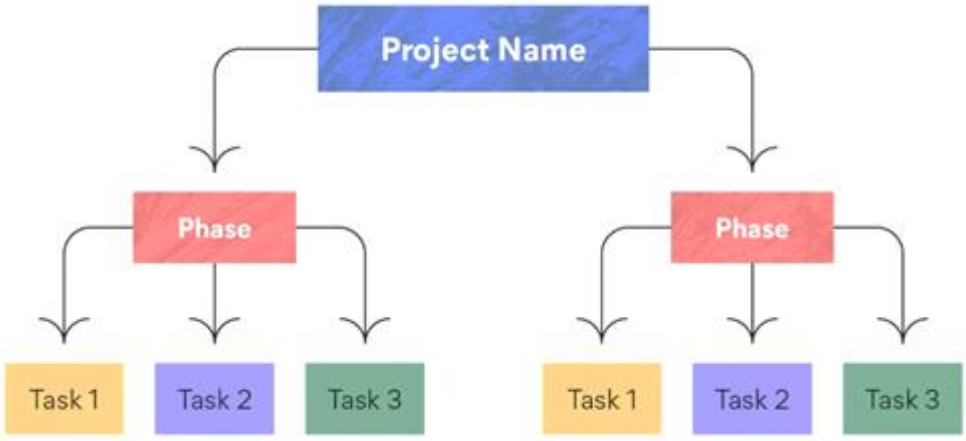


Figure 4.8 - Work Breakdown Structure

Suppose the IT team works on setting up a new internal system. Here's how the task breakdown may look (see Table 4.1).

Table 4.5 - Work Breakdown Structure for IT System Deployment Project

Task ID	Task	Duration (days)
A	Gather system requirements	2
B	Configure server environment	4
C	Develop core application code	6
D	Set up database	3
E	Test system functionality	3
F	Deploy system to production	1

Step 2. Identification of Dependent Elements.

Define the dependent tasks based on the work breakdown structure. This step shows which tasks a team can do at the same time as others.

Below is the task dependencies based on the example above:

- Task B depends on Task A
- Task C depends on Task B
- Tasks C and D run in parallel
- Task E depends on Task D
- Task F depends on Tasks C, D, and E

The list of dependent tasks forms the sequence of activities, which serves as the basis for the critical path.

Step 3. Creation of a Network Diagram.

The next step is to turn the work breakdown structure into a network diagram that shows the order of tasks. Draw a box for each task and use arrows to show the dependencies between them.

Add all components with their time frames into the network diagram until the full project schedule appears.

Step 4. Task Duration Estimation.

To calculate the critical path, which is the longest chain of key tasks, you first need to estimate the duration of each task.

You may estimate by:

- Assumptions based on experience and knowledge
- Data from previous projects
- Industry standards

Two methods help with this process:

- Forward Pass Method. This method calculates Early Start (ES) and Early Finish (EF) dates by using a given start date. Early Start equals the largest Early Finish of the previous task. Early Finish equals Early Start plus the task duration. The calculation begins at zero from the first task and moves forward through the schedule. This method shows when resources should be allocated.

- Backward Pass Method. This method calculates Late Start (LS) and Late Finish (LF) dates. Late Start equals Late Finish minus the duration. Late Finish equals the smallest Late Start of the following task. The calculation begins from the last scheduled task and moves backward.

By comparing early and late dates, you can determine float time and provide flexibility in the project schedule.

Step 5. Critical Path Calculation.

You can calculate the critical path by hand or by an algorithm (see Figure 4.9). Manual calculation goes as follows:

- Write start and finish time for each task. The first task starts at 0.
- For each next task, set its start time equal to the finish time of the previous task. Finish time equals start time plus duration.
- The total duration of a sequence equals the finish time of its last task.
- The sequence with the longest duration is the critical path.

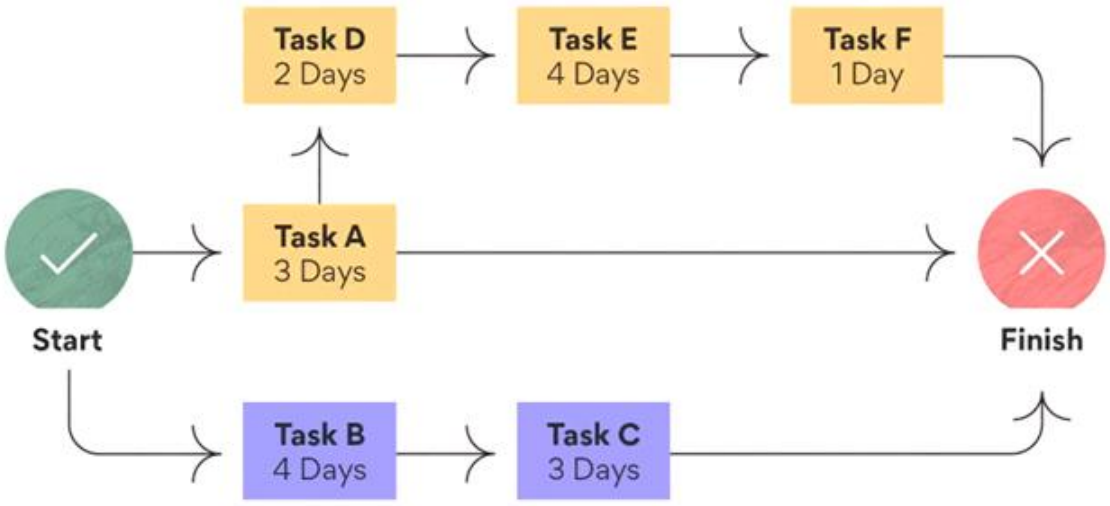


Figure 4.9 - Example of the critical path schedule

Once you identify the critical path, you can build the actual project schedule on its basis.

Time Reserve in Projects.

Float shows how much flexibility each task has. Critical tasks have zero float, so delay is impossible. Non-critical tasks have positive float, so delay is possible without harming the project deadline.

Two types exist:

- Total float: time a task may shift without delaying the project. Formula: LS – ES or LF – EF.
- Free float: time a task may shift without delaying the next task. Formula: ES (next task) – EF (current task).
- Float helps managers control deadlines, set priorities, and handle risks.

Table 4.6 - Milestone list example for Service Desk Web Portal v1.0

Milestone	Planned date/week	Owner	Acceptance condition
Project charter approved	Week 1	Sponsor	Signed charter and assigned PM
Requirements approved	Week 2	Sponsor and PM	Requirements list approved and baseline set
UI prototype approved	Week 3	UX and Service Desk manager	Prototype approved with no critical gaps
Architecture and database design approved	Week 3	Lead developer	Design document approved by PM and IT admin
MVP demo completed	Week 6	PM and Dev team	Core flow works (login, ticket creation and dashboard)

System testing completed	Week 8	QA	Test report completed, critical defects closed
UAT started	Week 9	PM	UAT plan approved and users invited
UAT accepted	Week 10	Sponsor	UAT checklist signed, no critical issues open
Go-live (production release)	Week 10	IT admin	Portal deployed and smoke test passed
Handover and project closure	Week 10	PM	Handover package delivered, lessons learned recorded

Learning objectives

By the end of this section, you can:

- define meaningful project milestones;
- identify critical path activities;
- understand which activities directly affect project duration.

Practical tasks:

1. Define 6-10 milestones (example: requirements approved, MVP demo, UAT start, go-live).

2. Identify the critical path activities based on your network and durations.

Table 4.7 - Milestone list

Milestone	Planned date/week	Owner	Acceptance condition

Checklist:

- Milestones include approvals, not only build steps.
- Critical path list matches network logic.
- Project finish date is realistic for 10 weeks.

CHAPTER 5. COST PLANNING

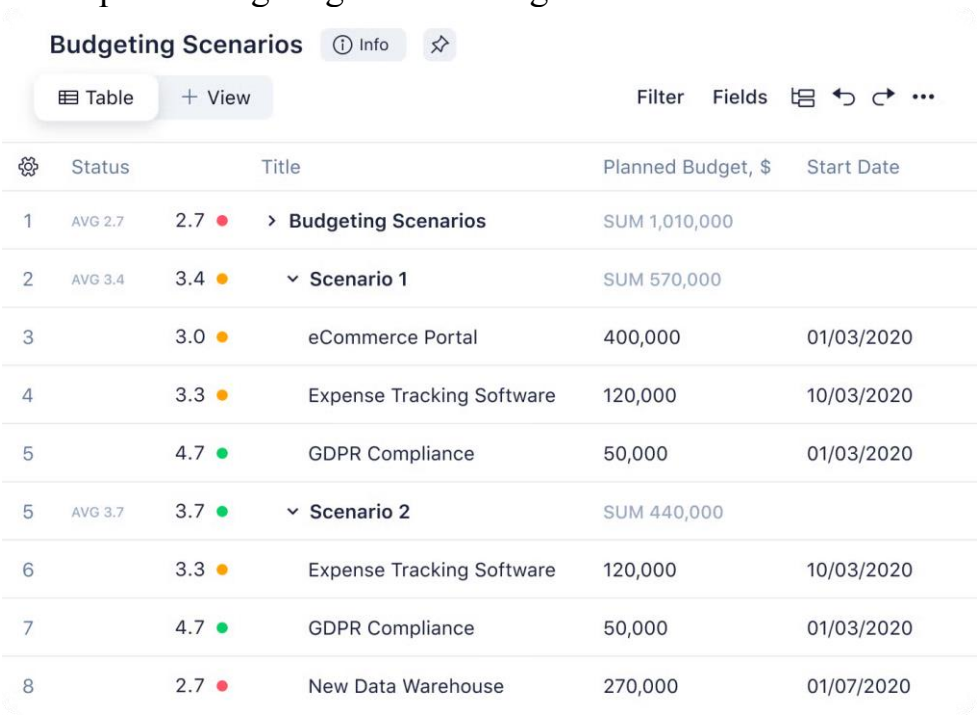
Cost management involves the estimation, allocation, and control of project costs. This process allows an organization to predict future expenses and reduce the risk of exceeding the budget. During the planning stage, the project team calculates projected costs and obtains approval before the project starts. As the project proceeds, the team documents and tracks all expenses to ensure compliance with the cost management plan. After project completion, the team compares estimated costs with actual costs, providing benchmarks for future budgets and cost management plans. Project cost management is important for an organization's project planning process. Without a clear budget, we cannot plan the resources our project requires. For example, renovating an office building requires hiring an architect, buying building materials, and paying construction workers. We must calculate all costs accurately and ensure the budget can cover them.

Benefits of Cost Management in Project Management.

Project managers should recognize the business value of effective cost management. Key benefits include:

- Prevents overruns: Allocating costs during the planning stage helps ensure the project does not exceed the budget in any area.
- Reduces risk: A well-prepared budget includes a risk allowance, protecting the project from unexpected expenses.
- Supports future planning: Cost reports provide insights into resource use, leading to more accurate budgets for future projects.

Example of budgeting shown in Figure 5.1.



	Status		Title	Planned Budget, $	Start Date
1	AVG 2.7	2.7	Budgeting Scenarios	SUM 1,010,000	
2	AVG 3.4	3.4	Scenario 1	SUM 570,000	
3		3.0	eCommerce Portal	400,000	01/03/2020
4		3.3	Expense Tracking Software	120,000	10/03/2020
5		4.7	GDPR Compliance	50,000	01/03/2020
5	AVG 3.7	3.7	Scenario 2	SUM 440,000	
6		3.3	Expense Tracking Software	120,000	10/03/2020
7		4.7	GDPR Compliance	50,000	01/03/2020
8		2.7	New Data Warehouse	270,000	01/07/2020

Figure 5.1 - Example of Budgeting

Cost management can be difficult. Common challenges include:

- Limited resources: A small budget makes it hard to secure the labor, materials, and other resources needed to complete the project.
- Inaccurate estimation: Poor forecasting from inexperience or lack of understanding of the project scope can cause cost overruns and reduce profitability.
- Outdated technology: Project managers require modern and user-friendly tools to track and control costs accurately.

5.1 Cost Estimates (Bottom-Up and Analogous)

Project cost estimation helps managers predict how much an IT project will cost before it starts. Accurate estimates prevent overspending and ensure proper resource allocation.

1. Analogous Estimating

Definition: Use the cost of a similar past IT project as a reference.

Advantage: Fast and simple.

Limitation: Less accurate if the new project is different.

Table 5.1 shows an example of analogous estimating for IT projects.

Table 5.1 – Example of Analogous Estimating in IT Projects

Past Project	Cost	New Project	Estimated Cost
Company Website Development	$30,000	E-Commerce Website	$35,000

2. Parametric Estimating

Definition: Use formulas or unit rates to calculate costs.

Advantage: More accurate with reliable data.

Limitation: Needs precise input data.

Table 5.2 shows an example of parametric estimating for an IT project.

Table 5.2 – Example of Parametric Estimating in IT Projects

Task	Unit	Cost per Unit	Total Cost
Web Page Development	20 pages	$1,000	$20,000

3. Bottom-Up Estimating

Definition: Estimate costs for each task separately, then sum them.

Advantage: Very accurate.

Limitation: Time-consuming and requires detailed task knowledge.

Table 5.3 shows an example of bottom-up estimating for an IT project.

Table 5.3 – Example of Bottom-Up Estimating in IT Projects

Task	Estimated Cost

Requirement Analysis	$3,000
Frontend Development	$15,000
Backend Development	$20,000
Testing	$5,000
Total	$43,000

4. Three-Point Estimating

Definition: Use three estimates for each task: optimistic, most likely, pessimistic. Calculate the expected cost.

Advantage: Accounts for uncertainty.

Limitation: Requires judgment to select estimates.

Formula: Expected Cost = (Optimistic + 4 × Most Likely + Pessimistic) ÷ 6

Table 5.4 shows an example of three-point estimating for an IT project.

Table 5.4 – Example of Three-Point Estimating in IT Projects

Task	Optimistic	Most Likely	Pessimistic	Expected Cost
API Integration	$4,000	$5,000	$6,000	$5,000

Table 5.5 - Cost estimates example for Service Desk Web Portal v1.0 (bottom-up)

Cost item	Category	Qty	Unit cost	Total	Basis/assumption
Cloud hosting (3 months)	Hosting	3	40	120	Small web app VM plan
Domain registration (1 year)	Domain	1	15	15	Standard .com/.kz domain
SSL certificate (basic)	Security	1	0	0	Use free certificate (e.g., Let's Encrypt)
Email service (transactional)	Service	1	20	20	Low-volume notifications
Backup storage (3 months)	Hosting	3	10	30	Weekly backups
UI kit/license (optional)	Tools	1	50	50	Only if paid UI kit used
Testing tool subscription (1 month)	Tools	1	25	25	Bug tracking/test tool plan
Documentation/printing	Materials	1	10	10	Short guides for training
User training session	Training	1	60	60	2-hour session and materials
Contingency for minor purchases	Reserve	1	100	100	Small unexpected costs

Learning objectives

By the end of this section, you can:

– estimate project costs using a bottom-up approach;
– identify common cost items in a project;
– explain an analogous cost estimate based on a similar project.

Practical tasks:

1. Create bottom-up estimates for at least 10 cost items (tools, hosting, certificates, printing, training).

2. Write one analogous estimate statement based on a similar software project.

Table 5.6 - Cost Estimation

Cost item	Category	Qty	Unit cost	Total	Basis/assumption

Checklist:

– Each estimate has a basis or assumption.
– Totals match Qty x Unit cost.
– Major costs align with scope and schedule.

5.2 Cash flow plan

Budget development refers to the process of combining all estimated project costs into a single, authorized financial plan. This plan includes direct costs (such as labor, materials, and equipment) and indirect costs (such as administration and overhead). The project budget provides the financial framework for monitoring and controlling costs throughout the project life cycle.

The key steps in budget development include:

1. Cost Estimation: Identify and calculate the expected costs for each activity or work package.

2. Cost Aggregation: Combine all cost estimates into higher-level categories, usually aligned with the Work Breakdown Structure (WBS).

3. Contingency Reserves: Add reserves to cover risks and uncertainties.

4. Management Reserves: Include an allowance for unknown risks, often at the organizational level.

5. Approval: Obtain authorization from project sponsors or stakeholders to use the budget as the official cost reference.

Table 5.7 shows a simple example of project budget development in IT projects.

Table 5.7 – Example of Project Budget Development

Work Package	Estimated Cost	Contingency Reserve	Total Cost

Software Design	$10,000	$1,000	$11,000
Development	$25,000	$2,500	$27,500
Testing	$8,000	$800	$8,800
Deployment	$5,000	$500	$5,500
Total Budget	$48,000	$4,800	$52,800

The cost baseline represents the approved version of the budget, excluding management reserves. It serves as the benchmark for measuring project performance. Project managers compare actual project costs against the cost baseline to identify variances, control spending, and ensure financial discipline. The cost baseline is typically presented as a time-phased budget, which shows how costs are expected to occur over the project timeline. This allows project managers to track expenditures in relation to planned progress. Figure 5.2 shows an example of a cost baseline curve (also known as an S-curve).

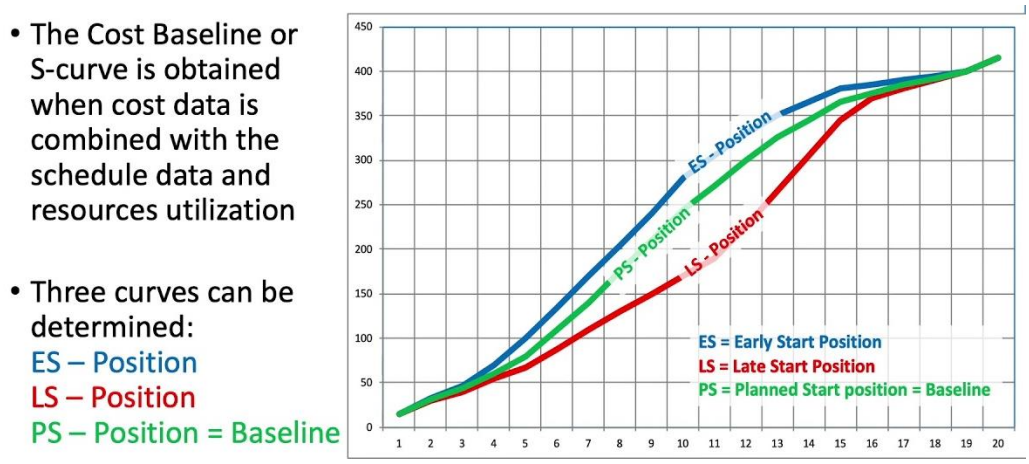


Figure 5.2 - Cost Baseline Curve

The horizontal axis represents time (project phases or months). The vertical axis represents cumulative cost. The curve shows planned expenditure over time, which can be compared with actual costs.

Earned Value Management (EVM) is one of the most effective techniques for monitoring and controlling project costs. It integrates scope, schedule, and cost to provide a clear picture of project performance. By comparing planned work with actual work and costs, project managers can detect deviations early and take corrective action.

To apply earned value methods, several key terms must be understood (see Table 5.8):

Budget for Completion (BAC): the total cost of the project, which includes all resources, expenses, services, and subcontract amounts.

Current Budget: the initial baseline budget adjusted by approved changes.

Planned Value (PV): the portion of the budget scheduled for completion by a specific date.

Actual Cost (AC): the real expenses recorded on a specific date. This amount includes labor hours, equipment uses, delivered materials, service fees, and other charges.

Earned Value (EV): the value of work completed. The calculation multiplies the project's percent complete (PC) by the total budget.

Table 5.8 – Key Terms in Earned Value Management

Term	Definition	Example (IT Project)
Planned Value (PV)	Budgeted cost of work scheduled	$20,000 planned for Month 2
Earned Value(EV)	Budgeted cost of work performed	$18,000 of tasks completed
Actual Cost (AC)	Actual cost of work performed	$22,000 spent on Month 2

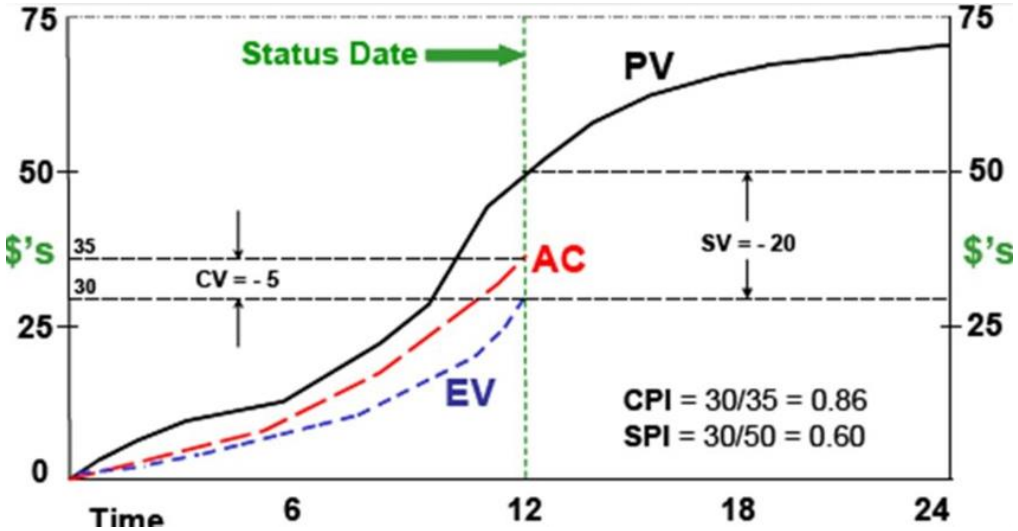


Figure 5.3 - Adapted from J. A. Lukas (2012), How to Make Earned Value Work on Your Project [14]

How EVM Works. The project budget is spread across the timeline according to the project schedule. This time-phased structure creates a basis for comparison:

- Planned Value shows the budgeted cost expected by the current date.
- Actual Cost shows the money already spent.
- Earned Value shows the budgeted worth of completed work.

A comparison of these three measures provides a reliable view of project performance and supports effective cost control.

Core formulas in Earned Value Management.

1. Cost Variance (CV):

$$CV = EV - AC \tag{5.1}$$

Positive CV = under budget.

Negative CV = over budget.

2. Schedule Variance (SV):

$$SV = EV – PV \tag{5.2}$$

Positive SV = ahead of schedule.
Negative SV = behind schedule.

3. Cost Performance Index (CPI):

$$CPI = EV / AC \tag{5.3}$$

CPI > 1: project under budget
CPI = 1: project on budget
CPI < 1: project over budget

4. Schedule Performance Index (SPI):

$$SPI = EV / PV \tag{5.4}$$

SPI > 1: Ahead of schedule
SPI = 1: On schedule
SPI < 1: Behind schedule

5. Estimate at Completion (EAC):

$$EAC = BAC / CPI \tag{5.5}$$

Forecast of total project cost based on current performance.

6. Estimate to Complete (ETC):

$$ETC = EAC − AC \tag{5.6}$$

Expected cost required to finish remaining work.

7. Variance at Completion (VAC):

$$VAC = BAC – EAC \tag{5.7}$$

Positive VAC = project expected under budget.
Negative VAC = project expected over budget.

Table 5.9 - Cash flow example for Service Desk Web Portal v1.0

Week	Planned spend	Main items	Notes
Week 1	15	Domain registration	Purchase after charter approval
Week 2	120	Hosting (3 months prepaid)	Environment setup starts
Week 3	25	Testing tool (1 month)	Start defect tracking
Week 5	20	Email service	Needed before notification testing
Week 6	30	Backup storage	Enable before system testing
Week 9	60	User training session	Before UAT / go-live
Week 10	10	Printing/documentation	Handover package
Week 10	100	Contingency reserve	Use only if approved

Learning objectives
By the end of this section, you can:
- plan project cash flow over time;
- link costs to the project schedule;
- understand how spending changes during project execution.

Practical tasks**:**
1. Create a simple week-by-week cash flow for key cost items.

Table 5.10 - Cash Flow

Week	Planned spend	Main items	Notes

Checklist:
- Spend timing matches schedule and procurement steps.
- Large costs appear before the activity that uses them.

5.3 Budget and Cost Baseline

After cost estimating, the project team defines a clear budget. A budget is the total approved amount allocated to the project. It consolidates planned cost items and

sets financial limits for delivery. A budget supports realistic planning, because scope and schedule decisions must fit the available funding.

A cost baseline is the approved version of the budget used for cost control. It becomes the reference point for tracking actual spending during execution. After approval, the project manager compares planned costs with actual costs and reports deviations. Any request that increases costs beyond the baseline requires a formal change decision.

Budgets often include reserves. A contingency reserve covers expected uncertainty and known risks, such as extra testing effort or minor tool purchases. A management reserve is optional and remains under sponsor control for unexpected events outside the planned scope. Reserve rules should define who can approve reserve usage and under what conditions.

For the Service Desk Web Portal v1.0, the cost baseline should include items such as hosting, domain, tools, training, and documentation, plus a defined contingency reserve (Table 5.11). Cost performance should be reviewed regularly as part of project status reporting.

Table 5.11 - Budget summary example for Service Desk Web Portal v1.0

Field	Sample value
Estimated total	430
Contingency reserve	50
Management reserve (optional)	0
Approved budget (baseline)	480
Notes	Baseline includes hosting, domain, tools, training, and documentation. Contingency covers minor unexpected costs and extra testing needs.

Learning objectives

By the end of this section, you can:

- calculate a total project budget from cost estimates;
- define a contingency reserve;
- justify the need for reserves to manage uncertainty.

Practical tasks:

1. Sum cost estimates and define a total budget for chosen project.
2. Define a contingency reserve (example: 10%) and explain why.

Table 5.12 - Budget Summary

Estimated total	
Contingency reserve	

Management reserve (optional)	
Approved budget (baseline)	
Notes	

Checklist:

- Baseline amount is clear and approved by the sponsor.
- Reserve levels have a simple justification.
- Budget aligns with constraints.

CHAPTER 6. QUALITY AND RISK PLANNING

6.1 Quality Standards and Metrics

Quality planning defines what "good" means for the project deliverables. A *quality standard* describes the expected level of performance or compliance. It sets clear targets for the product and for the project work. In IT projects, quality often includes usability, reliability, security basics, and maintainability, not only the absence of defects. Clear standards prevent disputes during testing and acceptance because the team and stakeholders share the same expectations.

A *quality metric* is a measurable indicator used to evaluate whether quality standards are met. Metrics support objective control because they replace vague statements such as "the system is easy to use" with specific targets such as "a user can create a ticket in under 2 minutes." Metrics may focus on product quality (response time, defect rate, availability) and process quality (test coverage, review completion, change stability). Each metric should include a target value, a measurement method, a data source, and an owner responsible for monitoring and reporting.

Quality standards and metrics should align with project objectives and stakeholder needs. High-impact areas usually include core user flows, data protection, system stability, and support readiness. Metrics should remain simple and practical for the project size. The goal is consistent delivery and controlled acceptance, not complex measurement.

"The purpose of the Software Quality Assurance Plan (SQAP) is to define the techniques, procedures, and methodologies that will be used in the project to ensure timely delivery of the software that meets specified requirements within project resources." [19]

What is Quality?

Before providing the definition of Quality please try to answer the following questions:

- How do you understand the general term of software product Quality?
- Do you think it should be connected to conformance to requirements?
- Do you think it should be connected to fitness or convenience of use?

So Quality is a complex notion of both conformance to requirements and fitness of friendly interface and use. There are two processes tightly connected to Quality:

- Verification – "The evaluation of whether a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with validation." [19]

- Validation – "The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with verification" [19]

Fault, failure and error. In addition to that there are three definitions of nonconformance of Quality:

- Fault/defect – a condition that may cause a failure in a system, also called a bug.

- Failure/problem – the inability of a system to perform a function according to its specification, result of a defect.
- Error – a mistake made by software engineer or programmer

For the Service Desk Web Portal v1.0, quality planning should prioritize reliable ticket submission, clear status tracking, correct role access, and stable performance during normal use. Defined metrics support testing, weekly reporting, and final acceptance decisions (see Table 6.1).

Table 6.1 - Quality metrics example for Service Desk Web Portal v1.0

Quality area	**Standard or target**	**Metric**	**Measurement method**	**Owner**
Functionality	Core ticket flow works end-to-end	% of critical user-flow test cases passed	Acceptance test results (pass/fail summary)	QA
Usability	Ticket creation is simple and fast	Average ticket creation time ≤ 2 minutes	Usability test with 5–10 users and time measurement	UX
Performance	Pages load fast under normal use	Dashboard load time ≤ 3 seconds	Browser timing / simple performance test	Developer
Security	Role access works correctly	Number of access-control violations = 0	Role-based access tests (attempt forbidden pages)	IT admin / security officer
Reliability	Stable operation during business hours	Uptime ≥ 99% during pilot period	Hosting logs or monitoring report	IT admin

Learning objectives

By the end of this section, you can:

– define quality standards for a project;
– select measurable quality metrics;
– explain how quality performance will be evaluated.

Practical tasks:

1. Define five quality standards for the chosen project.
2. Define five metrics and explain how you will measure each metric.

Table 6.1 - Quality Metrics

Quality area	Standard or target	Metric	Measurement method	Owner

Checklist:

- Each metric has a target and a measurement method.
- Owners are assigned for quality control tasks.

6.2 Quality checklist and review plan

Quality control requires simple, repeatable checks. A *quality checklist* is a practical tool that helps the team confirm that deliverables meet agreed requirements and standards. It reduces missed steps and supports consistent work, especially during testing and release. A checklist can cover functional items (features work as expected) and non-functional items (security basics, performance targets, documentation, and readiness for handover). Each checklist item should be clear, measurable, and easy to verify.

A *review plan* defines when quality reviews take place and who participates. Reviews are planned checkpoints that catch problems early, before they become expensive to fix. In software projects, typical review points include a design or UI review, a technical review, a test readiness review, and a release review. The plan should state the review purpose, timing (week or milestone), required inputs, and expected outputs such as approval, feedback, or a decision.

A strong review plan supports predictable delivery. It sets a clear rhythm for stakeholder involvement and prevents last-minute surprises during acceptance. It also clarifies responsibilities: who prepares materials, who approves, and who records actions.

Software development requires testing; a set of techniques used to detect and fix errors in a software product. This process can be automated and should be applied to all aspects of the project.

A *test plan* is a document that outlines the scope, approach, resources, and schedule for all planned testing activities. It serves as a record of the testing strategy.

The document identifies several key components:

- Test items are what specifically will be tested.
- Features to be tested are specific functions and features to be included.
- Testing tasks are the actions that testers will perform.
- The personnel are the person who is responsible for each testing task and their level of independence.
- Test environment is the necessary hardware, software, and tools.
- Test design is techniques that will be used.
- Entry and exit criteria are conditions required to start and finish a testing phase.
- Risks are potential issues and plans for how to address them.

Types of test plans:

- Master test plan is a comprehensive, high-level plan that consolidates all other test plans for a single project or product.
- Level-specific test plans are separate plans for each stage of testing:
    - Unit test plan is for testing individual components.

- Integration test plan is for testing how components work together.
- System test plan is for testing the entire product as a whole system.
- Acceptance test plan is for verifying that the system meets user requirements.

- Type-specific test plans are for plans created for major types of testing, such as a Performance Test Plan or a Security Test Plan.

Who does the testing?

Testing is a collaborative process involving several roles to ensure software quality:

- Programmers (developers) - they perform unit testing and component testing on the code they write to find and fix errors early.
- Testers - dedicated quality assurance professionals who perform various types of testing, including functional, integration, and system testing.
- Users - end-users participate in real-world testing:
  - Alpha testing is a type of acceptance testing done by internal users or teams to find bugs before a product is released to the public.
  - Beta testing is a testing done by a select group of external, real-world users in a live environment.

Testing targets different parts of the software at different stages (see Figure 6.1). The main test levels are the following.

- **Unit testing** focuses on individual, isolated modules or components of the source code.
- **Functional testing** verifies that the software's features and functions behave as specified in the requirements.
- **Component testing** checks a complete, individual component to make sure it works correctly on its own.
- **Integration and system testing** include several types:
  - **Integration testing** verifies that different components work together correctly.
  - **System testing** tests the entire integrated system to confirm it meets all functional and technical requirements.
  - **Smoke test** is a quick, basic check to ensure the core functionality works before more detailed testing begins.

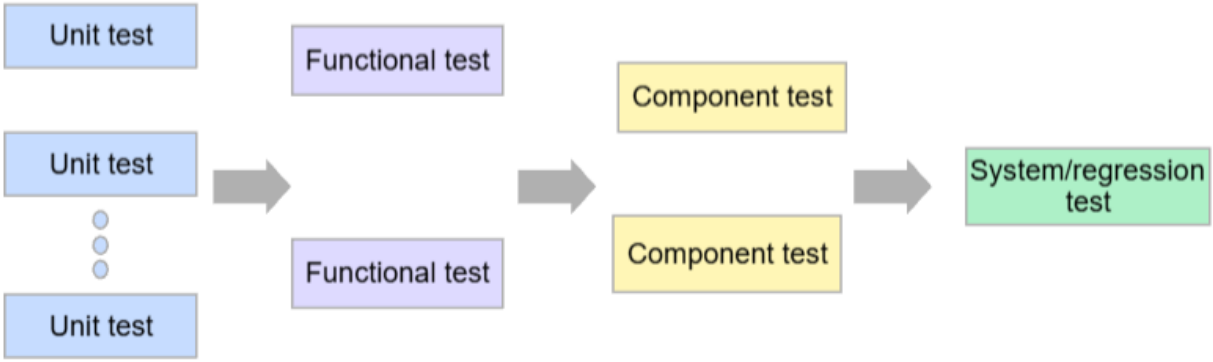


Figure 6.1 - Testing hierarchy

Test case metrics

These metrics describe the execution status and outcomes of test cases.

– Passed vs. failed test cases: the core indicator showing how many tests succeeded and how many failed, giving a quick snapshot of product quality.

– Not run test cases: the number of test cases that exist but have not been executed yet; a rising value may signal schedule slippage, blocked work, or incomplete test coverage.

Bug metrics

These metrics summarize the volume, characteristics, and progress of defect handling.

– Open vs. closed bugs: a basic measure of defect resolution progress and how quickly issues are being addressed.

– Reopened vs. closed bugs: tracks defects that were marked as fixed but appeared again, often indicating an incomplete fix or gaps in verification.

– Rejected vs. opened bugs: helps evaluate the quality of defect reporting (e.g., invalid reports, duplicates, or insufficient evidence).

– Bugs by severity: groups defects by impact on the system (for example, critical, major, or minor).

– Bugs by priority: ranks defects by urgency of fixing them (for example, high, medium, or low).

For the Service Desk Web Portal v1.0, a checklist can confirm core functions such as ticket submission, status updates, and role access, along with performance and notification rules (Table 6.2). A review plan can schedule key checkpoints such as requirements approval, UI prototype review, MVP demo review, test readiness review, and final release approval.

Table 6.2 - Release checklist example for Service Desk Web Portal v1.0

Check item	Pass/Fail	Notes
Login and role access work		
Ticket submission and ID creation work		
User dashboard shows correct status		
Agent assignment and status updates work		
Email notifications work		
Admin settings and basic reports work		
No critical defects open		
Production smoke test passed		

Learning objectives

By the end of this section, you can:

– define quality control checks for a project release;

– plan formal reviews and assign responsibility;
– support consistent quality before delivery.

Practical tasks:
1. Create a release checklist for the chosen project (at least 12 checks).
2. Create a review plan with four review points and owners.

Table 6.3 - Release checklist

Check item	Pass/Fail	Notes

Table 6.4 - Review Plan:

Review point	When (week)	Owner	Output

Checklist:
– Checklist items map to requirements and risks.
– Review points fit the schedule and milestones.

CHAPTER 7. PROJECT RESOURCES AND TEAM

7.1 Team roles and RACI matrix

Successful projects depend on clear responsibility. Team members often work in parallel, and many tasks require input from several roles. Without role clarity, work may be delayed, duplicated, or left unfinished. This section defines team roles and uses a RACI matrix to show responsibility for key activities.

Team roles describe who does what on the project. Each role includes a main purpose, typical responsibilities, and expected outputs. In IT projects, common roles include the sponsor, project manager, developers, QA, UX, and system administrator. Roles may be full-time or part-time, but each responsibility must have a clear owner.

A **RACI matrix** is a simple responsibility chart that connects activities to roles. RACI uses four responsibility types:

- *R (Responsible):* performs the work and produces the output.
- *A (Accountable):* owns the result and approves it. Only one role should be Accountable for each activity.
- *C (Consulted):* provides input or expertise before completion.
- *I (Informed):* receives updates after decisions or completion.

The RACI matrix supports faster decisions and reduces confusion during execution. It also clarifies approval points, which are critical for requirements, scope changes, testing, and release decisions.

For the Service Desk Web Portal v1.0, the RACI matrix should cover activities such as requirements approval, UI review, development, testing, deployment, training, and final acceptance (Table 7.1). A clear RACI helps the team deliver results within time and budget limits while keeping stakeholders aligned.

Table 7.1 - RACI matrix example for Service Desk Web Portal v1.0

Activity	Sponsor	PM	Dev	QA	UX	IT admin
Approve project charter	**A**	R	I	I	I	I
Approve requirements	**A**	R	C	C	C	C
Approve UI prototype	**A**	C	C	I	**R**	I
Develop core features (ticket flow)	I	**A**	**R**	C	C	C
System testing and defect reporting	I	**A**	C	**R**	C	I
UAT and final acceptance	**A**	R	C	C	C	I
Production deployment	I	**A**	C	I	I	**R**

Learning objectives

By the end of this section, you can:

– define quality control checks for a project release;
– plan formal reviews and assign responsibility;

– support consistent quality before delivery.

Practical tasks:

1. Define roles for PM, Dev, QA, UX, and Sponsor for this case.

2. Create a RACI for 10 activities, including requirements approval, testing, deployment, and training.

Table 7.2 - RACI matrix

Activity	Sponsor	PM	Dev	QA	UX

Checklist:

– Each activity has one Accountable role.

– RACI matches the stakeholder list and team structure.

7.2 Resource plan and workload table

A project schedule shows when work should happen, but it does not automatically show who has the capacity to do it. A *resource plan* connects project work with people and roles. It shows who works on which deliverables, the level of effort required, and the time period when that effort is needed. A clear resource plan supports realistic deadlines and prevents gaps in responsibility.

A *workload table* is a simple view of resource demand over time (often by week). It helps identify overload, where too much work is planned for one person or one role in the same period. Overload increases risk: delays, lower quality, and missed reviews. The workload table also shows underuse, where a role is available but not assigned clear work.

Resource planning should reflect real constraints. Team members may be part-time, may support several projects, or may have fixed availability. Resource planning should also include key support roles such as IT administration and security review, because these roles often affect deployment and acceptance timing.

For the Service Desk Web Portal v1.0, the resource plan should define the main roles (PM, Dev, QA, UX, IT admin) and show how effort shifts across the 10-week timeline. Early weeks usually require more PM and UX work, middle weeks require more development, and later weeks require more QA, deployment support, and closure documentation (Table 7.3).

Table 7.3 - Workload table example for Service Desk Web Portal v1.0

Week	PM (hrs)	Dev (hrs)	QA (hrs)	UX (hrs)	Notes
Week 1	12	0	0	4	Charter, kickoff, stakeholder list

Week 2	10	8	0	10	Requirements and early design input
Week 3	8	16	0	12	UI prototype and design approval
Week 4	6	24	4	4	Architecture and start core development
Week 5	6	32	4	2	Core features build
Week 6	6	32	6	2	MVP demo and improvements
Week 7	6	28	10	0	Feature completion and test preparation
Week 8	6	20	16	0	System testing and defect fixes
Week 9	8	12	12	0	UAT support and release readiness
Week 10	10	8	6	0	Go-live, handover, closure documents

Learning objectives

By the end of this section, you can:

– estimate workload in hours for project roles;
– distribute effort over the project timeline;
– identify workload risks and define mitigation actions.

Practical tasks:

1. Estimate weekly effort (hours) for each role for 10 weeks.
2. Identify one workload risk and one mitigation action.

Table 7.4 - Workload table

Week	PM (hrs)	Dev (hrs)	QA (hrs)	UX (hrs)	Notes
1					
…					

Checklist:

– Workload fits realistic capacity per role.
– Peak weeks have a plan (support, scope cut, or re-sequencing).

7.3 Team rules and conflict resolution plan

Project work depends on coordination, deadlines, and decisions. Even in a small team, misunderstandings and conflicts can appear because of unclear expectations, different priorities, or limited time. Team rules provide simple agreements that guide daily work. A conflict resolution plan defines how the team handles disagreements in a professional and timely way.

Team rules set common working standards. They define how the team communicates, how tasks are assigned, how progress is reported, and how documents are updated. Typical rules include meeting frequency, response time expectations, file naming and versioning, and basic quality expectations such as code reviews or checklist use. Clear rules reduce confusion and help the team work consistently.

A *conflict resolution plan* describes steps for resolving issues before they harm schedule or quality. The plan should encourage direct and respectful communication, fast escalation when needed, and clear decision authority. Most conflicts can be resolved at the team level through discussion and facts (scope baseline, requirements, schedule constraints). If the team cannot resolve the issue, the project manager escalates it to the sponsor or another decision maker.

For the Service Desk Web Portal v1.0, conflicts may occur when stakeholders request extra features, when roles disagree on priorities, or when technical constraints affect deadlines. Team rules and a clear escalation path help the team protect the scope baseline and deliver within the 10-week timeline.

Team agreement (outline) for Service Desk Web Portal v1.0:

1. Communication channels (daily, weekly, urgent)
- Daily: team chat (project channel) for coordination and quick questions.
- Weekly: email summary from the PM to sponsor and key stakeholders.
- Urgent: phone call or direct message to PM; PM escalates to sponsor if needed.
2. Meeting rhythm (standup, planning, review)
- Standup: 10–15 minutes, 3 times per week (Mon/Wed/Fri).
- Planning: 45 minutes at the start of each week (task priorities and owners).
- Review/demo: 30 - 45 minutes at the end of each week (progress, feedback, decisions).
- Optional: short QA triage meeting during testing weeks.
3. Definition of done (basic). A work item is “done” when:
- Function works as described in the requirement.
- Basic tests pass (no critical defects).
- Changes are documented (release notes or update log).
- Relevant checklist items are completed.
4. Decision process (who decides, how to escalate)
- PM decides day-to-day task priorities and internal workflow.
- Technical decisions: lead developer decides after consultation with IT admin.
- Scope/schedule/budget changes: sponsor decides based on a change request.
- Escalation: team member → PM (within 1 day) → sponsor (within 2 days).
5. Conflict process (steps and time limits)
- Step 1: Direct discussion between involved team members (same day).
- Step 2: PM mediation with facts (requirements, scope baseline, schedule) (within 1 working day).
- Step 3: Sponsor decision for unresolved scope, time, or cost conflict (within 2 working days).
- Step 4: Decision recorded in the decision log and communicated to the team (same day).

Learning objectives
By the end of this section, you can:
– define basic rules for team collaboration;
– document a simple team agreement;
– describe a clear process for resolving conflicts.

Practical task:
1. Write a one-page team agreement and a short conflict resolution process.

Template: Team Agreement (outline)
1. Communication channels (daily, weekly, urgent).
2. Meeting rhythm (standup, planning, review).
3. Definition of Done (basic).
4. Decision process (who decides, how to escalate).
5. Conflict process (steps and time limits).

Checklist:
– Rules are specific and easy to follow.
– Conflict steps include escalation to the sponsor if needed.

CHAPTER 8. STAKEHOLDER COMMUNICATION AND ENGAGEMENT

8.1 Communication plan

A communication plan also supports project control. When communication is predictable, the team spends less time on ad-hoc explanations and more time on delivery. It also reduces the risk of "information gaps," where important stakeholders make assumptions because they did not receive timely updates. In practice, many project delays come from late approvals, unclear decisions, or missing feedback. A communication plan addresses these problems by defining when stakeholders must respond and what happens if they do not respond on time.

The plan should separate three communication types:

1. status communication (regular updates on progress, risks, and next steps);
2. coordination communication (daily or weekly team alignment on tasks and blockers);
3. decision communication (formal approvals, change decisions, and sign-offs).

Each type has a different audience, format, and level of detail. Mixing them often creates overload for stakeholders and reduces attention to critical items.

A strong communication plan also defines message content. For example, a weekly sponsor update should not include long technical details. It should include baseline progress (schedule and cost), key risks, open decisions, and support needed from the sponsor. In contrast, a team coordination meeting may focus on short-term tasks, dependencies, and blockers. For user stakeholders, the focus is often on usability feedback, training, and what changes to expect.

Communication must be two-way. Sending updates is not enough if the project needs feedback, approvals, or testing participation. For this reason, the plan should include response expectations, such as "approval required within three working days" or "feedback collected within one week after the demo." It should also define escalation rules when responses are delayed, for example: first reminder by the PM, then escalation to the sponsor if approval is blocked.

Documentation is part of communication quality. Key messages should leave a trace. Status reports, meeting minutes, and action logs provide a shared reference and reduce repeated discussions. They also support accountability because they record who agreed to what and when. For student projects, this is a useful professional habit that improves clarity and fairness in teamwork.

For the Service Desk Web Portal v1.0, the communication plan should include weekly sponsor updates, short team coordination meetings, and planned reviews such as requirements approval, prototype review, MVP demo, UAT planning, and release approval. The plan should also include a channel for urgent issues (for example, a dedicated chat group) and a shared document location where the latest versions of the schedule, change log, and risk register are stored. This structure supports alignment and smooth acceptance at the end of the project (see Table 8.1).

Table 8.1 - Communication plan example for Service Desk Web Portal v1.0

Audience	Information	Frequency	Channel	Owner	Purpose
Sponsor (Head of IT)	Weekly status (progress, risks, budget, decisions needed)	Weekly	Email and short meeting	PM	Visibility and approvals
Service Desk manager	Requirements, workflow decisions, demo feedback	Weekly	Meeting or demo	PM	Align process and priorities
Project team (PM/Dev/QA/UX)	Tasks, blockers, next steps	3× per week	Standup (online or in-person)	PM	Coordination and speed
IT admin	Hosting, deployment needs, access issues	Weekly (or as needed)	Direct chat and meeting	Dev or PM	Technical readiness
Security officer	Access control, data handling, key risks	At milestones	Review meeting	PM	Compliance sign-off
Support agents (L1/L2)	Demo updates, training, UAT tasks	Bi-weekly	Demo and chat	PM	Adoption and feedback
Student/staff reps	Pilot/UAT instructions, feedback request	During UAT	Email and survey	PM	Usability feedback and acceptance

Learning objectives

By the end of this section, you can:

- plan project communications for different stakeholders;
- select appropriate communication channels and frequency;
- define a standard status reporting format.

Practical tasks:

1. Create a communication plan for eight stakeholders or stakeholder groups.
2. Define a standard status report format and frequency.

Table 8.2 - Communication plan

Audience	Information	Frequency	Channel	Owner	Purpose

Checklist:

- Plan covers sponsor, users, and team.
- Frequencies are realistic (weekly, biweekly).

8.2 Meeting Agenda, Minutes, and Action Log.

Project meetings support coordination and decision-making, but they can waste time without structure. This section introduces three simple documents that keep meetings efficient and traceable: a meeting agenda, meeting minutes, and an action log.

A *meeting agenda* defines the purpose of the meeting and lists topics in the correct order. It should include time limits, required participants, and expected outputs such as a decision, approval, or a list of next steps. A clear agenda helps participants prepare and keeps the meeting focused on outcomes.

Meeting minutes provide a written record of what was discussed and agreed. Minutes should capture key decisions, risks or issues raised, and action items with owners and deadlines. Minutes do not need to be long. They should be clear enough that a person who did not attend can understand what was decided.

An *action log* is a running list of tasks that come out of meetings. It tracks what must be done, who owns the action, the due date, and the current status. The action log supports accountability and reduces repeated discussions, because decisions and responsibilities remain visible.

For the Service Desk Web Portal v1.0, these tools support reviews such as requirements approval, UI prototype review, MVP demo, test readiness review, and release approval. Consistent agendas, minutes, and action tracking help the team deliver within the 10-week timeline and reduce misunderstandings with stakeholders (see Table 8.3).

Table 8.3 - Meeting agenda example for Service Desk Web Portal v1.0

Field	Example content
Meeting title	Requirements approval meeting (SD portal v1.0)
Date/time	Week 2, Day 5, 14:00 - 15:00
Participants	Sponsor, PM, Service Desk manager, Lead developer, UX
Agenda items	1) Confirm project goals and scope boundaries 2) Review top requirements and priorities 3) Confirm out-of-scope items 4) Confirm milestones for Weeks 3 - 6
Decisions needed	Approve requirements list; approve scope statement;

	confirm MVP content
Preparation required	Read requirements list and scope statement; prepare questions and missing items list

Learning objectives

By the end of this section, you can:

- prepare a clear agenda for a project meeting;
- document meeting outcomes using minutes;
- track actions and responsibilities in an action log.

Practical tasks:

1. Create an agenda for a weekly project meeting.
2. Create a minutes template and an action log with five actions.

Table 8.4 - Meeting agenda

Meeting title	
Date/time	
Participants	
Agenda items	
Decisions needed	
Preparation required	

Table 8.5 - Action log:

Action	Owner	Due date/week	Status	Notes

Checklist:

- Minutes capture decisions and owners.
- Actions have due dates and status.

8.3 Stakeholder engagement plan

Stakeholder engagement should start early and continue throughout the project. Many project failures happen not because the technical solution is impossible, but because key people were not involved at the right time. The engagement plan reduces this risk by making participation predictable: who will be involved, when they will be involved, and what output is expected from each interaction (approval, feedback, decision, sign-off, adoption).

A good engagement plan also prevents two common problems. The first is “late surprise,” when a stakeholder sees the solution near the end and rejects it because expectations were not aligned. The second is “silent resistance,” when stakeholders do not openly disagree but also do not support the project, delay approvals, or avoid adoption. Regular, purposeful engagement makes these risks visible and manageable.

Engagement actions must match stakeholder needs and motivation. Some stakeholders need short, high-level updates, because they focus on budget, risk, and deadlines. Other stakeholders need hands-on involvement, because they care about usability, workload impact, or process changes. The plan must also consider stakeholder availability. If a sponsor can only meet once every two weeks, the project should schedule decision points in advance and prepare clear pre-reads to reduce delays.

Each engagement action should be written as a small “micro-contract” with four elements:

1. purpose (why the interaction is needed);
2. format (meeting, demo, review, workshop, survey);
3. expected output (approval, decision, feedback list, training completion);
4. owner (who prepares, who leads, who records outcomes).

This makes engagement measurable and easier to manage, instead of vague statements such as “keep stakeholders informed.”

The engagement plan should connect to the communication plan but should not duplicate it. The communication plan answers “how we send information.” The engagement plan answers “how we involve people in decisions and validation.” In practice, engagement includes checkpoints such as requirement workshops, prototype reviews, acceptance tests, training sessions, and go-live readiness reviews. These events reduce uncertainty and improve acceptance because stakeholders see progress and shape the result.

For the Service Desk Web Portal v1.0 (see Table 8.6), engagement should prioritize the sponsor for approvals, the Service Desk manager for workflow decisions, and support agents and user representatives for usability feedback and UAT. A practical engagement approach is to run short, structured sessions: a requirements workshop in Week 1–2, a clickable prototype review in Week 3, an MVP demo in Week 5–6, UAT preparation and training in Week 8–9, and release acceptance in Week 10. After each session, the team should record the outputs in meeting minutes and update the action log. This ensures traceability and prevents repeated discussions.

Finally, the plan should include one simple indicator of engagement health. For example, track attendance at key reviews, response time for approvals, or percentage of feedback items closed before UAT. These indicators help the project manager detect weak engagement early and apply corrective actions, such as a focused sponsor check-in or a targeted user feedback session.

Table 8.6 - Engagement plan example for Service Desk Web Portal v1.0

Stakeholder	Current level	Desired level	Engagement action	When / output

Sponsor (Head of IT)	Informed	Approver	Weekly status review and decision points	Weekly / approvals and fast decisions
Service Desk manager	Consulted	Co-owner	Requirements workshop and workflow review	Week 2 - 3 / feedback and sign-off
Support agents (L1)	Informed	Active user	Demo sessions and workflow test	Week 6 - 9 / feedback and adoption readiness
Student representative	Unaware	Tester	Usability test and UAT participation	Week 9 / usability feedback and acceptance notes
Staff representative	Unaware	Tester	UAT briefing and feedback survey	Week 9 / UAT feedback and acceptance notes
IT admin	Consulted	Owner (deployment)	Deployment planning and release checklist review	Week 8 -10 / deployment readiness and go-live support

Learning objectives

By the end of this section, you can:

- plan stakeholder engagement actions;
- link engagement activities to timing and expected outcomes;
- support stakeholder approval and adoption.

Practical tasks:

1. Write engagement actions for at least six stakeholders. Include timing and expected output (approval, feedback, adoption).

Table 8.7 - Engagement plan

Stakeholder	Current level	Desired level	Engagement action	When / output

Checklist:

- Actions target high-power stakeholders first.
- Each action has timing and a measurable output.

CHAPTER 9. RISK MANAGEMENT

9.1 Risk register

Every project includes uncertainty. A *risk* is an uncertain event or condition that may affect project objectives such as scope, schedule, cost, quality, or stakeholder acceptance. Risks can have negative effects (threats) or positive effects (opportunities), but most project risk management focuses on threats that can cause delays, extra cost, or reduced quality.

A *risk register* is the main document for risk management. It records identified risks in a clear and structured format. A basic risk register includes a risk description, the cause, the possible impact, a probability and impact rating, a planned response, and an assigned risk owner. The register supports early action and regular review during status reporting.

Risk identification should be systematic. It can start from project documents such as the scope statement, schedule, assumptions, and stakeholder analysis. Team brainstorming and lessons learned from similar projects also help. Risks should be written clearly and in one sentence, so they are easy to understand and track.

Using *risk categories* improves completeness. Categories help the team think broadly and avoid missing important risks. Common categories include:

- *Scope:* unclear requirements, scope growth, missing deliverables.
- *Schedule:* unrealistic timelines, late feedback, resource conflicts.
- *Cost:* underestimated costs, unexpected purchases.
- *Technical:* integration issues, performance limits, security gaps.
- *People:* limited availability, skill gaps, team turnover.
- *External:* policy changes, vendor delays, infrastructure constraints.

The example for the Service Desk Web Portal v1.0 is shown in Table 9.1.

Table 9.2 - Risk register example for Service Desk Web Portal v1.0

ID	Risk	Category	Cause	Impact	Owner	Response	Status
R1	Late stakeholder feedback	Schedule	Slow reviews	Delay and rework	PM	Set deadlines; escalate	Open
R2	Scope creep	Scope	Extra feature requests	Timeline risk	PM	Use change control; defer to v2.0	Open
R3	Login integration issues	Technical	Access not ready	Block core functions	Dev	Early check; IT admin support	Open

R 4	High defects in testing	Quality	Unstable builds	Go-live risk	QA	Prioritize critical fixes	Open
R 5	Deployment access delay	External	Hosting approvals	Go-live delay	IT admin	Confirm access early; checklist	Open
R 6	Low user adoption	Stakeholder	Users avoid portal	Benefits not achieved	SD manager	Training; pilot support	Open

Learning objectives
By the end of this section, you can:
- identify and categorize project risks;
- assign ownership for risk monitoring;
- support proactive risk management.

Practical tasks:
1. Identify at least 12 risks for the portal and assign a category.
2. Assign a risk owner for each risk.

Table 9.2 - Risk Register

ID	Risk	Cate-gory	Cause	Impact	Owner	Re-sponse	Status

Checklist:
- Risks are specific and linked to causes.
- Each risk has an owner and an initial response.

9.2 Risk Analysis (Probability-Impact Matrix)

After risks are identified, they must be prioritized. Risk analysis helps determine which risks require immediate action and which risks only require monitoring. A simple and widely used method is the *probability–impact matrix*.

Probability describes how likely a risk is to occur. *Impact* describes how strongly the risk can affect project objectives such as schedule, cost, scope, quality, or stakeholder acceptance. Both factors are usually rated as **High (H), Medium (M), or Low (L)** [17] (see Table 9.1). The matrix combines the two ratings and produces an overall risk level. Risks with high probability and high impact receive the highest priority and need a planned response and active tracking.

The probability–impact matrix supports consistent decisions. It also helps explain risk priorities to stakeholders, because the ranking is clear and visible. Risk ratings should be based on evidence, when possible, such as experience from similar projects, known constraints, and team input. Ratings should be reviewed during the project, because probability and impact can change over time.

Table 9.1 - Probability and impact matrix

Inherent Risk				
		Impact		
		Low	Medium	High
Probability	High	Medium	High	High
	Medium	Low	Medium	High
	Low	Low	Low	Medium

Hint. Consider the intersection of Impact and Probability; for instance, if the probability of a situation occurring is LOW, but the impact, when it does happen, is HIGH, then the intersection of these two factors is deemed MEDIUM. In general, this risk is classified as Medium.

Table 9.2 - Probability chart

Probability		
	Qualitative	Quantitative (if measurable)
Low	Very low chance of risk occurring.	Less than <10% chance of risk occurring.
Medium	Medium chance of risk occurring.	10%-49% chance of risk occurring.
High	High chance of risk occurring.	50%-100% chance of risk occurring.

Table 9.3 - Impact chart

Types of Impact	Low	Medium	High
Financial	Low financial impact, costing the company $0-$14,000	Medium financial impact, costing the company $15,000 - $29,000	High financial impact, costing the company $30,000 or more
Operational	Low impact to project operations, causing delays of a few days to a few weeks	Medium impact to project operations, with potential to delay project by a	High impact to project operations, with potential to cause project failure

		month or more	
People	Low impact to employee attrition, with 5%+ of employees quitting	Medium impact to employee attrition, with 25%+ of employees quitting	High impact to employee attrition, with 50%+ employees quitting

Figure 9.1 shows a qualitative analysis of risk probability and impact.

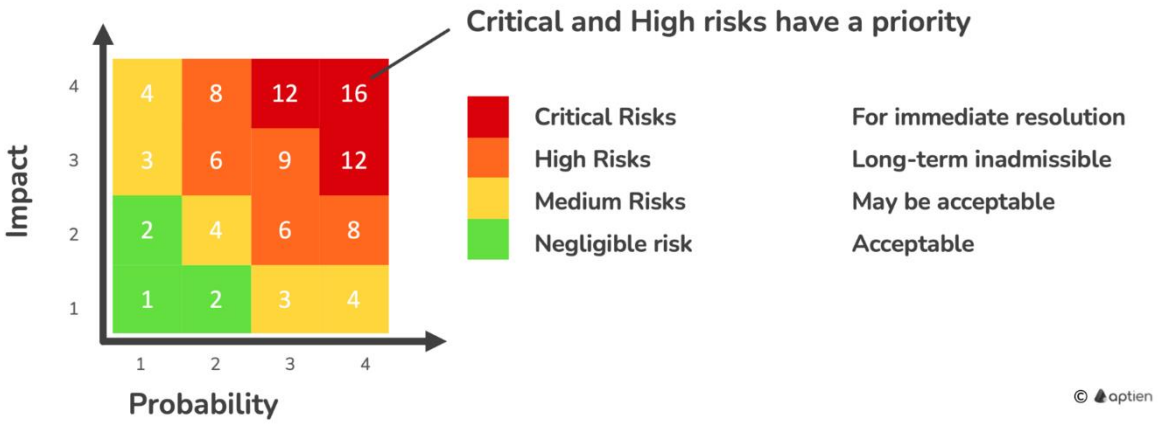


Figure 9.1 - Qualitative Analysis of Risk Probability and Impact Matrix

After assigning probability and impact ratings, each risk receives a priority score. A simple method uses a numeric scale (for example: Low = 1, Medium = 2, High = 3) and calculates:

$$\text{Priority score} = \text{Probability} \times \text{Impact} \tag{9.1}$$

Next, list all risks in the risk register, calculate the priority score for each one, and sort risks in descending order. This ranking highlights the risks that require the most attention.

To focus effort, select a threshold for “critical” risks. The threshold can be defined as the top group (for example, the 5 - 10 highest scores) or as any risk with a score above a chosen value (for example, ≥ 6 on a 1 - 9 scale). The result of the qualitative analysis is a short list of priority risks and a clear decision on how to handle them (mitigate, avoid, transfer, or accept).

When quantitative risk evaluation is feasible, the same critical risks can be assessed with numeric impact estimates such as expected delay (days) or expected cost exposure. A simple format includes best/most likely/worst case impact or an expected value estimate. Example formats for quality and control metrics appear in Table 9.4.

Table 9.4 - Quantitative risk analysis

Rank	Risk	Probability	Impact	Priority	Actions

1	Delay in delivery of a software tool	50%	$10,000	$5,000	Weekly monitoring; consider the possibility of temporary release.
2	Changes in requirements	40%	$7,000	$2,800	Monitor progress every two weeks.
3	Aggressive performance requirements	30%	$9,000	$2,700	Prototypes; performance testing.
4	Loss of team members	5%	$50,000	$2,500	Pair programming.
5	User interface design issues	5%	$1,000	$50	Development using the Model-View-Controller (MVC) framework.

After the team identifies the critical risks, the project must define mitigation measures for each one. Four main response methods can be used. These responses often require extra time and budget, and in extreme cases they can remove the expected project benefits. Table 9.4 shows example of Service Desk Web Portal v1.0.

Table 9.4 - Risk rating example for Service Desk Web Portal v1.0

Risk ID	Probability (H/M/L)	Impact (H/M/L)	Priority	Notes
R1 (Late stakeholder feedback)	H	H	High	Approval delays affect all milestones
R2 (Scope creep)	H	M	High	Extra features threaten the 10-week plan
R3 (Login integration issues)	M	H	High	Login blocks core user access
R4 (High defects in testing)	M	H	High	Risk of release delay or poor quality
R5 (Deployment access delay)	M	M	Medium	Hosting approvals may slip
R6 (Low user adoption)	L	M	Low	Training and support reduce impact

Learning objectives

By the end of this section, you can:

- assess risk probability and impact;
- prioritize risks based on their severity;
- justify focus on the most critical risks.

Practical tasks

1. Rate probability and impact for each risk (H/M/L).
2. Select top five risks and explain why they are top priority.

Table 9.4 - Risk Rating

Risk ID	Probability (H/M/L)	Impact (H/M/L)	Priority	Notes

Checklist:

- Top risks have clear reasoning.
- Ratings look consistent with the case constraints.

9.3 Risk response plan and risk owner assignment

After risk analysis, the project must define how each key risk will be handled. A **risk response plan** describes the chosen response strategy and the concrete actions required. The plan ensures that risk management becomes part of project work, not only a list in a document. Each priority risk should have a response, a due time, and an owner who tracks it.

Common response strategies include (see Figure 9.2):

- **Avoid:** change scope, approach, or requirements to remove the risk source.
- **Mitigate:** reduce probability or impact through preventive actions.
- **Transfer:** move responsibility to another party, often through a contract or service agreement.
- **Accept:** take no preventive action but monitor the risk and prepare a fallback plan if needed.

Figure 9.2 – Risk mitigation types

A strong response plan focuses on practical measures. Actions should be specific and measurable. For example, “reduce risk of late feedback” can become “set review deadlines, schedule approval meeting, and escalate after two working days.” In many IT projects, mitigation also includes early prototypes, early integration tests, and clear acceptance criteria.

Each risk must have a *risk owner*. The risk owner is responsible for monitoring the risk, implementing the planned response actions, and reporting status. The owner is not always the project manager. Ownership should match control. For example, a technical lead can own integration risks, while a Service Desk manager can own adoption risks. Examples of possible compensatory measures during a software project are presented below in Table 6.2.

Table 9.5 - Examples of risk response measures in a software project

Risk	Actions
Staffing Issues	Hiring talented personnel; cross-training; personal contracts; team building; aligning roles and skills.
Unrealistic Schedule and Budget	Detailed schedule and cost estimation by milestones; design cost orientation; incremental development; reuse of existing modules; requirements revision.
Performance Issues in Real-Time Mode	Simulations; prototypes; performance testing; parameter tuning and optimization; modeling.
Continuous Changes in Software Requirements	Increasing thresholds for accepting new changes; incremental development (deferring new requirements to later stages).
Development of an Incorrect User Interface	Developing prototypes, using scenarios, task analysis for users, and actively involving users in development.

Monitoring and control in risk management ensure that identified risks remain under observation throughout the project and that response strategies remain effective. This stage allows the project team to track existing risks, identify new ones, and evaluate whether previously selected measures still serve their purpose.

The process usually involves three main activities:

- Tracking risks: The team observes the status of each risk and records any changes in probability or impact.
- Reviewing responses: The effectiveness of chosen strategies is assessed, and adjustments are made if they no longer reduce threats as planned.
- Detecting new risks: The environment and project progress are examined regularly to reveal emerging risks that may require attention.

Effective monitoring relies on tools such as risk registers, progress reports, and regular review meetings. Clear communication between project members and stakeholders plays a central role, as timely information helps in quick decision-making. Ultimately, continuous monitoring and control create a proactive approach that improves the chances of project success and reduces the likelihood of unexpected failures.

Monitoring and control represent the stage where the risk list is revised and assessments are updated. Outdated measures must be replaced with more effective ones

when required. At this point, realized risks are recorded, contingency plans are put into action, and the overall project plan is updated to reflect the new situation.

A key element of this process is clear communication. The project team must exchange essential information, make timely decisions, and ensure that every critical risk has an assigned owner. In addition, the communication flow and the target audience for risk-related updates must be defined to maintain accountability and transparency.

The general structure of monitoring and control activities is summarized in Table 9.6.

Table 9.6 - Project Risk Communication Plan

Name	Frequency	Method of Communication	Audience	Responsible Party
Project Team Meeting	Daily	Meeting	Project Team	Project Manager
Stakeholder Updates	Monthly	Email with a report	Stakeholders	Project Manager
Updates to the Steering Committee	Every two weeks	Meeting and risk status presentation	Steering Committee	Project Manager and Lead Developer
Meeting with Contractor	Weekly	Online meeting	Project Manager, Lead Developer, and Contractor Representative	Lead Developer

In summary, the planning stage focuses on the identification of the most critical risks, the definition of their priority, and the preparation of a compensation plan as a necessary condition for project success. Since risks have a probabilistic character, regular monitoring and control of their status are essential to ensure timely activation of the plan when required. Example of Service Desk project is shown in the Table 9.7.

Table 9.7 - Response plan example (top risks) for Service Desk Web Portal v1.0

Risk ID	Preventive action	Fallback action	Owner	Trigger	Due week
R1 (Late feedback)	Set review deadlines and schedule approval meetings	Escalate to sponsor and approve using baseline assumptions	PM	No response within 2 working days	Week 2

R2 (Scope creep)	Confirm out-of-scope list and enforce change request form	Move request to v2.0 back-log and keep v1.0 scope	PM	New feature request af-fects base-line	Week 3
R3 (Login integration)	Run early inte-gration test with IT admin	Use temporary test login for development and UAT	Dev and IT admin	Login fails in integra-tion test	Week 4
R4 (High defects)	Start testing early and run weekly defect triage	Freeze new features and focus on criti-cal fixes	QA	Critical de-fects > 5 in a week	Week 8
R5 (Deployment access)	Confirm hosting and access ap-provals early	Move go-live date by 1 week and run pilot on staging	IT admin	No produc-tion access by Week 9	Week 9

Learning objectives

By the end of this section, you can:

- define response strategies for key project risks;
- distinguish between preventive and fallback actions;
- prepare practical plans to reduce risk impact.

Practical tasks:

1. For the top five risks, write a response plan with one preventive action and one fallback action.

Table 9.8 - Response plan (top risks)

Risk ID	Preventive action	Fallback action	Owner	Trigger	Due week

Checklist:

- Actions are practical and fit the timeline.
- Triggers describe when fallback starts.

9.4 Contingency and reserve plan

A risk response plan defines preventive and fallback actions, but some risks still require extra time or budget. A contingency plan states what the project will do if a risk

occurs. It links each top risk to a practical fallback action and a clear trigger. A contingency plan supports faster decisions because the team does not invent a solution under pressure.

A reserve plan sets aside schedule or budget to handle uncertainty. A schedule reserve (buffer) protects key milestones when risks cause delays. A cost contingency reserve covers expected extra costs such as additional testing effort, extra tool usage, or small technical purchases. Reserves must not appear as "free space." They must link to specific risks and use clear approval rules.

For the Service Desk Web Portal v1.0, common reserve drivers include late stakeholder feedback, login integration issues, high defect levels during testing, and deployment access delays (see Table 9.9). A basic reserve plan improves control and helps the sponsor understand how the team protects the 10-week delivery target.

Table 9.9 – Example of reserves for Service Desk Web Portal v1.0

Field	Example content
Schedule buffer (days/weeks)	5 working days (1 week) buffer before go-live
Cost contingency (%)	10% of the estimated budget
Which risks justify reserves	R1 late feedback; R3 login integration issues; R4 high defects; R5 deployment access delay
Approval rule to use reserves	PM can use up to 20% of reserve for approved corrective actions; sponsor approval required for any use above that or for any baseline change

Learning objectives

By the end of this section, you can:

- define time and cost reserves;
- link reserves to identified project risks;
- explain how reserves support risk response and control.

Practical task:

1. Define time and cost reserves and link reserves to the top risks.

Table 9.10 - Reserves

Schedule buffer (days/weeks)	
Cost contingency (%)	
Which risks justify reserves	
Approval rule to use reserves	

Checklist:

- Reserve size is realistic and justified.
- Use of reserves has an approval rule.

CHAPTER 10. PROCUREMENT AND CONTRACTS

10.1 Make-or-buy decision.

Some project needs can be delivered by the internal team, while others are better purchased or outsourced. A make-or-buy decision compares these options and supports a clear, documented choice. The decision should consider cost, schedule, available skills, quality expectations, security requirements, and long-term support.

Make means the team builds or performs the work internally. This option provides more control over features and changes, but it can require more time and specialist skills. *Buy* means the project purchases a product or service or uses an external provider. This option can reduce delivery time, but it may introduce vendor dependence, recurring costs, and integration limits.

A simple make-or-buy decision works best when it focuses on a small number of criteria and supports the project constraints. The decision should be recorded with a short justification so that stakeholders understand the trade-offs. For the Service Desk Web Portal v1.0, typical make-or-buy items include hosting, email/SMS notification service, UI components, and testing tools (see Table 10.1). The chosen option should protect the 10-week timeline and ensure stable delivery.

Table 10.1 - Make-or-buy example for Service Desk Web Portal v1.0

Item	Option (make/buy)	Pros	Cons	Estimated cost	Decision
Hosting environment	Buy	Fast setup; reliable uptime; managed backups available	Monthly fee; vendor limits	120 (3 months)	Buy
Email notifications	Buy	Quick integration; stable delivery; reporting available	Service limits; recurring cost	20 - 30 (project period)	Buy

Learning objectives

By the end of this section, you can:

- analyze simple make-or-buy decisions;
- compare internal and external options;
- support procurement decisions based on project needs.

Practical tasks:

1. Choose two items for a make-or-buy decision (example: hosting, SMS service, UI design). Compare options.

Table 10.2 - Make-or-Buy

Item	Option (make/buy)	Pros	Cons	Estimated cost	Decision

Checklist:

- Criteria include time and risk, not only cost.
- Decision matches scope and constraints.

10.2 Procurement plan and vendor criteria

A procurement plan defines what the project will purchase, when each item is needed, and how vendor selection will work. It supports cost control and prevents delays caused by late purchases or unclear responsibilities. A procurement plan also improves transparency because stakeholders can see planned spending and selection rules in advance.

Vendor criteria support fair and consistent selection. Criteria define what matters most for the project, such as price, reliability, security, delivery time, and support. A simple scoring model assigns weights (for example: price 40%, reliability 30%, support 30%) and helps compare options objectively. The plan should also define the selection method, such as direct purchase, simple quote comparison, or approved vendor list.

For the Service Desk Web Portal v1.0, procurement is usually small but still important. Items such as hosting, domain services, email delivery, and testing tools can affect schedule and release readiness. Clear criteria and owners reduce last-minute decisions and improve project control.

Table 10.3 - Procurement plan example for Service Desk Web Portal v1.0

Item	Need date	Budget	Selection method	Criteria	Owner
Domain registration	Week 1	15	Direct purchase	Price 40%, reliability 30%, support 30%	PM
Cloud hosting (VM)	Week 2	120	Compare 2-3 providers	Price 40%, reliability 40%, support 20%	IT admin
Email notification service	Week 5	30	Compare 2 providers	Reliability 40%, price 30%, limits 30%	Developer
Backup storage	Week 6	30	Use hosting add-on	Reliability 50%, price 30%, recovery options 20%	IT admin

Testing/issue tracking tool	Week 3	25	Approved tool list	Usability 40%, price 30%, reporting 30%	QA
Training materials (printing)	Week 10	10	Local supplier	Price 50%, delivery time 30%, quality 20%	PM

Learning objectives

By the end of this section, you can:

- identify items that require procurement;
- define clear supplier selection criteria;
- apply weighted scoring to compare procurement options.

Practical tasks:

1. List items to procure (at least five).
2. Define selection criteria and scoring weights (example: price 40%, reliability 30%, support 30%).

Table 10.4 - Procurement plan

Item	Need date	Budget	Selection method	Criteria	Owner

Checklist:

- Need dates match the schedule.
- Criteria are measurable and weighted.

10.3 Contract types overview

Projects often involve external suppliers for services, tools, or outsourced work. A contract type defines how payment works and how risk is shared between the buyer and the vendor. The right contract type supports control, cost predictability, and timely delivery.

Fixed price contracts set a defined total price for a clearly described scope. This type works best when requirements are stable and deliverables are easy to verify. Fixed price provides cost predictability, but it can reduce flexibility. If scope changes, the vendor usually requests a change order, which can increase cost or delay delivery. Clear acceptance criteria are essential.

Time and materials (T&M) contracts pay for the supplier's time (hourly or daily rate) plus approved expenses. This type suits work with uncertain scope or evolving requirements, such as configuration, integration, or short-term support. T&M offers

flexibility, but it requires strong control. The project must define weekly limits, clear reporting, and approval rules to prevent budget growth.

Cost reimbursable contracts pay the supplier's actual costs plus a fee or profit. This type suits research-like work or projects with high uncertainty, where accurate fixed pricing is not realistic. Cost reimbursable requires close oversight, detailed reporting, and clear rules for allowable costs. It often involves approval checkpoints and audit-style control.

Learning objectives

By the end of this section, you can:

- select appropriate contract types for procurement items;
- explain contract choices based on risk and uncertainty;
- support clear and justified procurement decisions.

Practical tasks:

1. For two procurement items, select a contract type and explain the reason in one paragraph each.

Checklist:

- Contract type matches scope certainty and risk.
- Explanation includes a control approach.

CHAPTER 11. PROJECT EXECUTION AND CONTROL

11.1 Status Report

A weekly status report is a short document that shows project health in a consistent way. It helps stakeholders understand progress without long meetings and supports early correction when problems appear. A good status report stays factual and focuses on what changed since the last update, what is at risk, and what support is needed.

The report usually includes an overall status (Green/Yellow/Red), a short progress summary, and a comparison of planned versus actual work. It also highlights the top issues and risks and lists next week priorities. The final section should clearly state any requests or decisions needed from the sponsor or stakeholders. This structure improves transparency and reduces delays, because decision makers can act quickly.

For the Service Desk Web Portal v1.0, a weekly report supports a fast 10-week timeline. It can track requirements approval, prototype progress, development status, and readiness for testing and deployment (see Table 11.1). It also provides a simple record of decisions and changes, which supports project control.

Table 11.1 - Weekly status report example for Service Desk Web Portal v1.0 (Week 3)

Field	Example content
Week / date	Week 3 (Day 15–21)
Overall status (Green/Yellow/Red)	Yellow
Progress summary	UI prototype completed; architecture draft prepared; authentication approach confirmed with IT admin.
Planned vs actual (key)	Planned: UI prototype approval in Week 3. Actual: prototype review done, final approval pending.
Top issues	1) Sponsor feedback on prototype delayed. 2) Category list for tickets not finalized.
Top risks	R1 late feedback (High); R3 login integration issues (Medium).
Next week plan	Approve UI prototype; finalize database schema; start authentication module and ticket creation form.
Requests / decisions needed	Sponsor approval of UI prototype; confirm final ticket categories and priorities.

Learning objectives

By the end of this section, you can:

– create a clear weekly project status report;
– report progress using planned data as a reference;
– communicate project status in a structured and concise format.

Practical tasks:

1. Create a one-page weekly status report template and fill it for Week 3 (use your plan as reference).

Table 11.2- Weekly status report

Week / date	
Overall status (Green/Yellow/Red)	
Progress summary	
Planned vs actual (key)	
Top issues	
Top risks	
Next week plan	
Requests / decisions needed	

Checklist:

– Report includes at least one metric (schedule, defects, scope items).
– Issues and decisions have owners and due dates.

11.2 Issue log and decision log

An *issue* is a current problem that already exists and requires action. Issues differ from risks. A risk is uncertain, but an issue is real and active. An issue log provides a single place to record problems, assign owners, set priorities, and track resolution. This tool prevents missed actions and supports clear status reporting.

A *decision log* records key project decisions and the reason for each decision. It protects the project from repeated debates and confusion, especially when stakeholders change their opinion later. The decision log should capture what was decided, who approved it, and the date. It should also link the decision to the scope baseline, schedule, or change request when relevant.

GitHub as the most popular version control system.

Version control systems (VCS) are key tools in software development for managing changes to source code, enabling teams to collaborate and track different code versions efficiently. There are two main types of version control systems [29]:

1. Centralized Version Control Systems (CVCS) rely on a single central server that stores all project files and version history. Developers must connect to this server to access the code, check out files to work on them, and then commit their changes back to the server. This provides a simple, consolidated management approach. Examples include Subversion (SVN) and Perforce.

2. In a Distributed Version Control System (DVCS), every developer has a complete copy of the entire repository, including its full history, on their local machine. This allows for more flexibility and resilience. Developers can commit changes locally and then sync them with other team members. The lack of a single point of failure means work can continue even if the central server is down. Git, Mercurial, and Bazaar are all examples of distributed version control systems.

Table 11.3 - Comparative review of Version Control Systems

Version control system	Market share	Statistics of use	Reason of popularity
Subversion (SVN)	2-4%	Rarely for corporate old project	The simplicity of the model and the centralized approach that some organizations prefer due to the ease of access control and oversight. Industries with strict compliance requirements sometimes prefer SVN due to its single point of control.
Mercurial	<1%	Very seldom for old project	Simplicity and performance. Some projects, such as the original repositories for Python and Mozilla Firefox, used Mercurial before switching to Git..
Git (GitHub, GitLab)	90-95%	~93%	Distributed, fast, branchable, and widely adopted by major platforms like GitHub, GitLab, and Bitbucket, Git's flexibility allows it to handle projects of any size, making it a favorite among small teams and large enterprises alike.
Perforce	1-2%	Corporate specific	High performance and reliable processing of large data sets, widely used in game development, digital media and enterprise environments.
Bazaar	<1%	Very seldom	Although Bazaar was initially popular for its ease of use and decentralized nature, it did not receive significant updates, leading to its decline.

A Distributed Version Control System (DVCS) operates on a peer-to-peer model, where every developer has a complete copy of the repository on their local machine. The main repository is typically stored on a hosting service, such as GitHub or GitLab.

In a DVCS, developers can work on any file simultaneously without needing to lock it, because they are modifying a local copy. When a developer is ready, they push

their changes to the master repository on the hosting service. The version control system then automatically detects any conflicts between the new changes and those already in the main repository, allowing the developer to resolve them before the changes are integrated [23].

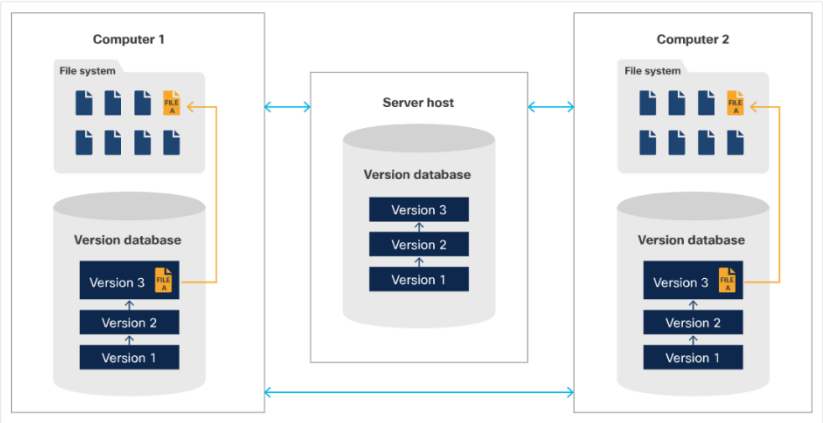


Figure 11.1 - Distributed Version Control System (DVCS)

Git is an open-source, distributed version control system that has become a standard in software development. Unlike older systems, it is unique in how it handles data.

Git's key features:

– Distributed Model: Git is a peer-to-peer system. This means that every developer has a complete copy of the repository on their local machine, allowing for offline work and improved resilience.

– Client Installation: To use Git, the client software must be installed on your machine. It is available for all major operating systems, including macOS, Windows, and Linux/Unix.

– Snapshot-Based Storage: A key difference between Git and other version control systems is its data storage method. While other systems store data as a set of changes (deltas) from a previous version, Git stores a complete snapshot of the entire repository at each commit. This makes operations like viewing history and branching faster.

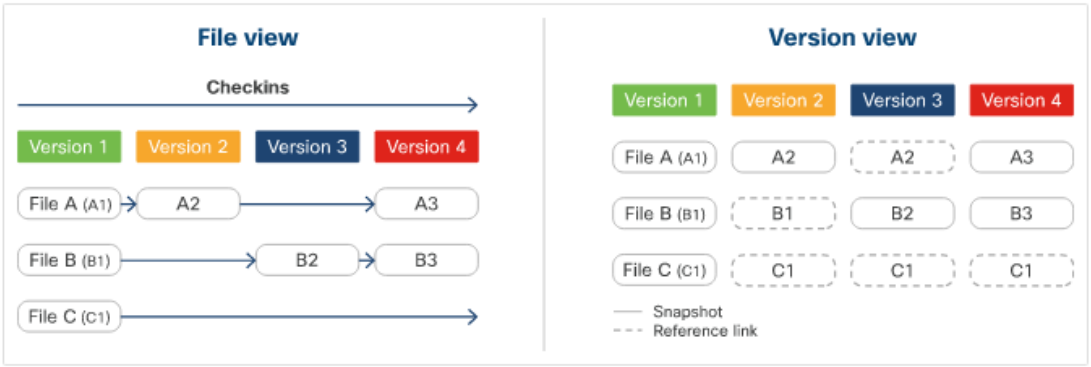


Figure 11.2 - Git version control feature

Usage trends and analytics. There is a growing adoption of distributed version control systems (DVCS), as the move towards DVCS, led by Git, is driven by the need

for greater collaboration, improved branching and merging capabilities, and autonomous operation [36]. DVCS offers greater flexibility and is more resilient to single points of failure compared to centralized systems and offers easy integration with DevOps and CI/CD pipelines.

Version control systems are an integral part of DevOps practices and CI/CD pipelines. Most modern CI/CD tools such as Jenkins, CircleCI, and GitLab CI offer native integration with Git, further increasing its adoption [36].

Cloud platforms such as GitHub, GitLab, Bitbucket, and Azure DevOps have further increased the popularity of Git [35]. These platforms provide additional services such as issue tracking, code review, and collaboration features, making them indispensable in modern development workflows.

The GitHub platform has over 100 million developers and over 330 million repositories (as of 2023 [34]), highlighting the dominance of Git and its ecosystem.

The GitLab platform is also very popular for its all-in-one DevOps platform, offering similar features to GitHub with integrated CI/CD and has a growing user base.

Enterprises are increasingly choosing Git for its scalability and compatibility with agile and DevOps methodologies [34]. Large organizations, including tech giants like Google, Facebook, and Microsoft, have standardized on Git.

For the Service Desk Web Portal v1.0, an issue log can track items such as delayed feedback, access problems, defects, and deployment constraints (see Table 11.3). A decision log can capture scope boundaries, tool choices, workflow rules, and release approvals. Together, these logs improve accountability and support project control.

Table 11.4 – Issue log example for Service Desk Web Portal v1.0

ID	Issue	Owner	Priority	Due date/week	Status
I1	Sponsor review of UI prototype is late	PM	H	Week 3	Open
I2	Ticket category list is not finalized	Service Desk manager	H	Week 3	Open
I3	Test environment access not confirmed	IT admin	H	Week 4	Open
I4	Email notification service credentials missing	Developer	M	Week 5	Open
I5	Password reset flow not defined for users	PM	M	Week 4	Open
I6	Slow dashboard load on test server	Developer	M	Week 6	Open
I7	Several critical defects found in ticket workflow	QA	H	Week 8	Open
I8	UAT participants not confirmed	PM	M	Week 9	Open

Learning objectives

By the end of this section, you can:

- document project issues in a structured issue log;
- record key decisions and their rationale;
- support transparency and consistent project control.

Practical tasks**:**

1. Create an issue log with at least eight issues and a decision log with at least five decisions for the case.

Table 11.5 - Issue log

ID	Issue	Owner	Priority	Due date/week	Status

Table 11.6 - Decision log

ID	Decision	Date/week	Owner	Reason/impact

Checklist:

- Issues have owners and due dates.
- Decisions record the reason and impact.

11.3 Change request form and change log

Projects often receive requests for new features, updates, or improvements. A change request provides a formal way to capture the proposed change and evaluate its impact before work begins. This process protects the scope baseline and supports fair decisions, especially when a change affects schedule, cost, or quality.

A *change request form* records the change description, the reason for the request, and the expected impact on scope, schedule, and cost. The form also records the final decision (approve, reject, or defer). A clear change request prevents informal agreements such as “just add it quickly,” which usually creates delays and quality problems.

A *change log* tracks all change requests in one place. It records the request ID, the change summary, who requested it, the decision, and the date. The change log supports transparency and helps the team avoid duplicate work. It also provides evidence for project reporting and closure.

For the Service Desk Web Portal v1.0, change control is important because stakeholders may request extra features during development, such as file attachments,

SMS notifications, or advanced reporting. The change request and change log ensure that any added work follows an approved decision.

Table 11.7 - Change request example for Service Desk Web Portal v1.0

Field	Example content
Change title	Add file attachment to ticket submission
Requester	Service Desk manager
Date/week	Week 5
Description	Allow users to attach one file (max 5 MB) when creating a ticket.
Reason	Users often need to share screenshots or documents to explain issues.
Impact on scope	New feature in ticket form and ticket details page; storage and security checks required.
Impact on schedule	+3 days development and testing; affects Week 6 - 7 tasks.
Impact on cost	+10 (extra storage) if needed; otherwise no change.
Decision (approve/reject/defer)	Defer to v2.0

Table 11.8 - Change log example for Service Desk Web Portal v1.0

ID	Change	Requester	Decision	Date	Notes
CR-01	Add file attachment to ticket submission	SD manager	Defer	Week 5	Plan for v2.0 backlog
CR-02	Add “Urgent” priority option	Sponsor	Approve	Week 6	Update admin settings and workflow
CR-03	Add extra report: tickets by category	Sponsor	Approve	Week 7	Include in basic reporting page

Learning objectives

By the end of this section, you can:

- document a change request with clear impact information;
- maintain a change log to track approved changes;
- support formal control of scope, schedule, and cost.

Practical tasks:

1. Fill one change request example (a new portal feature).
2. Create a change log and add three sample changes.

Table 11.9 - Change request

Change title	

Requester	
Date/week	
Description	
Reason	
Impact on scope	
Impact on schedule	
Impact on cost	
Decision (approve/reject/defer)	

Table 11.10 - Change log

ID	Change	Requester	Decision	Date	Notes

Checklist:

- Change request includes impact analysis.
- Log records decisions and dates.

11.4 Earned Value (EVM)

Earned Value Management (EVM) is a simple method to track schedule and cost performance using the project budget. It compares what was planned, what was completed, and what was actually spent. EVM supports early correction because it shows performance trends before the project finishes.

EVM uses three core values:

- PV (Planned Value): the budget value of the work planned to be completed by a given date/week.
- EV (Earned Value): the budget value of the work actually completed by that date/week.
- AC (Actual Cost): the actual cost spent by that date/week.

To build an EVM table, each work package needs an approved budget and a planned completion percentage for the selected week. Budgets and cost baselines come from Chapter 5 (cost planning). Use those values as the reference for PV and EV calculations.

Table 11.11 - EVM table example for Service Desk Web Portal v1.0 (Week 5)

Work package	Budget	Planned %	Actual % complete	PV	EV	AC
1.3.1 Authentication and roles	80	100%	90%	80	72	85
1.3.2 Ticket creation form	70	80%	70%	56	49	55
1.3.3 User dashboard	60	60%	50%	36	30	35
1.4.2 Agent queue and assignment	60	40%	30%	24	18	22
1.2.2 UI prototype	30	100%	100%	30	30	28
Total	300			226	199	225

(Interpretation: Week 5 shows both cost and schedule underperformance; CPI and SPI are below 1.0.)

CPI = EV / AC = 199 / 225 = 0.88
SPI = EV / PV = 199 / 226 = 0.88
Practical tasks:
1. Create a small EVM example for Week 5 with five work packages. Calculate PV, EV, AC, CPI, and SPI.

Table 11.12 - EVM table

Work pack-age	Budget	Planned %	Actual % complete	PV	EV

Add totals and indices below

AC (total)	
CPI (EV/AC)	
SPI (EV/PV)	
Interpretation (1-2 sentences)	

Checklist:
- PV, EV, and AC are totals across work packages.
- CPI and SPI calculations are correct.
- Interpretation states what action is needed.

11.5 Performance dashboard

A performance dashboard provides a quick view of project health. It summarizes key indicators in one place and supports faster decisions. For small projects, a simple KPI table is enough. It helps track progress against the plan, highlights problems early, and shows where corrective action is needed.

A good KPI dashboard uses a limited number of indicators that link to project objectives and baselines. KPIs must stay measurable and easy to update each week. Typical areas include schedule, scope stability, quality, risk, stakeholder engagement, and readiness for release. The dashboard should assign an owner for each KPI and record a clear action when the KPI is off target.

The dashboard is not a report for reporting's sake. It is a control tool. It helps the project manager answer three questions every week: Are we on plan? What has changed? What must we do next? To support this, each KPI should have (1) a clear definition, (2) a target or threshold, (3) a data source, and (4) an update rule (who updates it and when). If KPI definitions change during the project, the dashboard loses comparability, so definitions should stay stable after the baseline is approved.

For student projects, the main benefit is learning to connect planning documents to real control data. The schedule baseline supports schedule KPIs (for example, schedule variance). The scope baseline supports stability KPIs (for example, number of approved changes). The quality plan supports testing KPIs (for example, pass rate or open defects). The risk register supports risk KPIs (for example, number of high risks without response actions). This creates traceability between what was planned and what is actually happening.

For the Service Desk Web Portal v1.0, the dashboard can track milestone progress, testing results, defect levels, change requests, and readiness for UAT and deployment. Example KPIs include: milestone completion rate, activities completed vs planned, number of open defects by severity, test case pass rate, number of open high-priority risks, number of overdue actions from meeting minutes, number of pending change requests, and UAT readiness status. A weekly update supports transparency and keeps the team aligned.

When a KPI is off target, the dashboard should not only show "red." It should also record one short corrective action and a due date. For example, if the test pass rate drops below the target, the action may be "fix top five defects and rerun regression by Friday, owner: QA." If change requests increase, the action may be "review scope impact with sponsor and freeze non-critical features for v1.0." This turns the dashboard into a practical decision instrument rather than a passive document.

Table 11.13 - KPI dashboard example for Service Desk Web Portal v1.0 (Week 6)

KPI	Target	Current	Trend (+/0/-)	Owner	Action
Milestones on time	≥ 90% on-time	80%	-	PM	Confirm next milestone dates; escalate blockers
Schedule performance (SPI)	≥ 1.0	0.95	-	PM	Replan Week 7 tasks; protect dev focus time
Cost performance (CPI)	≥ 1.0	1.02	+	PM	Keep spending within baseline

Scope change requests	≤ 2 open	1 open	0	PM	Review impact; decide approve/defer
Critical defects open	0	2	-	QA	Fix critical defects before new features
Acceptance test pass rate	≥ 85%	78%	-	QA	Add test cases for core flow; retest fixes
Stakeholder feedback turnaround	≤ 2 working days	3 days	-	PM	Schedule fixed review slot with sponsor
Release readiness (checklist complete)	≥ 70% by Week 6	60%	0	IT admin	Prepare deployment checklist and access requests

Learning objectives

By the end of this section, you can:

- define key performance indicators (KPIs) for a project;
- present KPIs in a simple dashboard format;
- monitor project performance during execution.

Practical tasks**:**

1. Create a KPI dashboard with eight KPIs for the portal project and update it for Week 6.

Table 11.14 - KPI dashboard

KPI	Target	Current	Trend (+/0/-)	Owner	Action

Checklist:

- KPIs include schedule, quality, and stakeholder indicators.
- Each KPI has an action if the trend is negative.

CHAPTER 12. PROJECT CLOSURE

12.1 Acceptance checklist and handover plan

Project closure ensures that the project ends in a controlled and professional way. Closure confirms that agreed deliverables meet requirements and that the sponsor or client accepts the results. Without formal acceptance, projects can stay "open" with unclear responsibilities and unresolved requests.

An *acceptance checklist* provides a structured way to confirm completion. It lists key deliverables, defines acceptance criteria for each deliverable, and records acceptance status and notes. Acceptance criteria should link to the scope statement, requirements, and test results. The checklist supports a clear final decision and reduces disputes after release.

A *handover plan* transfers the product to the group responsible for ongoing operation and support. It defines what knowledge, access, and documentation must be delivered and who becomes responsible after go-live. A handover plan typically includes access credentials, deployment details, backup and recovery steps, support contacts, training, and a short "how to operate" guide. It also defines the handover date and confirms that the receiving team is ready.

For the Service Desk Web Portal v1.0, closure should confirm that the portal works as agreed, that key users complete UAT, and that the sponsor signs acceptance (see Table 12.1). The handover plan should ensure that the IT team can host, monitor, and support the portal after release, and that Service Desk staff know how to use the system in daily operations.

Table 12.1 - Acceptance checklist example for Service Desk Web Portal v1.0

Deliverable	Acceptance criteria	Accepted (Y/N)	Notes
Authentication and role access	Users can log in; role restrictions work; no access violations		
Ticket submission and ID creation	Ticket form works; ticket ID generated; data stored correctly		
User dashboard and ticket status tracking	Users can view ticket list and correct status updates		
Agent workflow (queue, assignment, status updates)	Agents can assign tickets and update status correctly		
Admin settings (categories, priorities)	Admin can manage categories and priorities without errors		
Email notifications	Notifications send for create/update events		
Basic reporting	Reports show correct counts and summary values		

Deployment package and production release	Portal deployed; smoke test passed in production		
User guide and handover notes	Documentation delivered and approved for support use		

Learning objectives

By the end of this section, you can:

– Practical tasks:
– Create an acceptance checklist for v1.0.
– Create a handover plan with owners and dates.

Practical tasks:

1. Create an acceptance checklist for v1.0.
2. Create a handover plan with owners and dates.

Table 12.2 - Acceptance checklist

Deliverable	Acceptance criteria	Accepted (Y/N)	Notes

Table 12.3 - Handover plan

Item	Owner (project)	Owner (operations)	Date	Notes

Checklist:

– Checklist covers functional and non-functional acceptance.
– Handover includes access, documentation, and support rules.

12.2 Lessons learned report

A lessons learned report captures practical knowledge from the project and turns experience into improvement. It records what worked well, what did not work well, and what should change next time. This document supports team maturity because it helps avoid repeated mistakes and reinforces effective practices.

Lessons learned should be specific and evidence-based. The report should describe real situations, not general opinions. Each lesson should include a short root cause and a clear recommendation. Recommendations should be actionable, such as adding a review checkpoint, improving a template, changing a communication rule, or adjusting estimates.

Lessons learned are most useful when grouped by theme. Common themes include scope, schedule, quality, communication, and tools. This structure helps future teams locate relevant insights quickly. The report should be completed during closure, while project details remain fresh, and it should be shared with the sponsor and team.

For the Service Desk Web Portal v1.0, lessons learned can cover scope change control, stakeholder feedback timing, testing readiness, release planning, and tool choices. A strong report helps improve future IT projects and supports consistent project governance.

Table 12.4 - Lessons learned example for Service Desk Web Portal v1.0

Area	What went well	What went wrong	Root cause	Recommendation
Scope	Clear out-of-scope list reduced extra requests	A few feature requests still appeared late	Stakeholders saw demos and requested additions	Use a change request form from Week 2 and keep a v2.0 backlog
Scope	WBS improved completeness of deliverables	Some deliverables had unclear acceptance early	Acceptance criteria defined late	Add acceptance criteria in WBS dictionary for all work packages
Schedule	Weekly milestones supported control	Prototype approval slipped by several days	Sponsor had limited availability	Schedule fixed approval time slots in advance
Schedule	Workload table prevented dev overload	Testing time was tight near the end	Testing started too late	Start QA planning early and run mid-project test cycle
Quality	Release checklist reduced missed steps	Several defects appeared during system testing	Limited early testing of core flow	Add smoke tests after each build and review defects weekly
Quality	UAT feedback improved usability	Users asked for small UI changes late	Limited user involvement early	Include one user review session during prototype stage

Communication	Weekly status reports supported transparency	Some decisions were repeated	Decisions not recorded consistently	Maintain a decision log and share it after each review
Communication	Action log improved accountability	Some actions had no clear due date	Missing ownership rules	Require owner and due week for every action item
Tools	Tracker improved defect visibility	Some tasks were not updated on time	Inconsistent team habit	Set a weekly update deadline and check during standup
Tools	Simple templates saved time	Template versions became confusing	No versioning rule	Use version number and date in file names and archive old versions

Learning objectives

By the end of this section, you can:

– document lessons learned from a completed project;
– group lessons by management theme;
– use lessons learned to improve future projects.

Practical task:

1. Write at least 10 lessons learned and group them by theme: scope, schedule, quality, communication, and tools.

Table 12.5 - Lessons learned

Area	What went well	What went wrong	Root cause	Recommendation

Checklist:

– Each recommendation is practical and specific.
– Root causes avoid blame and focus on process.

12.3 Final project report

The final project report also records how the project was managed. It confirms the baseline that was used for control (scope, schedule, and budget) and explains any significant deviations. If the project finished later or cost more than planned, the report must state the main causes and the corrective actions that were applied. This helps

stakeholders understand whether the result is acceptable and what to avoid in future projects.

A strong final report is evidence-based. It does not rely on opinions such as "everything went well." It uses simple facts that come from project documents created during execution: schedule baseline and actual finish date, total approved changes, defect statistics, acceptance results, training completion, and open issues transferred to operations. Even in small projects, a few numbers improve credibility and support better decisions.

The report should clearly separate three categories of information:

1. what was delivered (final scope and deliverables);
2. what was planned versus what happened (performance results);
3. what should happen next (handover and recommendations).

This structure keeps the report easy to read and prevents it from becoming a long history of events.

For student projects, the final report also demonstrates professional communication skills. Students practice how to report results to a sponsor or management in a short and clear form. The report must show consistency with earlier documents in the workbook. For example, the success criteria defined in the project summary should appear again in the final report with a simple "met / not met" status and short evidence. The same rule applies to major risks, changes, and acceptance criteria.

The final report should end with closure confirmation. It must state that the deliverables were accepted, the handover was completed, and the project team's responsibility ended on a specific date. If any work remains, the report should list it as follow-up actions for operations or for the next release, with an owner and target date. This prevents unfinished work from staying hidden and supports a clean transition after closure.

For the Service Desk Web Portal v1.0, the final report should confirm that the portal was delivered and accepted, summarize milestone performance, note any approved changes or deferred features, and record readiness for ongoing support after handover. It should also include a short recommendation section, for example: user feedback collection after go-live, backlog priorities for v1.1, and improvements in response time or reporting based on early usage data.

Learning objectives

By the end of this section, you can:

- summarize project results for executive stakeholders;
- highlight key outcomes, issues, and decisions;
- present a clear final project overview in a professional format.

Practical task:

1. Write a two-page executive summary final report for the case project.

Template of the final report outline:

1. Project overview (purpose and scope)
2. Key deliverables and acceptance

3. Schedule result (planned vs actual)
4. Cost result (budget vs actual)
5. Quality result (metrics and defects)
6. Stakeholder feedback and adoption
7. Major risks and issues (how resolved)
8. Recommendations for next release (v1.1)

Checklist:
- Report states results with numbers where possible.
- Report aligns with original objectives and baselines.

CONCLUSION

This project management practical workbook bridges the gap between theory and practice and helps learners apply project management concepts in realistic scenarios. It provides structured tasks, examples, and guided exercises that support core competencies such as planning, stakeholder management, role definition, risk awareness, and decision-making.

The activities provide hands-on practice with common frameworks, tools, and techniques, including stakeholder analysis, RACI matrices, and basic governance structures. These competencies support projects of different sizes and complexity levels, especially in dynamic and technology-driven environments.

The workbook promotes a systematic and reflective approach to project work and highlights clear roles, alignment with stakeholder needs, and controlled execution. It also serves as a learning resource and a practical reference for real projects. The practices in this workbook provide a solid foundation for successful project delivery and long-term professional growth in project management.

REFERENCES

[1] "ISO 21500 – Guidance on Project Management." Accessed: July 09, 2025. [Online]. Available: https://euoci.eu/2024/04/02/iso-21500-guidance-on-project-management-in-3-minutes-2/

[2] "PMBOK® Guide." Accessed: Aug. 21, 2025. [Online]. Available: https://www.pmi.org/standards/pmbok

[3] "What is the project management triangle and how can it help your team?" Accessed: Aug. 21, 2025. [Online]. Available: https://asana.com/resources/project-management-triangle

[4] "The knowledge areas of PM." Accessed: Aug. 19, 2025. [Online]. Available: https://thedigitalprojectmanager.com/)

[5] J. Phillips, *PMP Project Management Professional Study Guide*. McGraw-Hill Professional, 2003.

[6] F. F. Tsui, O. Karam, and B. Bernal, *Essentials of Software Engineering*. Jones & Bartlett Learning, 2016.

[7] A. De Marco, *Project Management for Facility Constructions A Guide for Engineers and Architects*, VIII. 2011.

[8] B. Flyvbjerg and A. Budzier, "Why Your IT Project May Be Riskier than You Think," *SSRN Journal*, 2011, doi: 10.2139/ssrn.2229735.

[9] S. P. Robbins and T. A. Judge, *Organizational behavior*, 15th ed.

[10] *A Guide to the Project Management Body of Knowledge*. Project Management Institute, 2021.

[11] C. L.Hoover, M. Rosso-Llopart, and G. Taran, *Evaluating Project Decisions: Case Studies in Software Engineering*, I. Pearson Addison-Wesley, 2009.

[12] M. Cohn, *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley Professional, 2009.

[13] C. L. Hoover, M. Rosso-Llopart, and G. Taran, *Evaluating Project Decisions: Case Studies in SE*. Addison-Wesley Professional, 2009.

[14] L. Joseph A., "How to make earned value work on your project. Paper presented at PMI® Global Congress 2012," Project Management Institute. Accessed: Aug. 22, 2025. [Online]. Available: https://www.pmi.org/learning/library/make-earned-value-work-project-6001

[15] Г. Т. Алин and Н. К. Рахимжанова, "SOFTWARE DEVELOPMENT PROJECT MANAGEMENT: PROJECT RISK MANAGEMENT," *INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES*, vol. 4, no. 1(13), pp. 38–50, Mar. 2023, doi: 10.54309/IJICT.2023.13.1.004.

[16] G. T. Alin, "SOFTWARE DEVELOPMENT PROJECT MANAGEMENT: METRICS AND QUALITY MANAGEMENT IN PROJECTS," *INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES*, vol. 5, no. 3(19), pp. 8–22, Sept. 2024, doi: 10.54309/IJICT.2024.19.3.001.

[17] N. Doerry and M. Sibley, "Monetizing Risk and Risk Mitigation," *ASNE Naval Engineers Journal*, vol. 3, no. 127, pp. 35–46, 2015.

[18] B. E. Robinson, “Hierarchy of Needs.” Accessed: Aug. 22, 2025. [Online]. Available: https://www.writersdigest.com/publishing-insights/writers-maslows-hierarchy-of-needs

[19] S. McConnell, *Software Project Survival Guide*, Microsoft Press. 1997.

[20] P. Stephen, Robbins, and L. Phillip, *Training in Interpersonal Skills: Tips for Managing People at Work*. in 6th edition. Harvard Business School Publishing, 2011.

[21] P. L. Hunsaker and A. J. Anthony, *New Art of Managing People: Person-to-person Skills*. in 6th edition. Free Press, 2008.

[22] G. H. Hofstede and G. Jan. Hofstede, *Cultures and Organizations: Software of the Mind*. New York: McGraw-Hill, 2012.

[23] L. L. Thompson, *Making the Team: A Guide for Managers.* Boston, Ma. ; London: Pearson, 2011.

[24] P. Lencioni and K. Okabayashi, *The Five Dysfunctions of a Team: An Illustrated Leadership Fable.* in 1st edition. Jossey-Bass, 2011.

[25] R. Sommer, *Personal Space; the Behavioral Basis of Design.* Prentice-Hall, 1969.

[26] J. Schuyler, *Risk And Decision Analysis In Projects: Cases In Project And Program Management Series.* Newtown Square: Project Management Institute, 2001.

[27] C. W.L. Hill and S. L. McShane, *Principles of management*. The McGraw-Hill Companies, Inc., 2008.

[28] V. Bhoola, S. B. Hiremath, and D. Mallik, “An Assessment of risk response strategies practiced in software projects,” *AJIS*, vol. 18, no. 3, Nov. 2014, doi: 10.3127/ajis.v18i3.923.

[29] T. DeMarco, *The Deadline: A Novel About Project Management*, I. in 2nd Printing. 2018.

[30] 12. Michael Keeling, *Reflections on Software Engineering: Threshold of Success.* Neverletdown, 2010.

[31] N. E. Fenton and M. Neil, “Software metrics: roadmap,” in *Proceedings of the Conference on The Future of Software Engineering*, Limerick Ireland: ACM, May 2000, pp. 357–370. doi: 10.1145/336512.336588.

[32] K. Rubin, *Essential Scrum. A Practical Guide to the Most Popular Agile Process*. in 1st edition. Addison-Wesley Professional, 2012.

[33] T. Dohmke, “100 million developers and counting.” Accessed: Aug. 26, 2025. [Online]. Available: https://github.blog/news-insights/company-news/100-million-developers-and-counting/

[34] “Developer Survey.” Accessed: Aug. 16, 2023. [Online]. Available: https://survey.stackoverflow.co/2023/

[35] N. N. Zolkifli, A. Ngah, and A. Deraman, “Version Control System: A Review,” *Procedia Computer Science*, vol. 135, pp. 408–415, 2018, doi: 10.1016/j.procs.2018.08.191.

[36] N.Deepa, B.Prabadevi, K. L.B, and B.Deepa, “An analysis on Version Control Systems,” in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, Vellore, India: IEEE, Feb. 2020, pp. 1–9. doi: 10.1109/ic-ETITE47903.2020.39.

[37] S. Sterman, M. J. Nicholas, and E. Paulos, “Towards Creative Version Control,” *Proc. ACM Hum.-Comput. Interact.*, vol. 6, no. CSCW2, pp. 1–25, Nov. 2022, doi: 10.1145/3555756.