项目管理与经济学
Project Management & Economics

By, Fahad Sabah

TFSE_Fall2025_北京工业大学，中国

TFSE_Fall2025_Beijing University of Technology, China

CONTENTS

项目管理与经济学
Project Management & Economics

01
- ❑ COCOMO模型，增值管理（EVM）。
- ❑ COCOMO model, Earned Value Management (EVM).

02
- ❑利益相关者谈判中的博弈论。
- ❑Game theory in stakeholder negotiation.

软件项目管理（SPM）
Software Project Management (SPM)

Software Project Management (SPM) is a proper way of planning and leading software projects.

It is a part of project management in which software projects are planned, implemented, monitored, and controlled.

软件项目管理（SPM）是一种规划和领导软件项目的正确方式。

它是项目管理的一部分，软件项目被规划、实施、监控和控制。

SPM中的管理类型
Types of SPM

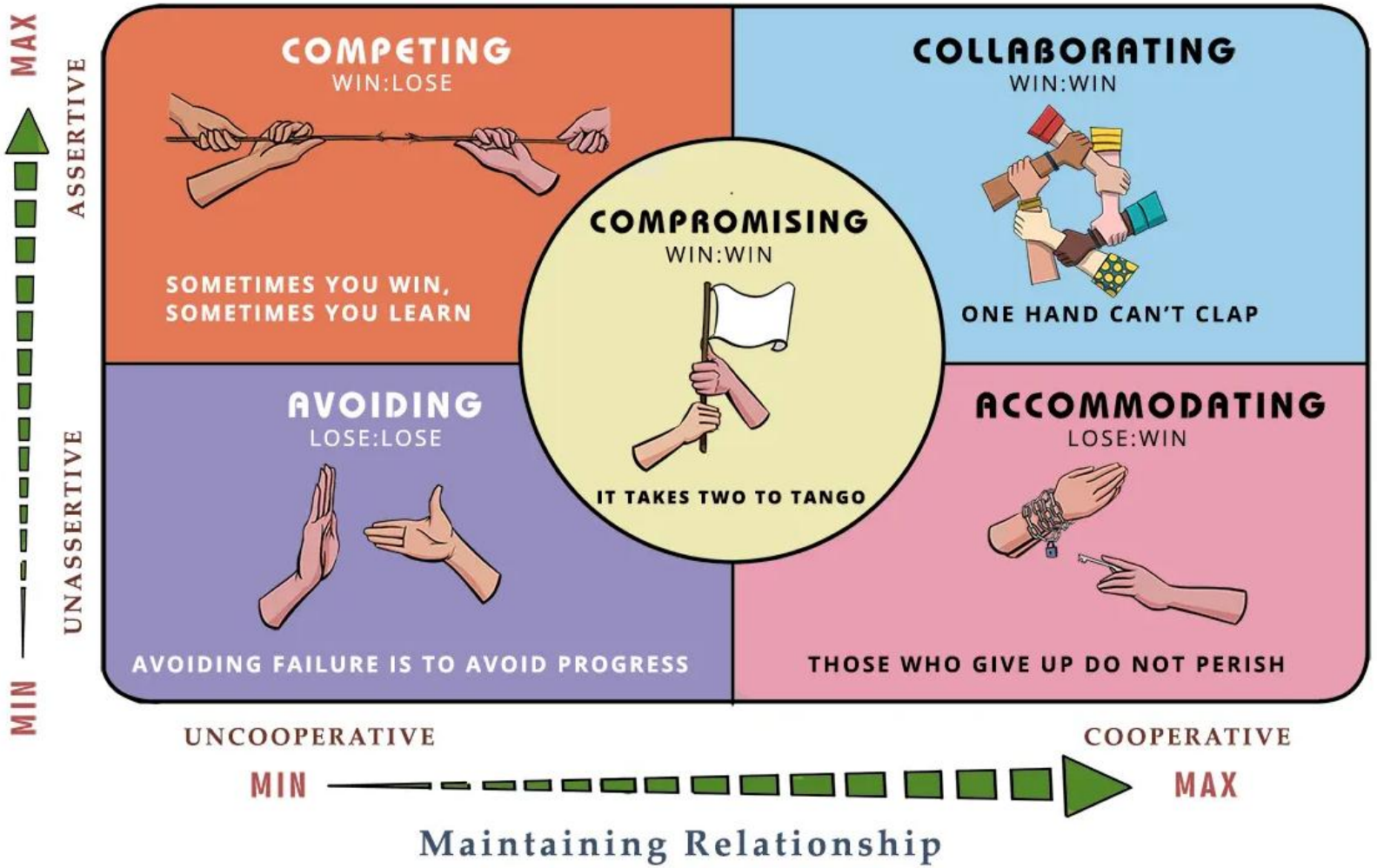
1. Conflict Management
2. Risk Management
3. Requirement Management
4. Change Management
5. Software Configuration Management
6. Release Management

1. 冲突管理
2. 风险管理
3. 需求管理
4. 变革管理
5. 软件配置管理
6. 发布管理

1. Conflict Management [冲突管理]

Conflic
proces
feature
increas
conflic

Proper
enhanc



突的负面
突积极特

提升团队

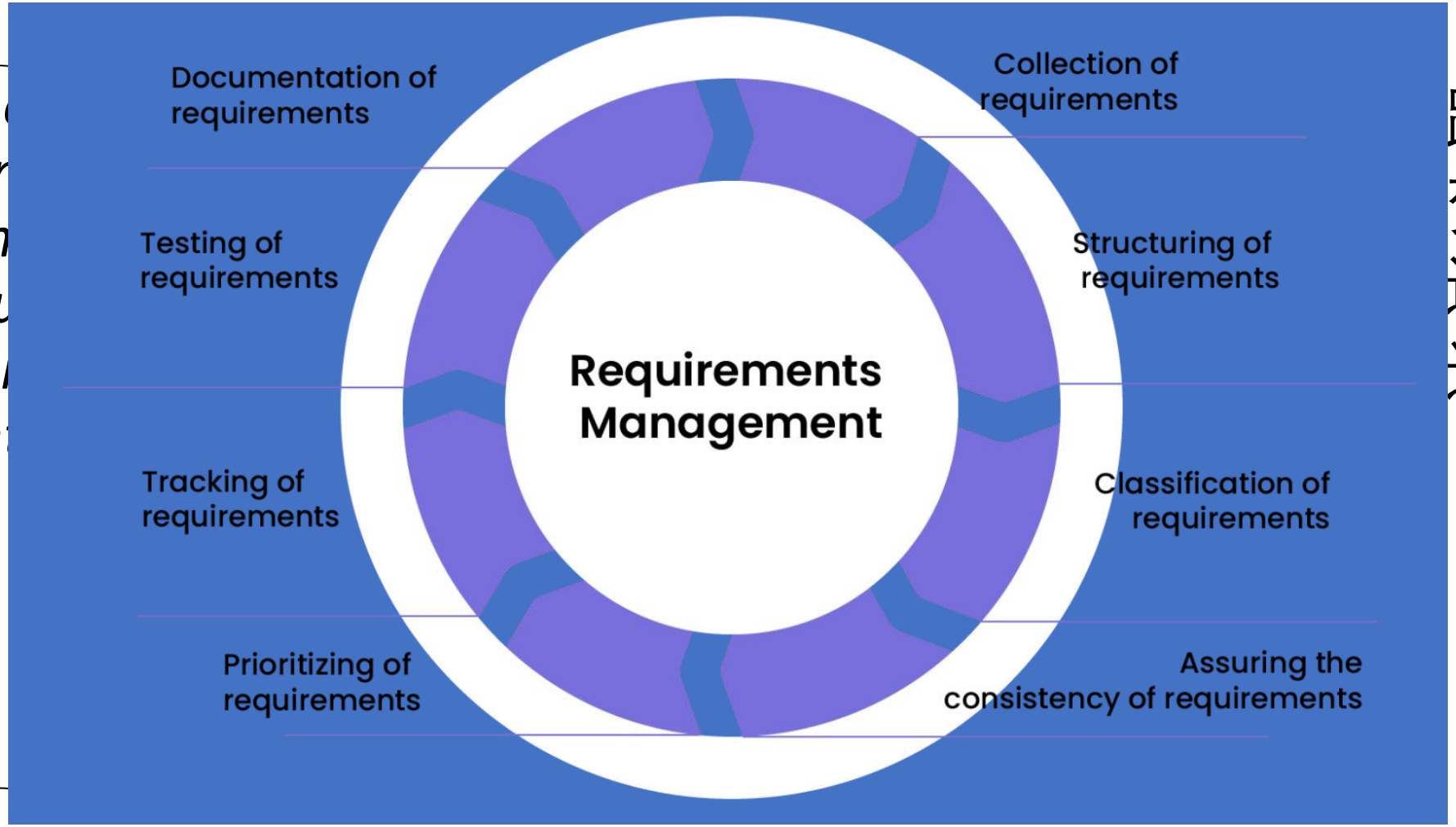
2. Risk Management [风险管理]

Risk management
and identification
followed by syn
economical imple
resources to minimi
control the possibi
unfortunate events
the realization of op

对风险进行分析
后同步且经济地
以最大限度减少、
不幸事件的可能
或最大化机会的

3. Requirement Management [需求管理]

It is th
prioritizir
documen
then su
communi
is a con
project.



跟踪变更过程。连续

4. Change Management [变革管理]

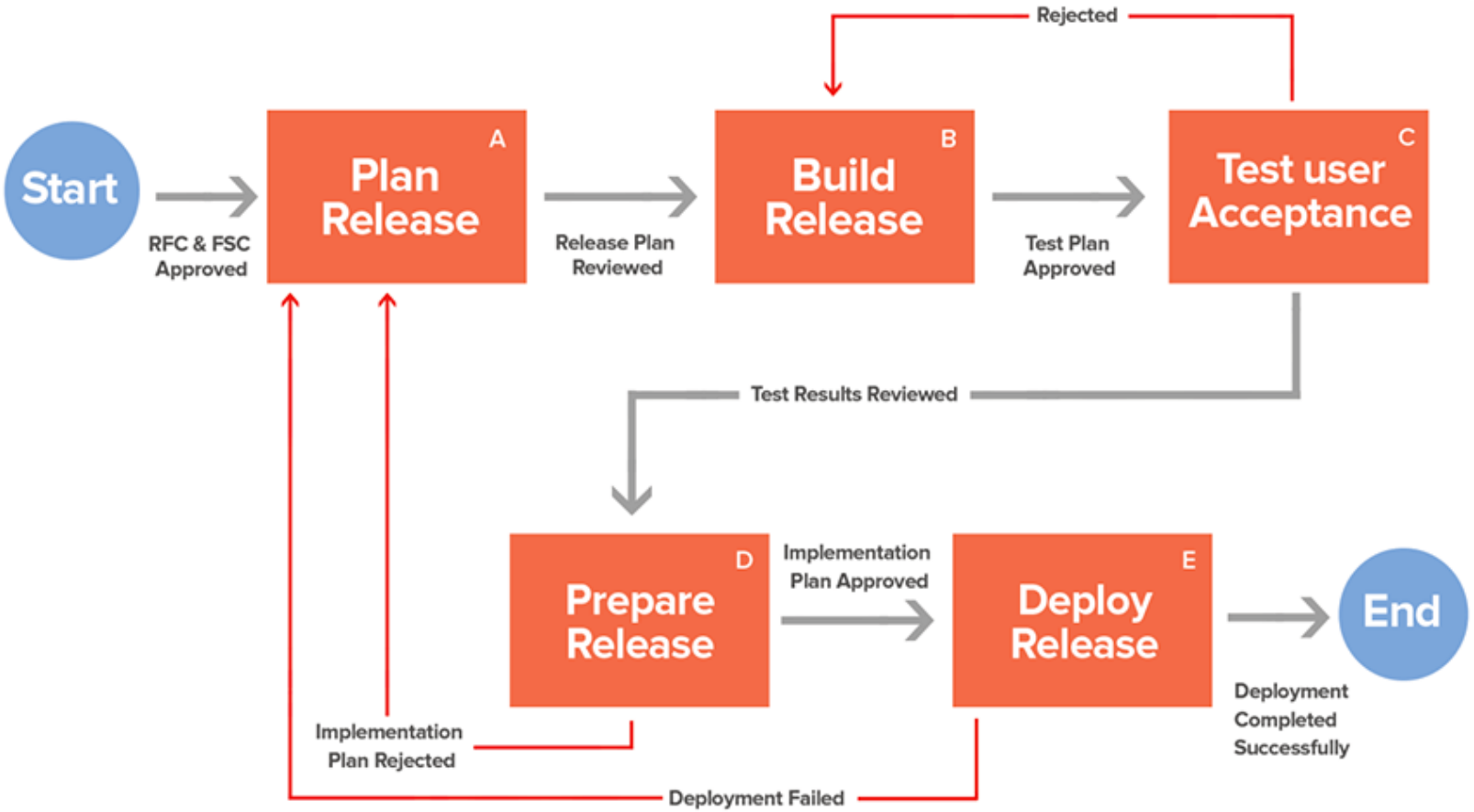
5. Software Configuration Management [软件配置管理]

The process of controlling and tracking changes in the software, part of the larger cross-disciplinary field of configuration management.

控制和跟踪软件变更的过程，是配置管理这一跨学科领域的一部分。

6. Release Management [发布管理]

The task and sch deploying managem organizati enhanced customer integrity c



部署
确保
整性
的新

软件项目管理的各个方面
Aspects of Software Project Management

1. Planning

2. Leading

3. Execution

4. Time Management

5. Budget

6. Maintenance

1. 规划
2. 领导
3. 处决
4. 时间管理
5. 预算
6. 维护

软件项目管理的各个方面
Aspects of Software Project Management

1. Planning

The Software Project Manager lays out the complete project's blueprint.

The project plan outlines the scope, resources, timelines, techniques, strategy, communication, testing, and maintenance steps.

1. 规划

软件项目经理会列出整个项目的蓝图。

项目计划概述了范围、资源、时间表、技术、策略、沟通、测试和维护步骤。

软件项目管理的各个方面
Aspects of Software Project Management

2. Leading

A software project manager brings together and leads a team of engineers, strategists, programmers, designers, and data scientists. Leading a team necessitates exceptional communication, interpersonal, and leadership abilities.

1. 规划

软件项目经理会列出整个项目的蓝图。

项目计划概述了范围、资源、时间表、技术、策略、沟通、测试和维护步骤。

软件项目管理的各个方面
Aspects of Software Project Management

3. Execution

SPM comes to the rescue here also as the person in charge of software projects will ensure that each stage of the project is completed successfully. measuring progress, monitoring to check how teams function, and generating status reports are all part of this process.

1. 规划

软件项目经理会列出整个项目的蓝图。

项目计划概述了范围、资源、时间表、技术、策略、沟通、测试和维护步骤。

软件项目管理的各个方面
Aspects of Software Project Management

4. Time Management

Abiding by a timeline is crucial to completing deliverables successfully. This is especially difficult when managing software projects because changes to the original project charter are unavoidable over time. To assure progress in the face of blockages or changes, software project managers ought to be specialists in managing risk and emergency preparedness.

1. 规划

软件项目经理会列出整个项目的蓝图。

项目计划概述了范围、资源、时间表、技术、策略、沟通、测试和维护步骤。

软件项目管理的各个方面
Aspects of Software Project Management

5. Budget

Software Project Managers, like conventional project managers, are responsible for generating a project budget and adhering to it as closely as feasible, regulating spending, and reassigning funds as needed.

1. 规划

软件项目经理会列出整个项目的蓝图。

项目计划概述了范围、资源、时间表、技术、策略、沟通、测试和维护步骤。

软件项目管理的各个方面
Aspects of Software Project Management

6. Maintenance

Software project management emphasizes continuous product testing to find and repair defects early, tailor the end product to the needs of the client, and keep the project on track.

1. 规划

软件项目经理会列出整个项目的蓝图。

项目计划概述了范围、资源、时间表、技术、策略、沟通、测试和维护步骤。

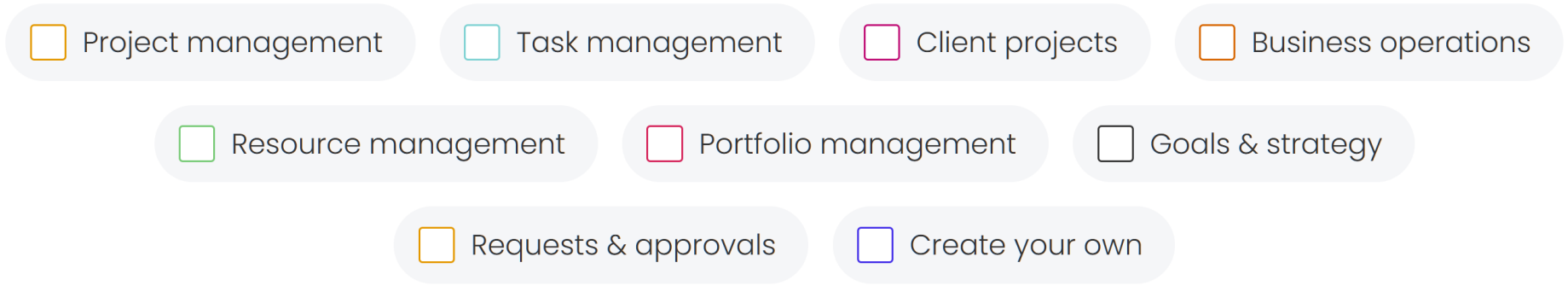
1. monday.com: Best for individual use
2. ClickUp: Best for complex workflows
3. Smartsheet: Best spreadsheet-style project management
4. Jira: Best for agile software teams
5. Wrike: Best for complex timelines
6. Notion: Best for combined notetaking and task tracking
7. Teamwork.com: Best for client-facing work
8. Miro: Best for collaborative ideation
9. Trello: Best for visual project planning
10. Asana: Best for basic task tracking

monday.com

Project management made easy

Manage simple to complex projects and everything in between with monday.com

Select what you want to manage:



Get Started →

monday work management

Who else is on your team?

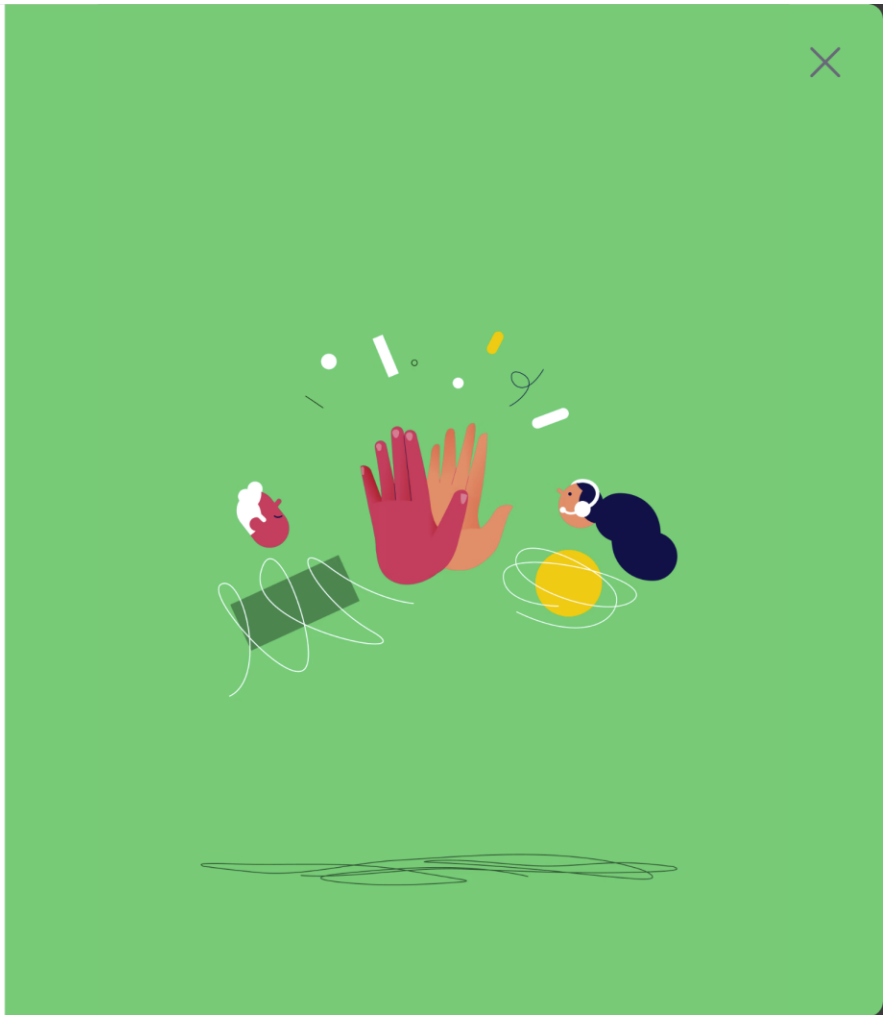
Add email here | Admin

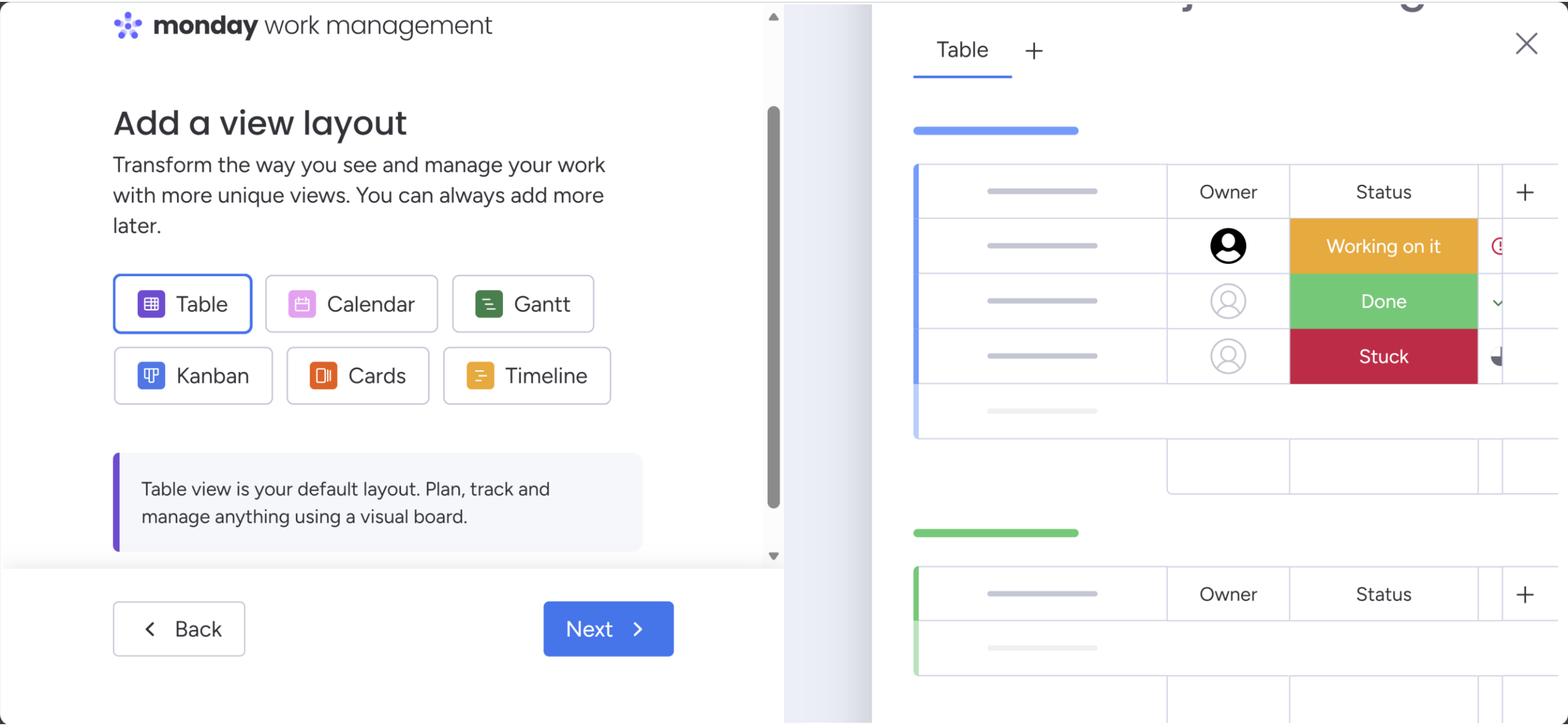
Add email here | Admin

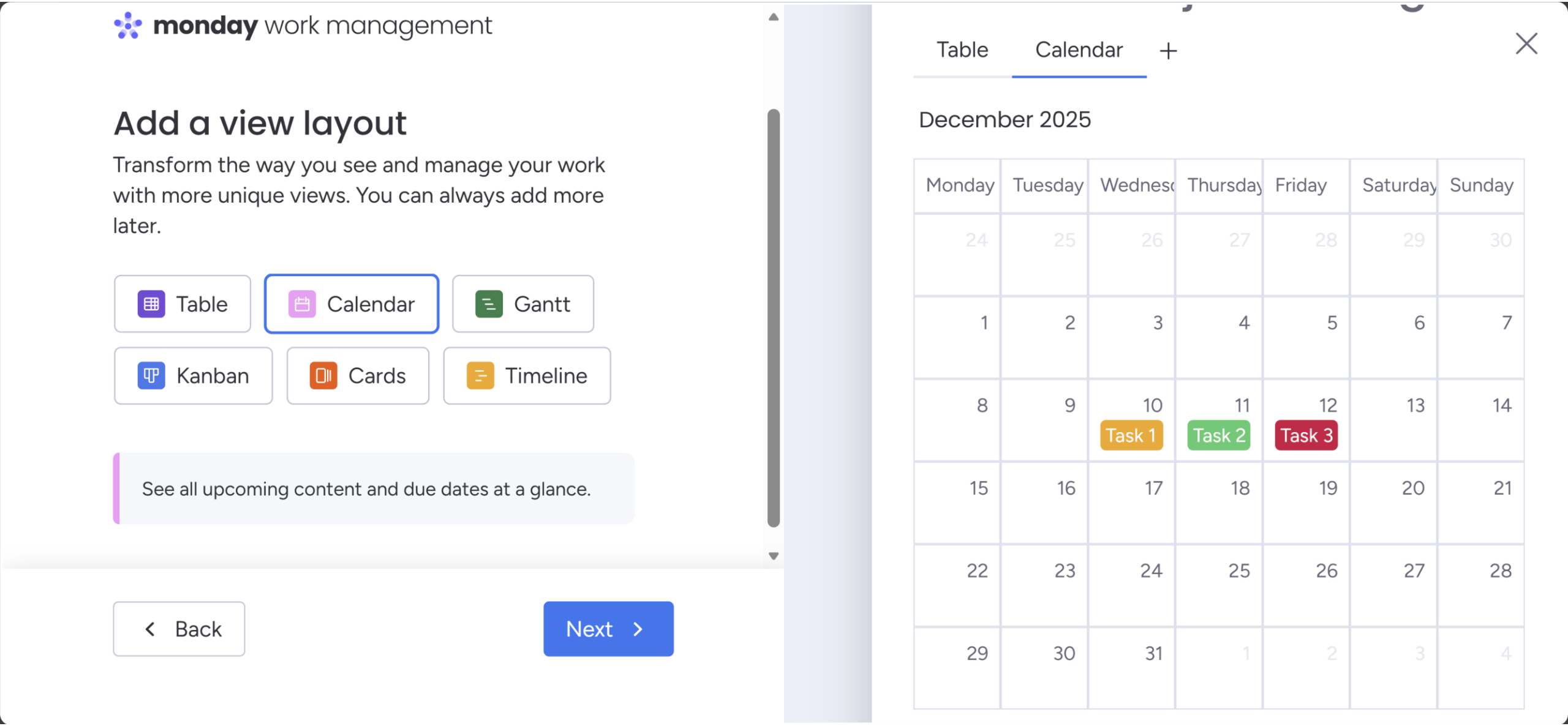
+ Add another

Remind me later

Invite your team




monday work management
Add a view layout
Transform the way you see and manage your work with more unique views. You can always add more later.
Table
Calendar
Gantt
Kanban
Cards
Timeline
Table view is your default layout. Plan, track and manage anything using a visual board.
Back
Next
Table
Owner
Status
Working on it
Done
Stuck
Owner
Status


monday work management
Add a view layout
Transform the way you see and manage your work with more unique views. You can always add more later.
Table
Calendar
Gantt
Kanban
Cards
Timeline
See all upcoming content and due dates at a glance.
Back
Next
Table
Calendar
December 2025
Monday
Tuesday
Wednesd
Thursday
Friday
Saturday
Sunday
24
25
26
27
28
29
30
1
2
3
4
5
6
7
8
9
10
Task 1
11
Task 2
12
Task 3
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
1
2
3
4

monday work management

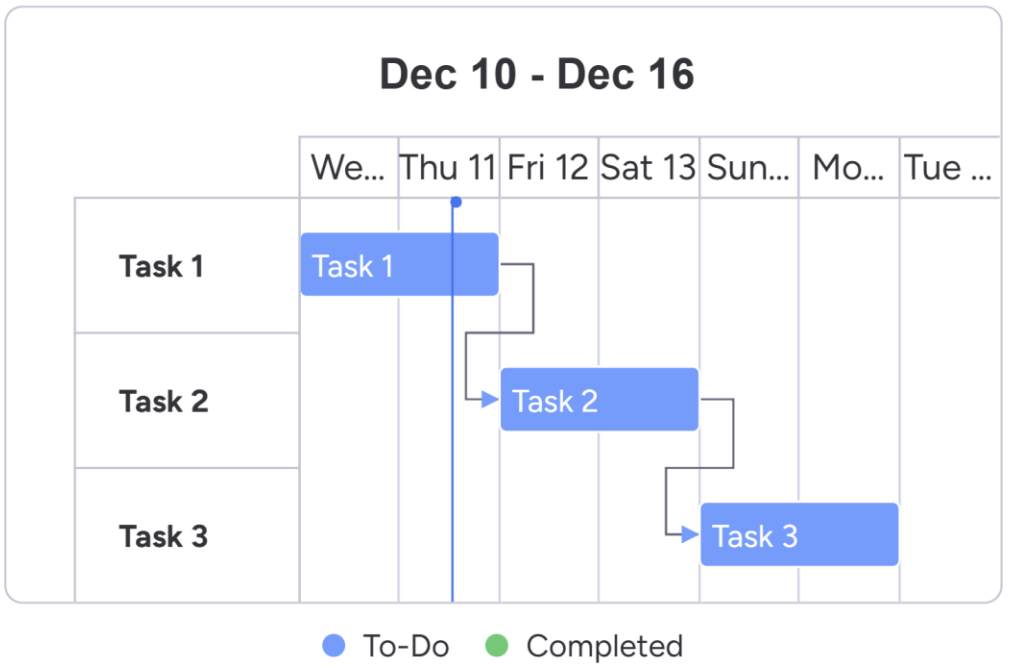
Add a view layout

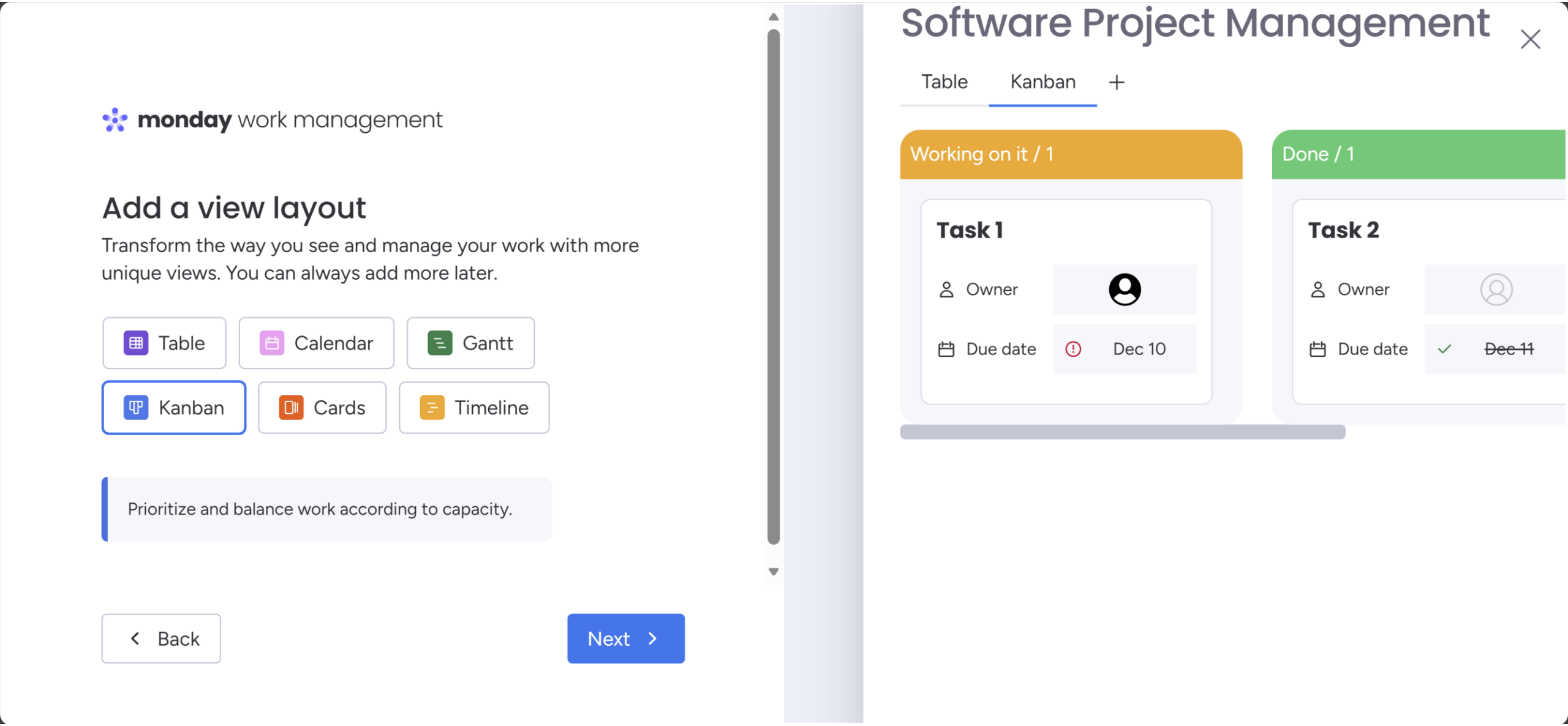
Transform the way you see and manage your work with more unique views. You can always add more later.

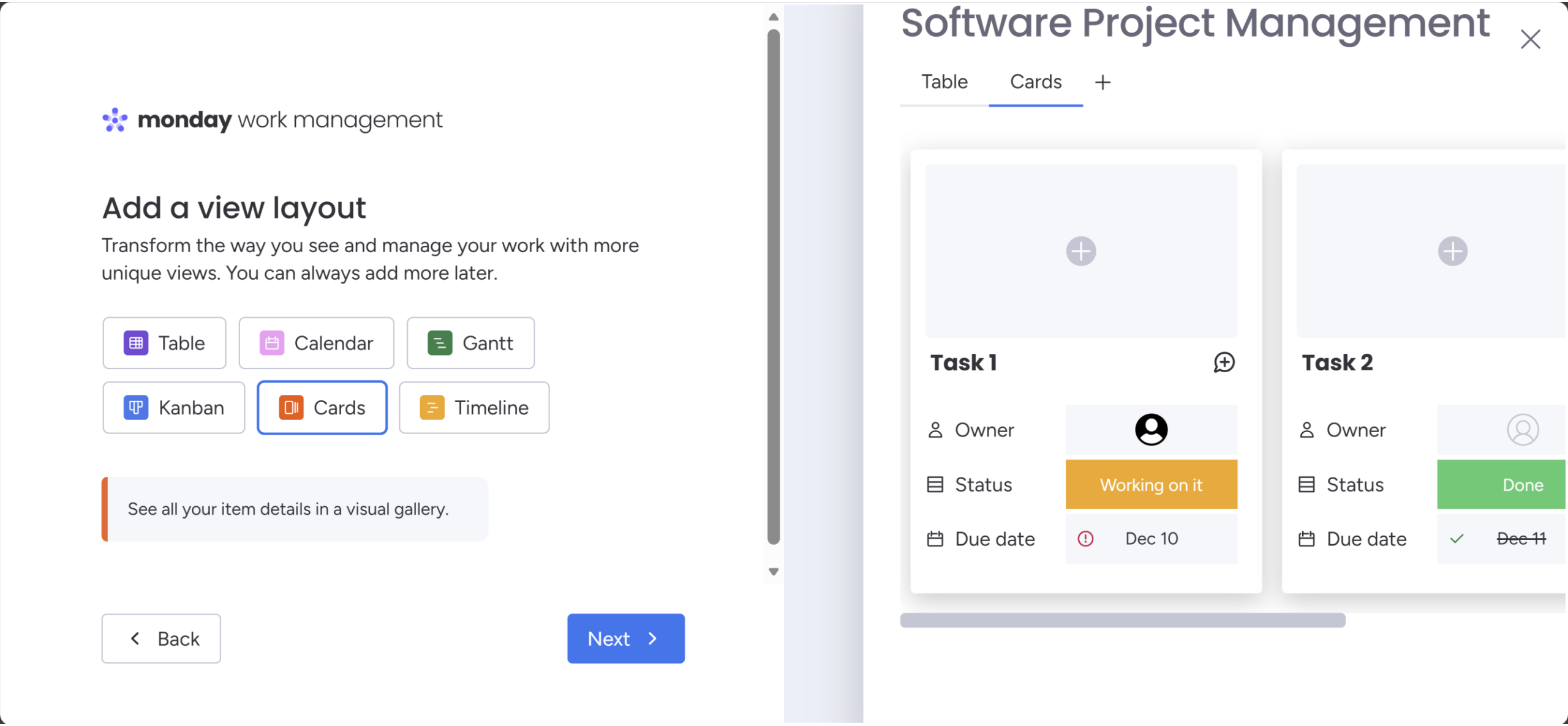
Table | Calendar | Gantt | Kanban | Cards | Timeline

Visualize project milestones and dependencies.

Back | Next




monday work management
Add a view layout
Transform the way you see and manage your work with more unique views. You can always add more later.
Table
Calendar
Gantt
Kanban
Cards
Timeline
Prioritize and balance work according to capacity.
Back
Next
Software Project Management
Table
Kanban
Working on it / 1
Task 1
Owner
Due date
Dec 10
Done / 1
Task 2
Owner
Due date
Dec 11


monday work management
Add a view layout
Transform the way you see and manage your work with more unique views. You can always add more later.
Table
Calendar
Gantt
Kanban
Cards
Timeline
See all your item details in a visual gallery.
Back
Next
Software Project Management
Table
Cards
Task 1
Owner
Status
Working on it
Due date
Dec 10
Task 2
Owner
Status
Done
Due date
Dec 11

monday work management

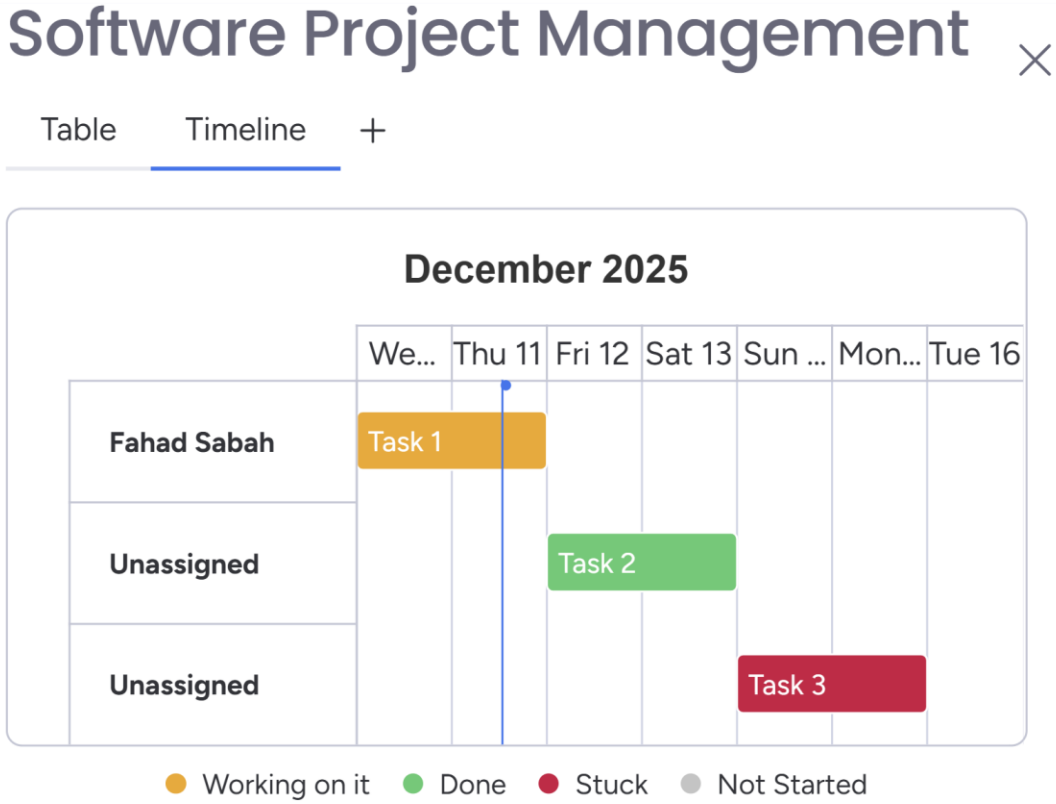
Add a view layout

Transform the way you see and manage your work with more unique views. You can always add more later.

Table | Calendar | Gantt | Kanban | Cards | Timeline

Stay on track with visual deadlines and timelines.

Back | Next



Standout features [突出特色]

- **Monday AI**: Utilizes artificial intelligence to automate routine tasks and provide data-driven insights
- **Embedded documents**: Allows integration and editing of documents directly within your monday boards
- **Custom notifications**: Enables personalized alerts to keep team members informed about relevant updates and priority tasks

Pros

•Highly customizable project boards
•Intuitive automation tool
•Over 200 project templates

Cons

•Time tracking is available only in higher tiers
•Per group seat pricing model can be confusing
•Slight learning curve

COCOMO模型 - 软件工程
COCOMO Model - Software Engineering

*The **Constructive Cost Model (COCOMO)** It was proposed by Barry Boehm in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models.*

*It is a Software **Cost Estimation Model** that helps predict the effort, cost, and schedule required for a software development project.*

建构成本模型（COCOMO）由巴里·博姆于1981年提出，基于对63个项目的研究，是文献资料最详尽的模型之一。

它是一种软件成本估算模型，帮助预测软件开发项目所需的工作量、成本和进度。

COCOMO模型中的项目类型
Types of Projects in COCOMO Model

In the COCOMO model, software projects are categorized into three types based on their complexity, size, and the development environment. These types are:

1. Organic
2. Semi-detached
3. Embedded

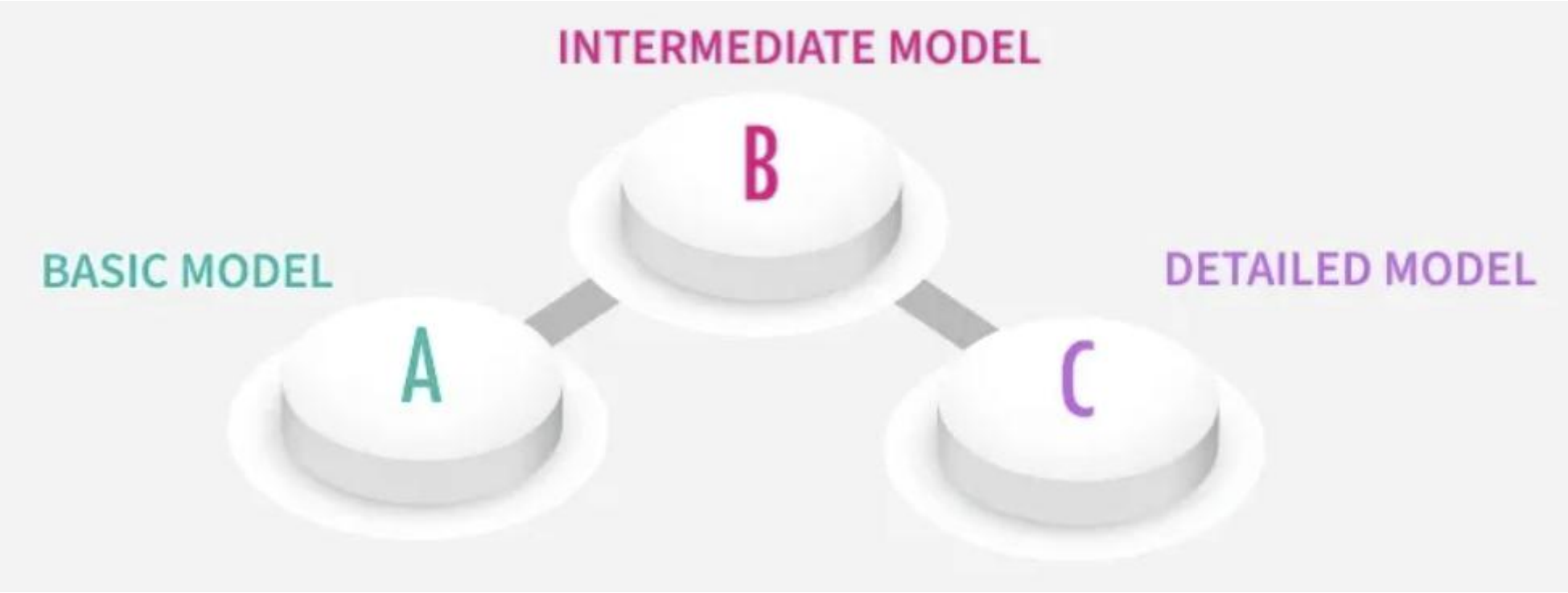
在COCOMO模型中，软件项目根据其复杂度、规模和开发环境被分为三类。这些类型包括：

1. 有机
2. 半独立式
3. 嵌入式

COCOMO模型中项目类型的比较
Comparison of Types of Projects in COCOMO Model

Aspects	Organic	Semidetached	Embedded
Project Size	2 to 50 KLOC	50-300 KLOC	300 and above KLOC
Complexity	Low	Medium	High
Team Experience	Highly experienced	Some experienced as well as inexperienced staff	Mixed experience, includes experts
Environment	Flexible, fewer constraints	Somewhat flexible, moderate constraints	Highly rigorous, strict requirements
Effort Equation	$E = 2.4(400)^{1.05}$	$E = 3.0(400)^{1.12}$	$E = 3.6(400)^{1.20}$
Example	Simple payroll system	New system interfacing with existing systems	Flight control software

COCOMO模型类型
Types of COCOMO Model



Basic COCOMO Model [基本COCOMO模型]

The Basic COCOMO model is a straightforward way to estimate the effort needed for a software development project. It uses a simple mathematical formula to predict how many person-months of work are required based on the size of the project, measured in thousands of lines of code (KLOC).

$$Effort(E) = a(kLOC)^b$$

$$Time(T) = c(E)^d$$

$$People\ required = \frac{E}{T}$$

Where,

- E is effort applied in Person-Months
- kLOC is the estimated size of the software product indicate in Kilo Lines of Code
- T is the development time in months
- a, b, c, and d are constants determined by the category of software.

Basic COCOMO Model [基本COCOMO模型]

The constant values a, b, c, and d for the Basic Model for the different categories of the software projects are:

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

1. The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. The development time is measured in months.
2. These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, and expertise is taken into account, henceforth the estimate is rough.

Intermediate COCOMO Model [中间COCOMO模型]

In reality, no system's effort and schedule can be solely calculated based on Lines of Code. For that, various other factors such as reliability, experience, and Capability. These factors are known as Cost Drivers (multipliers).

$$Effort(E) = a(kLOC)^b * EAF$$

$$Time(T) = c(E)^d$$

Where,

- E is effort applied in Person-Months
- kLOC is the estimated size of the software product indicate in Kilo Lines of Code
- **EAF is the Effort Adjustment Factor (EAF) is a multiplier used to refine the effort estimate obtained from the basic COCOMO model.**
- T is the development time in months
- a, b, c, and d are constants determined by the category of software.

Intermediate COCOMO Model [中间COCOMO模型]

The constant values a, b, c, and d for the Intermediate Model for the different categories of the software projects are:

Software Projects	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

COCOMO模型示例：图书馆管理系统
COCOMO Model Example: Library Management System

Project Overview

- Project Name: Digital Library Management System
- Type: Web-based application with mobile interface
- Platform: Web (React frontend, Node.js backend, PostgreSQL database)
- Team Size: 8 members
- Timeline: 12 months planned

PROJECT SPECIFICATIONS [项目规范]

Functional Requirements:

- ❑User Management (Members, Librarians, Administrators)
- ❑Book Catalog Management (100,000+ titles)
- ❑Loan Management (Issuing, Returns, Renewals)
- ❑Reservation System
- ❑Fine Calculation and Payment Processing
- ❑Reporting and Analytics Dashboard
- ❑Inventory Management
- ❑Integration with External Databases (ISBN lookup)
- ❑Mobile Application (iOS/Android)
- ❑Notification System (Email/SMS)

Technical Characteristics[技术特征]:

- ❑Complexity: Medium-High (integration with external systems)
- ❑Database: Relational with complex queries
- ❑Interfaces: Web, Mobile, API for third-party
- ❑Security Requirements: High (payment processing, personal data)
- ❑Reusability: Medium (60% components reusable)

SIZE ESTIMATION [尺寸估计]:

Module	Function Points	Language (JavaScript/TypeScript)	SLOC/FP	Estimated SLOC
User Management	45	JavaScript	53	2,385
Book Management	65	JavaScript	53	3,445
Loan Management	55	JavaScript	53	2,915
Reservation System	35	JavaScript	53	1,855
Payment Processing	40	JavaScript	53	2,120
Reporting Module	50	JavaScript	53	2,650
Mobile App	70	React Native	47	3,290
Admin Dashboard	45	JavaScript	53	2,385
API Services	60	TypeScript	50	3,000
Total	465	**Total Estimated SLOC = 24k (rounded)**		

COCOMO MODEL SELECTION[COCOMO模型选择]

- We'll use Intermediate COCOMO for better accuracy:
  - Basic Formula:
  - Effort = a × (KSLOC)^b × EAF
  - Development Time = c × (Effort)^d
  - Where for Semi-detached mode (mixed experience team, medium innovation):
    - a = 3.0
    - b = 1.12
    - c = 2.5
    - d = 0.35

努力调整因子（EAF）
EFFORT ADJUSTMENT FACTORS (EAF)

Cost Driver	Rating	Value	Explanation
Product Factors			
Required Software Reliability	High	1.15	Library transactions critical
Database Size	High	1.08	100,000+ book records
Product Complexity	High	1.17	Complex business logic
Platform Factors			
Execution Time Constraints	Nominal	1.00	No real-time constraints
Main Storage Constraint	Nominal	1.00	Standard requirements
Platform Volatility	Low	0.87	Stable technology stack

努力调整因子（EAF）
EFFORT ADJUSTMENT FACTORS (EAF)

Cost Driver	Rating	Value	Explanation
Personnel Factors			
Analyst Capability	High	0.86	Experienced analysts
Programmer Capability	High	0.86	Skilled developers
Applications Experience	Nominal	1.00	Some library system experience
Platform Experience	High	0.91	Familiar with chosen stack
Language & Tool Experience	High	0.91	Expert in JS/Node.js

努力调整因子（EAF）
EFFORT ADJUSTMENT FACTORS (EAF)

Cost Driver	Rating	Value	Explanation
Project Factors			
Use of Software Tools	High	0.91	Modern CI/CD tools
Multisite Development	Nominal	1.00	Single location
Required Development Schedule	Nominal	1.00	Standard 12-month schedule

EAF Calculation:

EAF = 1.15 × 1.08 × 1.17 × 0.87 × 0.86 × 0.86 × 0.91 × 0.91 × 0.91

EAF = 0.92 (slightly favorable conditions)

EFFORT CALCULATION [努力计算]

- Nominal Effort (without adjustment):

  Effort_nominal = 3.0 × (24)^1.12
  = 3.0 × (24^1.12)
  = 3.0 × 34.45
  = 103.35 person-months

- Adjusted Effort:

  Effort_adjusted = 103.35 × 0.92
  = 95.08 person-months

SCHEDULE CALCULATION [排程计算]

- Development Time:

  Tdev = 2.5 × (95.08)^0.35
  = 2.5 × (95.08^0.35)
  = 2.5 × 4.26
  = 10.65 months

- Average Team Size:

  Team Size = Effort / Tdev
  = 95.08 / 10.65
  = 8.93 ≈ 9 persons

按相位分布（基于COCOMO）
PHASE-WISE DISTRIBUTION (Based on COCOMO)

Phase	Effort %	Person-Months	Duration (Months)
Requirements & Planning	6%	5.70	1.2
System Design	16%	15.21	2.1
Detailed Design	24%	22.82	2.5
Coding & Unit Testing	38%	36.13	3.5
Integration & Testing	16%	15.21	1.35
Total	100%	95.08	10.65

RECOMMENDATIONS [建议]

Staffing Plan:

- ❑Start with 6 team members, ramp up to 10 during coding phase
- ❑Include 2 senior developers for complex modules

Project Management:

- ❑Use Agile methodology with 2-week sprints
- ❑Prioritize core modules (Book Management, Loan System)
- ❑Plan for 20% requirements change buffer

Cost-Saving Opportunities:

- ❑Use open-source libraries where possible
- ❑Consider cloud-based solutions to reduce infrastructure costs
- ❑Implement reusable components across web and mobile

Quality Assurance:

- ❑Allocate 25% of effort for testing (higher than typical 16% due to payment processing)
- ❑Implement automated testing from beginning

增值管理（EVM）
Earned Value Management (EVM)

增值管理（EVM）
Earned Value Management (EVM)

The Challenge of Traditional Project Reporting

- Traditional project status reporting suffers from two major flaws:
  - Separate reporting: Schedule and cost are reported independently
  - Focus on inputs: Emphasis on money spent, not value delivered
- Example of the problem:
  - "We've spent 80% of our budget" - Sounds good?
  - "But we've only completed 50% of the work" - Uh oh!
  - Traditional reporting might miss this connection

The “EVM” Revolution [EVM 革命]

EVM was developed by the U.S. Department of Defense in the 1960s and has become a global standard (ANSI/EIA-748). It provides an integrated view of project health by answering three fundamental questions:

- ❑Where are we supposed to be? (Planned Value)
- ❑How much have we actually accomplished? (Earned Value)
- ❑How much did it cost to get there? (Actual Cost)

The Three Pillars of EVM [EVM 的三大支柱]

Pillar 1: Planned Value (PV) - The Plan

- ❑Definition: The authorized budget assigned to scheduled work
- ❑Also called: Budgeted Cost of Work Scheduled (BCWS)
- ❑At completion: PV equals Budget at Completion (BAC)

The Three Pillars of EVM [EVM 的三大支柱]

Pillar 2: Earned Value (EV) - The Reality

- Definition: The value of work actually completed, measured against the plan
- Also called: Budgeted Cost of Work Performed (BCWP)
- Calculation methods: Percent complete × Task budget

The Three Pillars of EVM [EVM的三大支柱]

Pillar 3: Actual Cost (AC) - The Spending

- ❑Definition: The actual costs incurred for the work performed
- ❑Also called: Actual Cost of Work Performed (ACWP)
- ❑Important: Includes all costs - labor, materials, equipment, etc.

整合的力量：绩效指数
The Power of Integration: Performance Indices

- Schedule Performance Index (SPI)
  - SPI = EV / PV
- Interpretation:

  SPI = 1.0: Exactly on schedule

  SPI > 1.0: Ahead of schedule (good!)

  SPI < 1.0: Behind schedule (bad!)

  Example: SPI = 0.85 means you've earned only 85 cents of value for every dollar of planned work

整合的力量：绩效指数
The Power of Integration: Performance Indices

- Cost Performance Index (CPI)
  - CPI = EV / AC
- Interpretation:

  CPI = 1.0: Exactly on budget

  CPI > 1.0: Under budget (good!)

  CPI < 1.0: Over budget (bad!)

  Example: CPI = 0.90 means you've earned 90 cents of value for every dollar spent

Forecasting: Seeing the Future [预测：预见未来]

- Estimate at Completion (EAC)
- Most common formula (assuming current trends continue):
  - EAC = BAC / CPI
- Other formulas for different scenarios:
  - EAC = AC + (BAC - EV): When remaining work will be done at planned rate
  - EAC = AC + [(BAC - EV) / (CPI × SPI)]: When both cost and schedule issues affect remaining work

Forecasting: Seeing the Future [预测：预见未来]

- Variance at Completion (VAC)
  - VAC = BAC - EAC
  - Positive VAC: Project will finish under budget
  - Negative VAC: Project will finish over budget

- To Complete Performance Index (TCPI)
  - TCPI = (BAC - EV) / (BAC - AC)
  - The efficiency needed on remaining work to meet the original budget
  - TCPI > 1.0: Need to work more efficiently than planned
  - TCPI < 1.0: Can work less efficiently than planned

实用应用 - 图书馆管理系统
Practical Application - Library Management System

- Project Overview
  - Project: Library Management System (LMS)
  - Budget at Completion (BAC): $100,000
  - Duration: 5 months
  - Reporting Period: End of Month 3

Project Breakdown Structure [项目分解结构]

Work Package	Budget	Duration	% Complete at Month 3
1. Requirements Analysis	$15,000	Month 1	100%
2. System Design	$20,000	Month 2	100%
3. Database Development	$25,000	Months 2-3	80%
4. User Interface	$20,000	Months 3-4	40%
5. Testing	$15,000	Months 4-5	0%
6. Deployment	$5,000	Month 5	0%
TOTAL	$100,000	5 months	

计划价值（PV）计算
Planned Value (PV) Calculation

- Based on the project schedule, what should we have spent by Month 3?

Work Package	Monthly Planned Spend	Cumulative PV by Month 3
1. Requirements	$15,000 (Month 1)	$15,000
2. System Design	$20,000 (Month 2)	$35,000 ($15k + $20k)
3. Database Dev	$12,500/month (Months 2-3)	$47,500 ($35k + $12.5k)
4. User Interface	$10,000 (Month 3)	$57,500
5. Testing	$0 (starts Month 4)	$57,500
6. Deployment	$0 (starts Month 5)	$57,500

Total PV at Month 3 = $57,500

劳动价值（EV）计算
Earned Value (EV) Calculation

- What value have we actually earned based on work completed?

Work Package	Budget	% Complete	Earned Value
1. Requirements	$15,000	100%	$15,000
2. System Design	$20,000	100%	$20,000
3. Database Dev	$25,000	80%	$20,000
4. User Interface	$20,000	40%	$8,000
5. Testing	$15,000	0%	$0
6. Deployment	$5,000	0%	$0
TOTAL	$100,000		$63,000

Total EV at Month 3 = $63,000

Actual Cost (AC) Data [当前成本（AC）日期]

- What have we actually spent?

Work Package	Actual Cost Incurred
1. Requirements	$16,000 (overspent due to scope changes)
2. System Design	$22,000 (team overtime)
3. Database Dev	$18,000 (efficient work)
4. User Interface	$12,000 (ahead but expensive)
5. Testing	$1,000 (early prep work)
6. Deployment	$0
TOTAL	$69,000

LMS项目的EVM分析
EVM Analysis for LMS Project

Performance Metrics Calculation

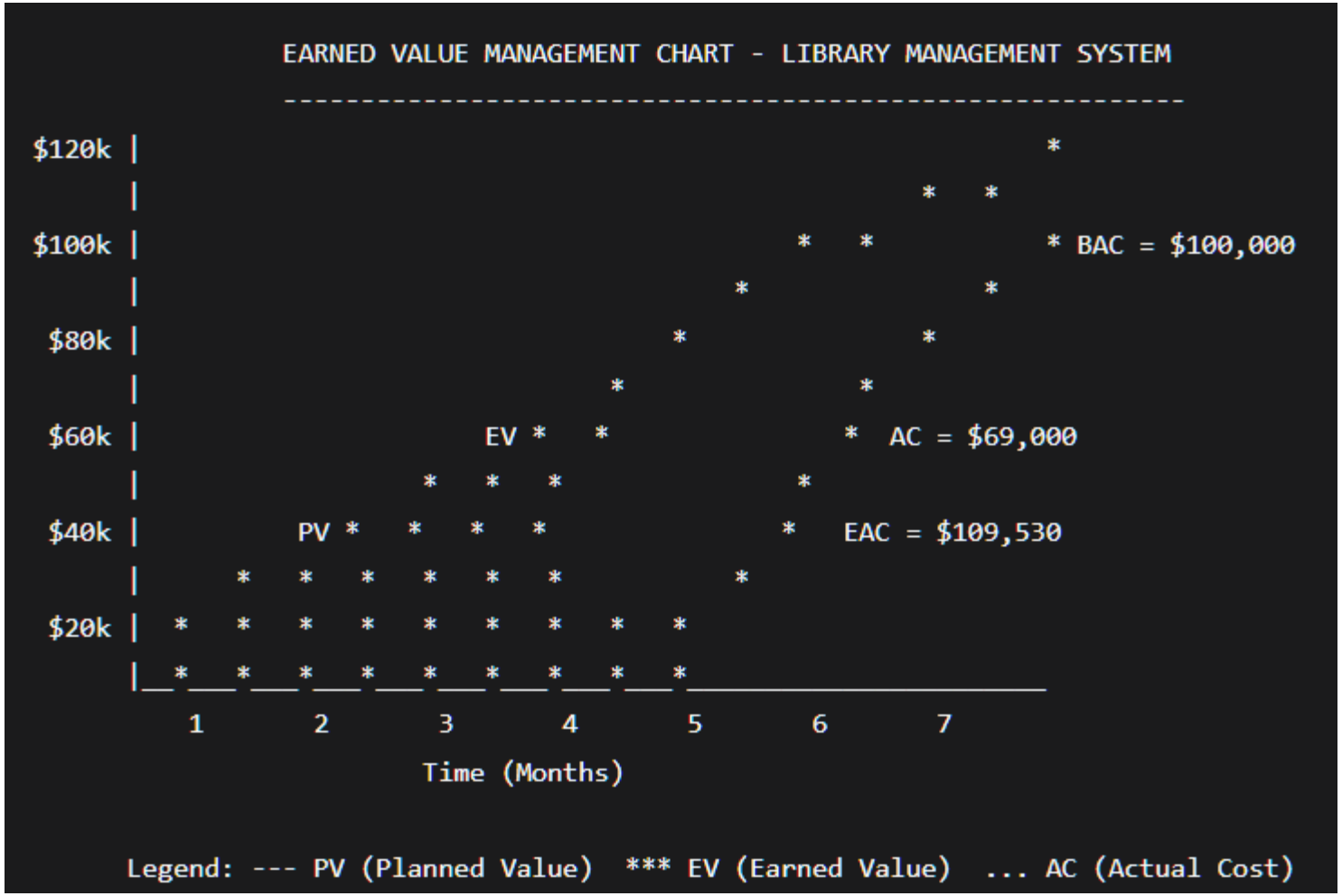
Metric	Formula	Calculation	Result	Interpretation
Schedule Variance (SV)	EV - PV	$63,000 - $57,500	+$5,500	Ahead of schedule
Cost Variance (CV)	EV - AC	$63,000 - $69,000	-$6,000	Over budget
Schedule Performance Index (SPI)	EV / PV	$63,000 / $57,500	1.096	9.6% ahead of schedule
Cost Performance Index (CPI)	EV / AC	$63,000 / $69,000	0.913	8.7% over budget

LMS项目的EVM分析
EVM Analysis for LMS Project

Forecast Calculations

Forecast	Formula	Calculation	Result	Interpretation
Estimate at Completion (EAC)	BAC / CPI	$100,000 / 0.913	$109,530	Will cost ~$9,530 more than planned
Variance at Completion (VAC)	BAC - EAC	$100,000 - $109,530	-$9,530	Will finish $9,530 over budget
Estimate to Complete (ETC)	EAC - AC	$109,530 - $69,000	$40,530	Need this much to finish
To Complete CPI (TCPI)	(BAC - EV) / (BAC - AC)	($100k - $63k) / ($100k - $69k)	1.194	Need 19.4% better efficiency on remaining work

Visual “EVM” Analysis [可视化“EVM”分析]



Managerial Insights from EVM [EVM的管理洞察]

- **Early Warning System:** EVM caught the cost problem in Month 3, giving us 2 months to correct course
- **Trade-off Analysis:** We're trading cost for schedule. Is this acceptable? Need stakeholder alignment.
- **Objective Decision Making:** Hard numbers replace opinions:
  - "We need more money" → "We need $9,530 to complete based on current CPI of 0.913"
  - "We're doing well" → "We're 9.6% ahead on schedule but 8.7% over on cost"
- **Proactive Management:** Can now make informed decisions:
  - Option A: Request additional funds ($9,530)
  - Option B: Reduce scope by ~9.5% to stay within budget
  - Option C: Improve efficiency by 19.4% on remaining work

利益相关者谈判中的博弈论
Game Theory in Stakeholder Negotiation

为什么选择博弈论在项目管理中？
Why Game Theory in Project Management?

The Stakeholder Dilemma

- Every project is a complex ecosystem of competing interests:
  - Clients want maximum features for minimum cost
  - Contractors want maximum profit with minimum effort
  - Team Members want work-life balance and career growth
  - Regulators want compliance and safety
  - End Users want usability and reliability

Game Theory: Systematic analysis of incentives, predictions of behavior, strategic design of interactions

项目经理作为博弈论家
The Project Manager as Game Theorist

- As a PM, you must:
  - Model the negotiation as a game
  - Predict stakeholder moves
  - Design rules and incentives for better outcomes
  - Intervene strategically to change the game

项目经理核心博弈论概念
Core Game Theory Concepts for Project Managers

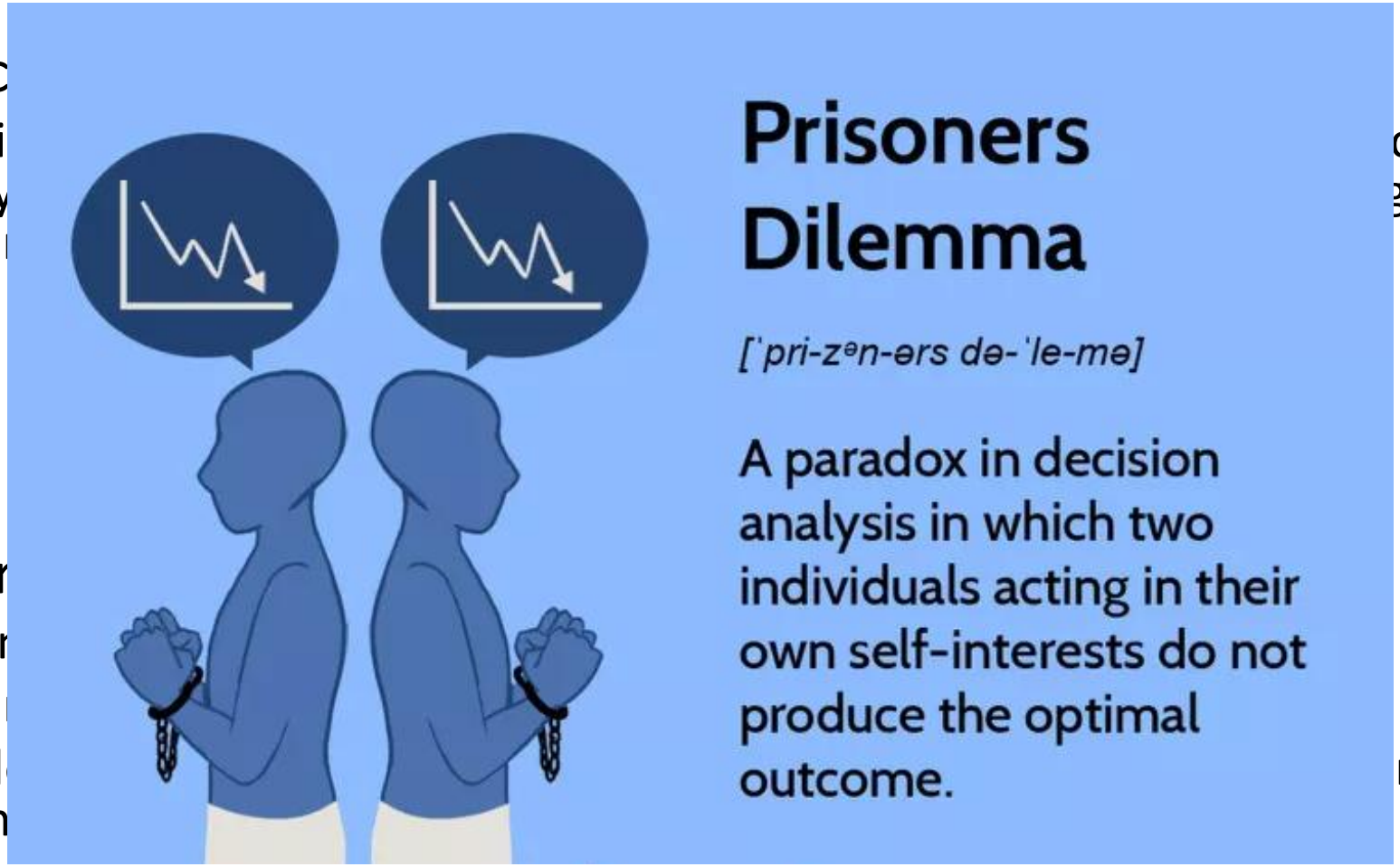
- **Zero-Sum Games (Win-Lose)**
  - Definition: One player's gain equals another's loss
  - Project Example: Fixed-price contract negotiation over scope
  - Total "value" is fixed: Client saves $10,000 = Contractor loses $10,000
  - Strategy: Maximize your share through tough negotiation

- **Non-Zero-Sum Games (Win-Win or Lose-Lose)**
  - Definition: Outcomes where players can both gain or both lose
  - Project Example: Early delivery with bonus for both parties
  - Total value can grow: Efficient work creates surplus to share
  - Strategy: Create value first, then claim your share

- **Simultaneous vs. Sequential Games**
  - Simultaneous: Players act without knowing others' choices (e.g., sealed bids)
  - Sequential: Players take turns (e.g., negotiation rounds)

囚徒困境——信任问题
The Prisoner's Dilemma - The Trust Problem

- Classic Sc
  - Two cri
    confess, each
    gets 3 y
    goes free and
    the sile
- Project Tr
  - Share ir
  - Invest i
  - The Dil
    rse collective
    outcom



Real life Examples [现实生活中的例子]

Example 1: The Pizza Dilemma (The Zero-Sum Mindset)

Scenario: You and your sibling are sharing a pizza. There are 8 slices.

- Zero-Sum Thinking: "If I get 5 slices, they only get 3. I win, they lose."
- Problem: This creates a fight. You might end up with uneven slices, someone crying, and parents getting involved.
- Game Theory Lesson: Sometimes treating things as purely competitive ("I win, you lose") makes everyone worse off.

Real life Examples [现实生活中的例子]

Example 2: The Group Project (Positive-Sum Thinking)

Scenario: Your team has a group project due Friday. Everyone is busy.

- Bad Approach: "I'll do as little as possible and hope others do the work." (This is the Prisoner's Dilemma)
- Result: If everyone thinks this way, the project fails. Everyone gets an F.
- Good Approach: "Let's each do what we're good at. You research, I'll write, they'll edit. We'll all get a better grade with less individual work."
- Game Theory Lesson: Cooperation creates a bigger pie for everyone. You all get B+ instead of risking an F.

Real life Examples [现实生活中的例子]

Example 3: The Last Slice of Cake (The Trade-Off)

Scenario: You and your friend both want the last slice of chocolate cake.

- Zero-Sum Solution: Fight over it, or cut it in half (both get less than they want).
- Creative Solution: "I'll trade you the cake slice for your extra movie ticket tomorrow." OR "You have the cake, and I'll choose the movie tonight."
- Game Theory Lesson: Find differences to trade. Maybe you value the cake more, they value something else more. Trading makes you both happier than simple splitting.

任 何 谈 判 的 三 大 关 键 博 弈 论 规 则
The 3 Key Game Theory Rules for Any Negotiation

- **Think "Win-Win," Not "I Win"**
  - Ask: "How can we BOTH be better off?" not "How do I beat you?"
  - Simple test: If the other person looks unhappy with the deal, it's probably not sustainable.
- **Your BATNA = Your Power**
  - BATNA = Best Alternative To a Negotiated Agreement
  - Translation: Your best "walk away" option.
  - Example: If you're selling your old video game, your BATNA might be:
    - Selling it online for $20
    - Trading it to your cousin
    - Keeping it
    - This means you shouldn't accept less than $20 from a friend.
- **Repeat the Game**
  - One-time interactions: People might be selfish.
  - Ongoing relationships: Cooperation pays off.
  - Example: If you borrow notes just before the test and never return the favor, friends stop helping. If you help each other all semester, everyone does better.