Module 4.1:
How to work with PM

We assume:

- you have a dedicated PM
- developers working full-time
- you're building an MVP (3-6 months)
- and you don't want to manage developers directly

What Your PM Should Be Doing

A good PM is your eyes and ears on the project.
They spot problems early, and keep things moving.

Run 2-Week Sprints

Sprint planning at the start, demo at the end, keep developers focused and unblocked.

Give You Visibility

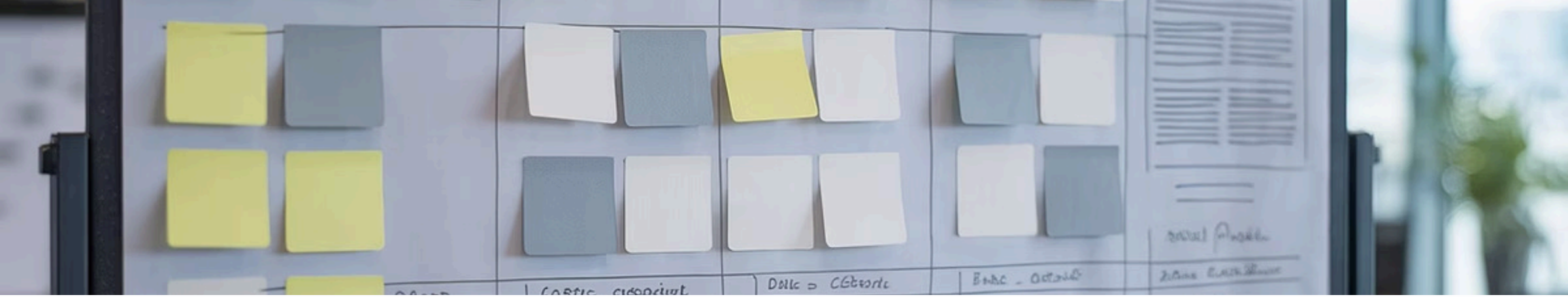
You know what's happening without asking. You see working software every 2 weeks. Problems surface early.

Protect Timeline & Budget

Flag slippage early, push back on scope creep, make trade-off recommendations.

Manage Team Day-to-Day

Developers know what to build, blockers get resolved fast, quality stays consistent.



The 2-Week Sprint: How It Should Work

Sprint Planning (Monday, Week 1 - 1 hour)

Who's there: PM + Dev team + You

What happens:

- review what got done last sprint
- pick features for this sprint
- break features into tasks
- developers estimate and commit

PM's output to you (same day): Email or Slack message with sprint plan.

Sprint Plan Example

Sprint 5 Plan (Oct 24 - Nov 4)

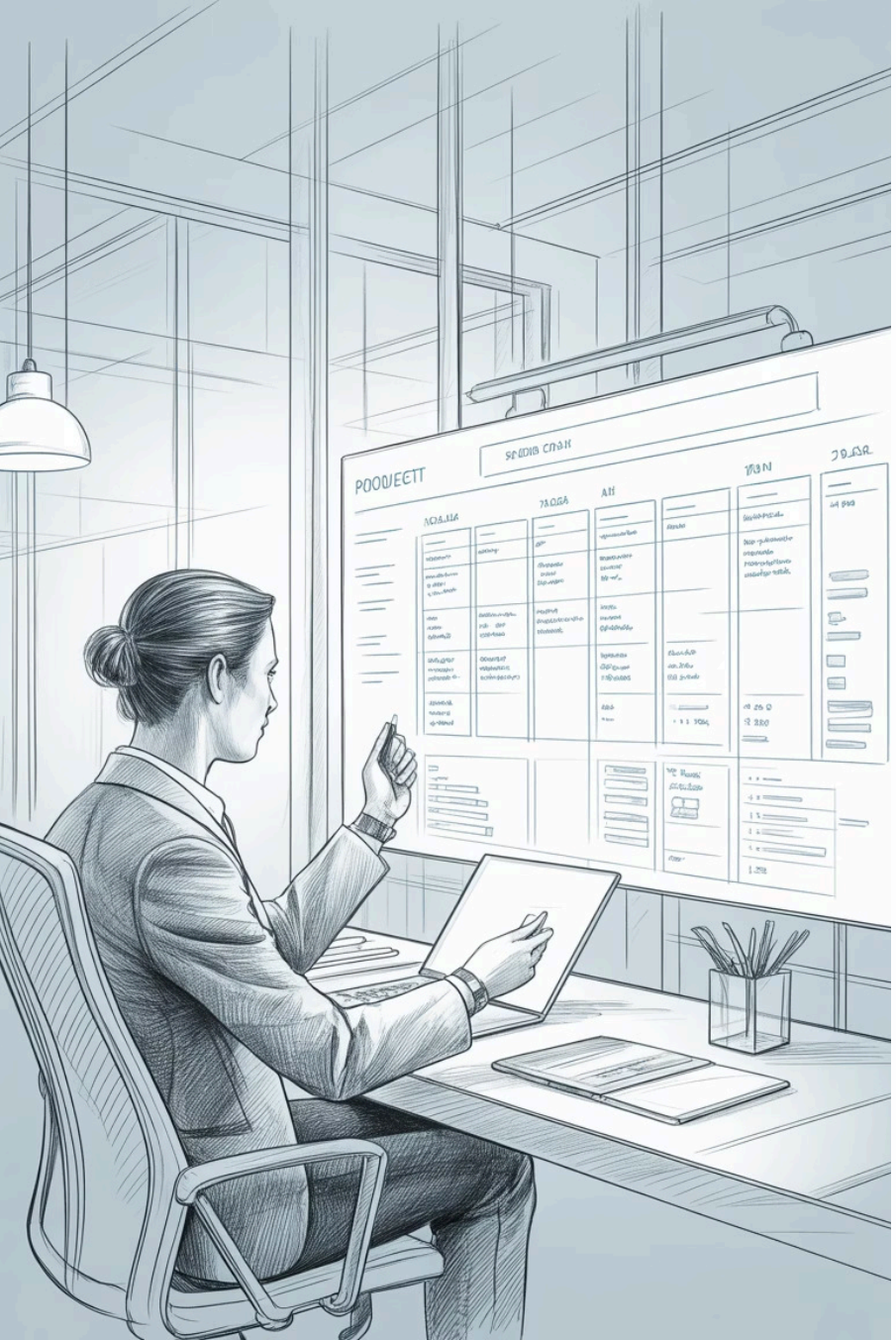
Completed last sprint:
✓ User authentication
✓ Profile creation
  Email notifications (pushed to this sprint)

This sprint we're building:
1. Email notifications (3 days - carried over)
2. Task creation flow (4 days)
3. Task assignment (3 days)

Team commitment: 8/10 confidence
Risk: None identified
Need from you: Approve notification email copy by Wednesday

Red Flags

- No written plan sent to you
- Plan is vague ("work on backend")
- No specific deliverables
- No ask/blocker mentioned



During Sprint

You Shouldn't Be Doing

- Daily standups with developers
- Checking in on individual tasks
- Asking "how's it going?"

Your PM handles this.

What Your PM Should Be Doing

- Daily standup with dev team (15 min)
- Unblocking developers same day
- Keeping you updated
- Maintaining task board

Mid-Sprint Check-In

> ## Sprint 5 Update (Day 4/10)
>
> **On track:**
> ✓ Email notifications - 80% done, testing tomorrow
> ✓ Task creation - backend done, frontend in progress
>
> **Watch item:**
> ⚠️ Task assignment - waiting on your feedback about permission levels (sent questions Tuesday, need answer by Friday)
>
> No blockers. Demo still set for Nov 4.

Red Flags

- Radio silence for 5+ days
- You find out about problems through developers
- Vague updates ("making progress")
- You're unblocking developers directly

Sprint Demo (Friday, Week 2 - 1 hour)

Who's there: PM + Developers + YOU (this one is important)

1

Part 1: Show Working Software

PM or developers demo what was built. You see it working live. You test it yourself during the call. (30 min)

2

Part 2: Retro & Next Steps

What went well? What slowed us down? Preview next sprint priorities. Timeline check. (30 min)

What Good Demos Look Like

01

You See Features Working

"Here's the task creation flow. I'm going to create a task now. There - it appears on the board. Your turn - want to try creating one?"

02

Honest About What's Not Done

"Task assignment is 70% done. The UI works, but there's a bug when you assign to multiple people. We'll fix it Monday. "

03

Clear on Timeline Impact

"We completed 2/3 planned features. Email notifications took longer because of spam filter requirements."

What Bad Demos Look Like

Code Walkthrough Instead of Working Software

"So if you look at this file structure, we implemented the API for task creation. Let me show you the code..."

Your response: "Can I see it working instead?"

Excuses Instead of Demos

"We would demo but there's a small bug that's preventing it from running. We're 95% done though."

Your response: "Show me the 95% that works."

PM Doesn't Know What Was Built

PM: "Hey dev team, who wants to show what they worked on?"

This means PM isn't managing the sprint.

Critical rule: If you don't see working software every 2 weeks, something is wrong.

Your Weekly Touchpoint (30 minutes)

When: Middle of each sprint (Wednesday or Thursday)
Purpose: Stay aligned, make decisions, keep PM unblocked

Example Good PM Update

✅ On track
Completed: 60% (email notifications done, task creation in progress)

Decisions needed from you:
1. Task templates - build now or push to v1.1?
   (Would add 1 week if we do now)
2. Should "delete task" be hard delete or move to archive?
3. Approve these notification email designs [link]

Next sprint preview:
Task editing + Task comments + Search functionality

Timeline: Still targeting Nov 30 launch
Budget: Used $32K of $50K (64% through timeline, 64% through budget ✅)

Risks: None currently

Decision-Making Protocol

Your PM should be filtering most questions from developers.

PM Should Handle

- Technical implementation details
- Minor UX decisions within spec
- Bug priority and triage
- Task sequencing and sprint planning

PM Should Bring to You

- Scope changes (new features)
- Timeline/budget impacts
- Product direction decisions
- User experience tradeoffs

Good Escalation from PM

The developer found an issue: our auth system doesn't support social login (Google/Facebook). Options:
A) Email/password only - no extra time
B) Add social login - adds 2 days and $600.
C) Use Auth0 - adds 1 day and $200/month.

My recommendation: Option A for MVP, add social in v1.1. Need your decision by Friday.

Bad Escalation

"Dev has a question about the login page. Can you jump on a call with them?"

Your response: "What's the question? Can you decide this?"

Red Flags

You're in the Dark

You have to ask for updates.
Surprises in demos.
You don't know what's being worked on.

Developers Reach Out to You Directly

"Quick question about the design..." "Should this feature work like X or Y?"

Demos Are Consistently Disappointing

Lots of "almost done" status.
Excuses instead of working features.

PM Is Just a Messenger

Forwards developer questions without filtering.
Doesn't push back on timeline slips.

Documentation Is Missing or Outdated

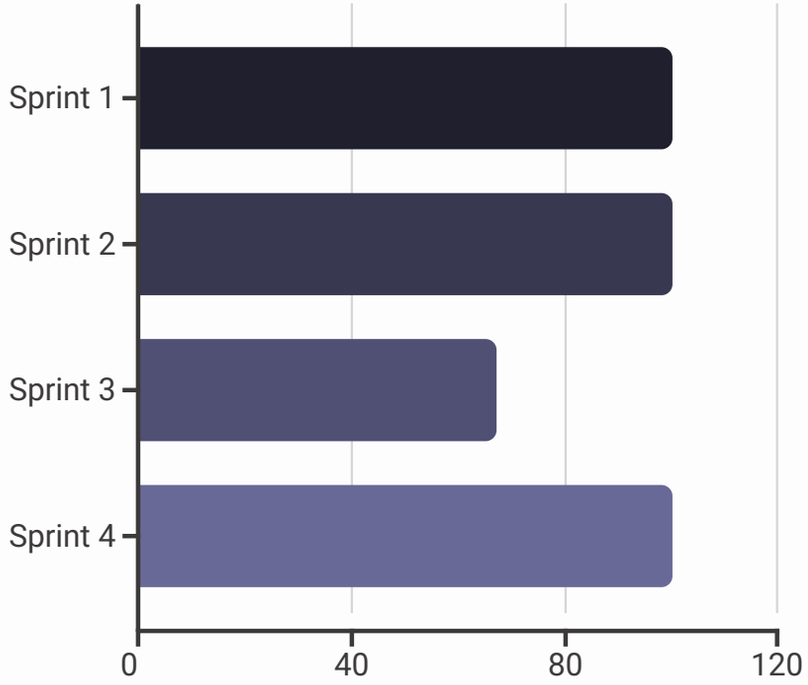
No sprint plans.
No decision log.
Bug tracker is mess or empty.

Timeline Is Mystery

Can't get straight answer on launch date.
No clear plan to address delays.

The 3 Metrics Your PM Should Track

Metric 1: Sprint Velocity



What you want to see:

80-100% completion rate.

If sprint 3 was 67%, sprint 4 should be adjusted (plan less). Pattern of consistent delivery.

What good PM does:

Adjusts future sprint capacity based on actual velocity.

Explains misses and what's being done differently. Doesn't over-promise.

Metrics 2 & 3: Bug Trends & Timeline Confidence

Metric 2: Bug Trends

Sprint	New	Fixed	Total Open	Status
2	5	0	5	✅ Normal
3	8	3	10	✅ Good
4	6	7	9	✅ Good
5	12	4	17	⚠️ Warning

What good PM does: Triages bugs (critical vs nice-to-have), allocates sprint time for bug fixes.

Metric 3: Timeline Confidence

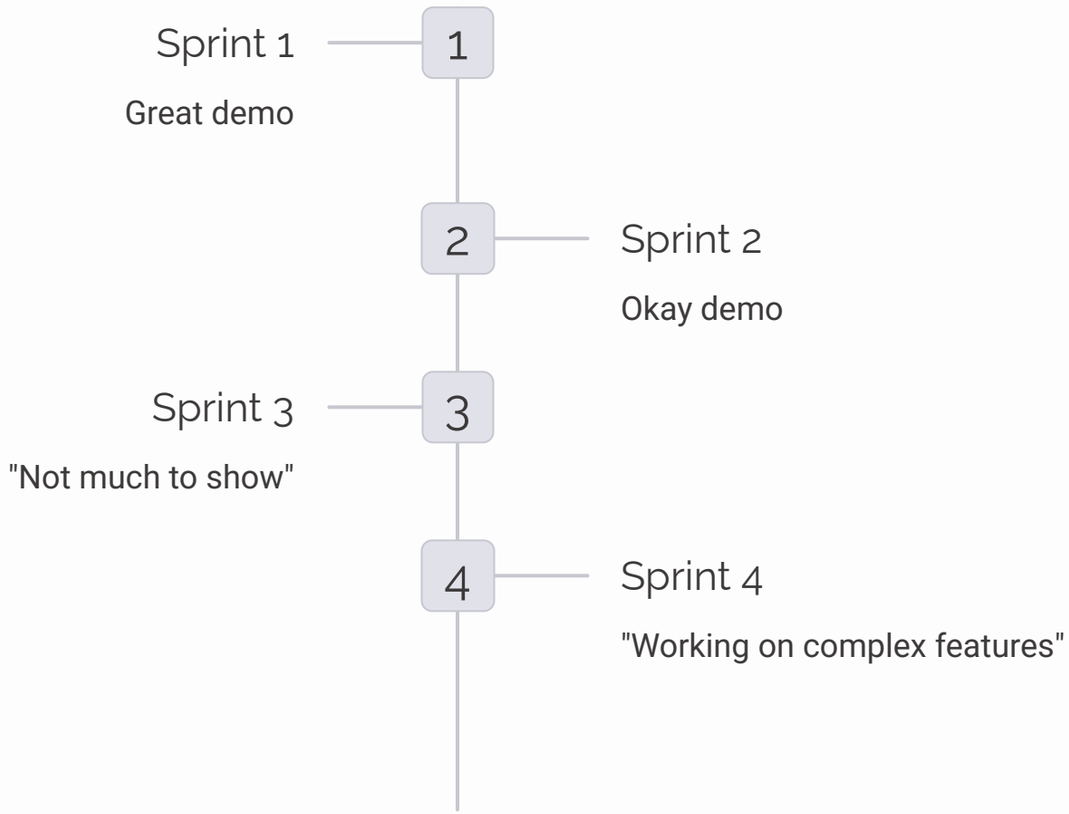
What good PM does when confidence drops:
Payment integration is more complex than estimated. Two options:

A) Push launch (1 week delay)

B) Simplify payments to credit card only, keep original date.
Recommendation: Option B.

When to Worry: Scenario 1

Demos Keep Disappointing



What this means: PM isn't managing scope or developers aren't productive.

What You Want to Hear

Payment API integration taking 2x longer than estimated.
Revised timeline: Dec 14 instead of Nov 30.
Or we can cut PayPal and keep Nov 30.
Your call.

Red Flag Response

Team is working hard, just some complexity.
We'll catch up.

When to Worry: Scenarios 2 & 3

Scenario 2: You Keep Getting Pulled Into Details

Pattern: Developers Slacking you directly with questions. PM forwarding every technical decision. You're making UI/UX calls that should be done.

What this means: PM isn't managing.

Scenario 3: Timeline Keeps Slipping

Pattern: Week 6: "Launch on Week 12" → Week 8: "Might need until Week 13" → Week 10: "Looking like Week 14" → Week 12: "Need at least Week 15"

What this means: PM isn't managing scope or tracking properly.

If PM can't give you clear answers about timeline slips => you might need a new PM or technical advisor to audit the project.

Documents Your PM Should Maintain

Sprint Plans

Updated every 2 weeks. Goals, tasks, team capacity, allocation (60h features, 12h bugs, 8h buffer).

Decision Log

Date, decision, reasoning, who decided. Example: Oct 15 - Use Stripe not PayPal for MVP - Better API, can add PayPal v1.1 - You + PM.

Bug Tracker

ID, issue, priority (High/Medium/Low), status, sprint. Organized and up-to-date.

Timeline Tracker

Original estimate, current estimate, completed sprints, remaining sprints, completed features percentage, risks identified.

Your PM should update these weekly and share with you.

Your PM Evaluation Checklist

Use this every 2-4 weeks to evaluate if your PM is doing their job:

Communication (Most Important)

- I get sprint plans without asking
- I get mid-sprint updates proactively
- I'm never surprised in demos
- I know what's happening without daily check-ins
- Problems are flagged early with options

Execution

- Demos happen every 2 weeks (no cancellations)
- Demos show working software
- Sprint commitments are mostly met (80%+)
- Bugs are triaged and tracked
- Timeline is realistic and clear

Team Management

- Developers don't reach out to me directly
- Team seems unblocked and productive
- Quality is consistent
- No drama or confusion

Decision Support

- PM brings me options, not just problems
- Recommendations have clear reasoning
- Trade-offs are quantified (time/cost)
- I'm making strategic decisions, not tactical ones

Great PM

16-20 checks: Keep them

Good PM

12-15 checks: Some areas to improve

Struggling PM

8-11 checks: Needs direct conversation

Wrong PM

<8 checks: Time to replace

Your Job vs PM's Job

Your Job (CEO)

- Set product vision and priorities
- Make strategic decisions (major scope/budget/timeline)
- Attend sprint demos (see progress)
- Remove business blockers
- Approve final launch

PM's Job

- Manage day-to-day execution
- Keep developers productive
- Run sprints (planning + demo)
- Track metrics (velocity, bugs, timeline)
- Flag problems early
- Filter 90% of decisions from you

What You Should NEVER Do

- Daily standups with developers
- Assign individual tasks to developers
- Debug issues or review code
- Answer technical questions from developers
- Micromanage sprint execution

Your PM is your proxy

They should make you feel confident about the project without consuming your time.

2	1	90%
Hours Per Week	Hour Per Week	Decisions Filtered
Max time you should spend on project management	30 min check-in + 30 min demo = you always know what's happening	By PM

If you're spending >2 hours/week on project management stuff, either:
a) Your PM isn't doing their job
b) You're micromanaging
c) Project is in crisis (temporary).

Good PM = you spend 1 hour/week and you always know what's happening.

That's how you work with a PM.