# 🧭 Key Sprint Elements for Small Companies

| Item | Purpose | Adaptation in Small Company |
|---|---|---|
| **Sprint Length** | Timebox for focus and delivery rhythm | ✅ *Maybe* – Start with 1-week sprints to move fast early on; switch to 2 weeks once things stabilize |
| **Sprint Planning** | Decide what to build and align expectations | ✅ *Maybe* – Can be combined with retro; focus more on setting clear sprint goals |
| **Daily Standup** | Sync progress and surface blockers | ✅ *Yes* – Keep it async or under 10 minutes in person |
| | | 🟡 *Can Skip* – Good for team morale |

# ❗ Under-Estimating Tasks

**Problem:**
Developers think the task is simple, but hidden complexity or edge cases make it take much longer.

**Why It Happens:**

- Lack of clarity or detail
- Overconfidence from past similar tasks
- Missing consideration for testing, integration, error handling

**Suggestions:**
✅ Ask: “What could go wrong?”
✅ Add buffer *only* for known risk areas
✅ Break task into smaller subtasks
✅ Use past tickets as size reference

# ⚠️ Over-Estimating Tasks

**Problem:**

Developers add a large buffer due to fear of missing deadlines or being judged.

**Why It Happens:**

- Low confidence in requirements
- Fear of being blamed for delays
- Estimating solo without peer support

**Suggestions:**

✅ Normalize that estimates are not commitments
✅ Estimate as a team or pair (sanity check)
✅ Encourage iteration: “Let’s aim for X, but adjust if needed”
✅ Build trust that re-scoping is OK

# 🔍 Hidden Work / Missing Scope

**Problem:**

A task looks simple but includes unaccounted work like infra, setup, or integration.

**Why It Happens:**

- Vague task definitions
- No clear checklist or “definition of done”
- No one catches the gaps during planning

**Suggestions:**

✅ Define done before estimating

✅ Ask: “What’s needed before/after this task?”

✅ Add a buffer if dependent on others (e.g. API readiness)

✅ Use spike tasks to explore unknowns

# 🧱 Task is Too Big or Vague

**Problem:**

Hard to estimate large or fuzzy tasks like "build dashboard" or "improve UX".

**Why It Happens:**

- Task not broken down
- No measurable outcome defined
- Ambiguous or exploratory work

**Suggestions:**

✅ Break down by deliverable or flow
✅ Include measurable goals or acceptance criteria
✅ Add a design or spike task if details are unclear
✅ Ensure it can be completed in 1–2 days max

# 🧍 Estimating Alone

**Problem:**

Estimates are made in isolation, leading to bias and no shared ownership.

**Why It Happens:**

- Fast-moving culture skips planning
- Devs fear wasting team time
- No habit of peer-checking estimates

**Suggestions:**

✅ Estimate in pairs or with the team
✅ Use async planning (e.g. Slack thread, emoji votes)
✅ Create a shared “mental model” of task sizes
✅ Make it safe to say “I’m unsure — need input”