RADIANT AGILITY TECHNOLOGY

Your Partner in Agile Excellence

🧩 Epic Splitting AI Prompts Library

Transform 6-Month Nightmares into Sprint-Ready Stories

🚀 The 4-Step AI Epic Splitting Framework

Step 1: Epic Analyzer (2 minutes)

Copy this prompt exactly, replace the bracketed sections with your information:

```
Analyze this epic and provide a comprehensive breakdown:

Epic: [PASTE YOUR EPIC HERE]

Team Context:
- Team size: [NUMBER OF PEOPLE]
- Tech stack: [YOUR TECHNOLOGY]
- Sprint capacity: [TYPICAL STORY POINTS PER SPRINT]

Please identify:
1. Core user journeys (the main paths users will take)
2. Technical dependencies (what must be built in order)
3. Business value clusters (related features that deliver
value together)
4. Testable increments (smallest pieces that can be
validated)
5. Risk areas (complex or uncertain parts)
6. Integration points (where this connects to existing
systems)

Format your response with clear headings for each section.
```

💡 **Pro Tip:**

Be specific about your team context and tech stack. The more context you provide, the better the AI suggestions will be.

Step 2: INVEST Story Creator (3 minutes)

Use this prompt to transform epic components into proper user stories:

```
Create user stories from this epic analysis that follow
INVEST principles:

Epic Components: [PASTE OUTPUT FROM STEP 1]

For each user story, ensure it is:
- Independent: Can be developed without waiting for other
stories
- Negotiable: Scope can be discussed and adjusted
- Valuable: Delivers clear business or user value
- Estimable: Team can reasonably size the effort
- Small: Can be completed within one sprint
- Testable: Has clear, verifiable acceptance criteria

Format each story as:
**Story Title:** [Descriptive title]
**As a** [user type]
**I want** [functionality]
**So that** [business value/outcome]

**Acceptance Criteria:**
- [Specific, testable criterion 1]
- [Specific, testable criterion 2]
- [Specific, testable criterion 3]

**Story Points Estimate:** [1, 2, 3, 5, 8]
**Priority:** [High/Medium/Low]
**Dependencies:** [List any dependencies on other stories]

Create 5-8 stories that cover the epic scope.
```

💡 **Pro Tip:**

After AI generates the stories, review them with your team. The AI gives you a great starting point, but your domain expertise will refine them perfectly.

Step 3: Dependency Mapper (2 minutes)

Use this prompt to identify and visualize story dependencies:

```
Create a dependency analysis for these user stories:

Stories: [PASTE YOUR USER STORIES FROM STEP 2]

Analyze and provide:

**Dependency Matrix:**
- Which stories must be completed before others can start
- Which stories can be developed in parallel
- Which stories have soft dependencies (helpful but not
required)

**Critical Path Analysis:**
- Identify the longest chain of dependent stories
- Highlight which stories, if delayed, would impact the most
other work
- Suggest the minimum viable subset for early value delivery

**Sprint Groupings:**
- Recommend how to group stories into 2-week sprints
- Consider team capacity: [YOUR SPRINT CAPACITY] story
points
- Prioritize by business value and dependency requirements

**Risk Mitigation:**
- Identify stories with the highest risk of delay
- Suggest backup plans if critical stories are blocked
- Recommend parallel work streams to reduce bottlenecks

Format as a clear visual description I can easily translate
into a diagram.
```

💡 **Pro Tip:**

Use the dependency analysis to inform your sprint planning. Stories with many dependencies should be prioritized to unblock future work.

Step 4: Story Sizing Assistant (1 minute)

Get consistent, context-aware story point estimates:

```
Help estimate story points for these user stories:

Stories: [PASTE YOUR STORIES]

Team Context:
- Historical velocity: [AVERAGE POINTS PER SPRINT]
- Team experience: [New/Intermediate/Expert with this type
of work]
- Definition of Done: [YOUR DOD CHECKLIST]
- Technical complexity factors: [API integrations, database
changes, UI complexity, etc.]

For each story, consider:
- Frontend development effort (UI, user interactions,
responsive design)
- Backend development effort (APIs, business logic, data
processing)
- Database work (schema changes, migrations, queries)
- Integration complexity (third-party services, internal
APIs)
- Testing effort (unit tests, integration tests, manual
testing)
- DevOps work (deployment, configuration, monitoring)

Provide story point estimates (1, 2, 3, 5, 8, 13) with
reasoning for each estimate.

Also flag any stories that seem too large (>8 points) and
suggest how to split them further.
```

💡 **Pro Tip:**

Use these estimates as a starting point for team discussion, not final decisions. The AI context helps make your planning poker sessions more focused and productive.

🎯 Bonus Prompts for Advanced Teams

Epic Health Checker (30 seconds)

Quick validation that your epic splitting was successful:

```
Evaluate the quality of this epic splitting:

Original Epic: [YOUR ORIGINAL EPIC]
Resulting Stories: [LIST OF SPLIT STORIES]

Rate each aspect (1-10) and provide specific feedback:

**INVEST Compliance:**
- Independence: How well can stories be developed
separately?
- Negotiability: How flexible is the scope of each story?
- Value: Does each story deliver standalone user/business
value?
- Estimability: Can the team confidently size these stories?
- Size: Will each story fit comfortably in one sprint?
- Testability: Are acceptance criteria clear and verifiable?

**Strategic Alignment:**
- Value delivery: Do the stories deliver value
incrementally?
- User experience: Will the user journey be coherent across
stories?
- Technical coherence: Is the technical approach sound?
- Risk distribution: Are risks spread across multiple
stories?

**Actionable Feedback:**
- Which stories need further splitting?
- Which stories might be combined?
- What dependencies need attention?
- What risks require mitigation?

Provide an overall readiness score (1-100) for sprint
planning.
```

Risk Assessment Specialist (2 minutes)

Identify potential risks and mitigation strategies:

```
Analyze risks for this epic splitting approach:

Epic: [ORIGINAL EPIC]
Split Stories: [YOUR STORY LIST]
Team Context: [SIZE, EXPERIENCE, CONSTRAINTS]
Timeline: [PROJECT DEADLINES]

Identify risks in these categories:

**Technical Risks:**
- Integration challenges between stories
- Technology unknowns or complex implementations
- Performance or scalability concerns
- Security or compliance requirements

**Business Risks:**
- Market timing or competitive pressure
- Stakeholder alignment and expectation management
- Resource availability and skill gaps
- Scope creep or requirement changes

**Process Risks:**
- Team coordination across multiple stories
- Testing and quality assurance coverage
- Deployment and rollout complexity
- User adoption and change management

For each risk, suggest:
- Probability (High/Medium/Low)
- Impact (High/Medium/Low)
- Mitigation strategies
- Contingency plans
```

📊 Real-World Example: E-commerce Checkout

Before Epic Splitting:

Epic: "As a user, I want a complete e-commerce checkout experience so I can purchase products online."

Size: 47 story points (impossible to complete in one sprint)

Timeline: 6+ months

Risk: Extremely high

Team Confidence: Very low

After AI Epic Splitting (8 minutes):

Story 1: Shopping Cart Management (3 points)

As a customer, I want to add, remove, and modify items in my cart so that I can control what I'm purchasing

Story 2: Discount Code Application (5 points)

As a customer, I want to apply discount codes to my order so that I can save money on my purchase

Story 3: Shipping Options Selection (3 points)

As a customer, I want to choose from available shipping options so that I can control delivery speed and cost

Story 4: Secure Payment Processing (8 points)

As a customer, I want to securely enter payment information so that I can complete my purchase safely

Story 5: Order Review and Confirmation (2 points)

As a customer, I want to review my order before confirming so that I can verify everything is correct

Results:

Total Stories: 5 (vs 1 epic)

Total Points: 21 (same scope, better organized)

Largest Story: 8 points (fits in one sprint)

Sprint Distribution: 2 sprints instead of 6+ months

Team Confidence: High

Risk Level: Manageable

✅ Epic Splitting Success Checklist

Before you start:

☐ Epic is clearly written with user value defined

☐ Team context information is gathered

☐ AI tool is ready (ChatGPT, Claude, etc.)

☐ 15 minutes blocked for the splitting process

During splitting:

☐ Step 1: Epic analyzed for components (2 min)

☐ Step 2: Stories created following INVEST (3 min)

☐ Step 3: Dependencies mapped and visualized (2 min)

☐ Step 4: Story points estimated with reasoning (1 min)

☐ Bonus: Health check completed if needed (30 sec)

After splitting:

☐ Stories reviewed with product owner

☐ Dependencies validated with technical team

☐ Acceptance criteria refined based on team input

☐ Stories added to backlog tool (Jira, Azure DevOps, etc.)

☐ Sprint planning scheduled with realistic expectations

🎯 Epic Splitting Quality Standards

🟢 Green Light (Ready for Sprint Planning):

- All stories are 8 points or smaller
- Each story delivers standalone user value
- Dependencies are clearly mapped and manageable
- Acceptance criteria are specific and testable
- Team confident they can estimate and deliver stories

🟡 Yellow Light (Needs Refinement):

- Some stories are 8+ points but can be split further
- Value delivery is clear but could be more incremental
- Dependencies exist but workarounds are possible
- Acceptance criteria need minor clarification

🔴 Red Light (Requires Major Rework):

- Stories are still too large (13+ points)
- Value delivery is unclear or only at epic completion
- Dependencies create major bottlenecks
- Acceptance criteria are vague or untestable
- Team has low confidence in estimates

💡 Pro Tips from the Field

Making AI Suggestions Better:

- Be specific about your context - team size, tech stack, industry
- Include examples of similar work your team has done
- Mention constraints - timeline pressures, resource limitations
- Reference your Definition of Done so AI understands your quality bar

Working with Your Team:

- Use AI as a starting point, not the final answer
- Review AI suggestions together in refinement sessions
- Let domain experts refine the technical and business details
- Document your reasoning for future reference

Continuous Improvement:

- Track your splitting success - how often do stories complete as planned?
- Note what AI missed and include that context in future prompts
- Refine your prompts based on what works for your team
- Share successes with other teams to spread the practice

🔧 Tool Integration Tips

For Jira Users:

- Copy AI-generated stories directly into Jira story format
- Use Jira's link feature to map dependencies visually
- Create custom fields for AI-generated risk assessments
- Set up automation rules based on story point thresholds

For Azure DevOps Users:

- Use work item templates that include the AI-generated acceptance criteria format
- Leverage dependency tracking features for AI-mapped relationships
- Create queries to find stories that might need re-splitting
- Use tags to mark AI-assisted vs manually created stories

For Linear/Notion/Other Tools:

- Adapt the story format to match your tool's structure
- Use relationship features to connect dependent stories
- Create templates with the AI prompt outputs pre-formatted
- Set up views that highlight large stories needing attention

📞

If your team needs support with epic splitting or AI-enhanced agile practices:

- Epic Splitting Workshops (2-4 hour facilitated sessions)
- AI Prompts Customization (tailored to your domain and tech stack)
- Team Training (teach your team to split epics independently)
- Ongoing Coaching (regular epic refinement support)

Contact: hello@radiantagility.tech