



# Interactive Prompt Builder - Detailed Flow

## The Core Innovation

This is the heart of Prompt Machine - where admins collaborate with Claude to build specialized AI tools.

## Conversation Flow Example

### Step 1: Initial Contact

```
javascript

// Admin starts new project
Admin: "I want to create a story writing assistant"

// Claude responds with targeted questions
Claude: "Great! I'll help you create a story writing assistant. Let me understand your vision better:

1. What genres should it support? (fantasy, sci-fi, romance, etc.)
2. Should it help with full stories or specific elements? (plot, characters, dialogue)
3. What level of customization do users need?
4. Any specific features you imagine? (story outlines, character sheets, world-building)"
```

### Step 2: Refinement Loop

```
javascript
```

*// Store conversation state*

```
const conversation = {  
  projectId: "uuid",  
  messages: [],  
  currentPhase: "discovery",  
  extractedRequirements: {  
    expertType: "Creative Writing Expert",  
    userInputFields: [],  
    outputFormat: null,  
    specialFeatures: []  
  }  
};
```

*// Claude guides the refinement*

**Admin:** "It should support fantasy and sci-fi, help with complete stories"

**Claude:** "Excellent! For a fantasy/sci-fi story writer, I suggest these user input options:

**\*\*Required Fields:\*\***

- Story **genre** (dropdown: fantasy/sci-fi/hybrid)
- Story **length** (short/medium/novel chapter)

**\*\*Optional Fields:\*\***

- Main character details
- Setting/world description
- **Tone** (dark/humorous/epic/mysterious)
- Plot elements to include

Would you like me to add any other options? How about features like:

- Character name generator
- Plot twist suggestions
- World-building assistant"

### Step 3: Prompt Construction

javascript

```

// Claude builds the actual prompt
const generatedPrompt = {
  systemPrompt: `You are an expert fantasy and science fiction writer...`,

  userPromptTemplate: `
Write a {{length}} {{genre}} story with the following details:
Main Character: {{characterDetails || 'Create your own'}}
Setting: {{worldDescription || 'Create your own'}}
Tone: {{tone}}
{{plotElements ? 'Include these elements: ' + plotElements : ''}}
`,

  fieldDefinitions: [
    {
      name: "genre",
      type: "select",
      options: ["Fantasy", "Sci-Fi", "Fantasy/Sci-Fi Hybrid"],
      required: true
    },
    {
      name: "length",
      type: "select",
      options: ["Short (1-3 pages)", "Medium (4-10 pages)", "Chapter (15-25 pages)"],
      required: true
    },
    // ... more fields
  ]
};

```

## Step 4: Testing Phase

```

javascript

// Admin can test with sample inputs
Claude: "Let's test your story writer! Here's a sample form..."

// Show preview of generated tool
// Admin provides test inputs
// Claude generates sample output
// Admin provides feedback

```

## Step 5: Deployment Configuration

javascript

**Claude:** "Your story writer looks great! Now **let's** configure the deployment:

1. **\*\*Tool Name\*\***: Fantasy & Sci-Fi Story Creator
2. **\*\*Subdomain\*\***: story-writer.tool.prompt-machine.com
3. **\*\*Description\*\***: Create engaging fantasy and sci-fi stories
4. **\*\*Monetization\*\***: Google AdWords (default placement)

Should we add any custom styling or branding?"

## Database Schema for Conversations

sql

*-- Store the entire conversation flow*

```
CREATE TABLE prompt_builder_sessions (  
  id UUID PRIMARY KEY,  
  project_id UUID REFERENCES projects(id),  
  status VARCHAR(50), -- active, completed, abandoned  
  current_phase VARCHAR(50), -- discovery, refinement, testing, deployment  
  conversation_history JSONB,  
  extracted_config JSONB,  
  created_at TIMESTAMPTZ,  
  completed_at TIMESTAMPTZ  
);
```

*-- Store reusable components*

```
CREATE TABLE prompt_components (  
  id UUID PRIMARY KEY,  
  type VARCHAR(50), -- field, feature, template  
  name VARCHAR(255),  
  configuration JSONB,  
  usage_count INTEGER DEFAULT 0  
);
```

## Claude Service Implementation

javascript

```
// services/claude.js - Prompt Builder specific
class PromptBuilderService {
  constructor() {
    this.phases = ['discovery', 'refinement', 'testing', 'deployment'];
  }

  async startSession(projectId, adminRequest) {
    const session = await this.createSession(projectId);

    // Initial Claude prompt for discovery
    const response = await this.sendToClaude({
      systemPrompt: this.BUILDER_SYSTEM_PROMPT,
      messages: [{
        role: 'user',
        content: adminRequest
      }],
      temperature: 0.7 // More creative for discovery
    });

    return this.processResponse(session, response);
  }

  async continueSession(sessionId, adminMessage) {
    const session = await this.loadSession(sessionId);

    // Add context about current phase
    const phaseContext = this.getPhaseContext(session.current_phase);

    const response = await this.sendToClaude({
      systemPrompt: this.BUILDER_SYSTEM_PROMPT + phaseContext,
      messages: session.conversation_history.concat({
        role: 'user',
        content: adminMessage
      })
    });

    return this.processResponse(session, response);
  }

  // Extract structured data from conversation
  extractConfiguration(messages) {
    // AI-powered extraction of fields, features, etc.
    // This is where Claude's understanding creates the tool config
  }
}
```

```
}  
}
```

## React Component for Builder

jsx

```
// app/src/components/PromptBuilder.jsx  
const PromptBuilder = ({ projectId }) => {  
  const [messages, setMessages] = useState([]);  
  const [currentPhase, setPhase] = useState('discovery');  
  const [isLoading, setLoading] = useState(false);  
  
  const sendMessage = async (message) => {  
    setLoading(true);  
  
    const response = await api.post('/prompts/builder/continue', {  
      projectId,  
      message  
    });  
  
    setMessages([...messages,  
      { role: 'user', content: message },  
      { role: 'assistant', content: response.data.reply }  
    ]);  
  
    setPhase(response.data.phase);  
    setLoading(false);  
  };  
  
  return (  
    <div className="prompt-builder">  
      <PhaseIndicator phase={currentPhase} />  
      <MessageHistory messages={messages} />  
      <InputArea onSubmit={sendMessage} disabled={isLoading} />  
      {currentPhase === 'testing' && <TestingPanel />}  
    </div>  
  );  
};
```

## System Prompt for Builder Claude

javascript

```
const BUILDER_SYSTEM_PROMPT = `
```

You are an expert AI prompt engineer helping administrators create specialized AI tools.

Your task is to guide them through creating prompts that will power single-page applications.

Phases:

1. **DISCOVERY:** Understand their vision, ask clarifying questions
2. **REFINEMENT:** Suggest specific fields, features, and configurations
3. **TESTING:** Help test the prompt with examples
4. **DEPLOYMENT:** Configure the final deployment settings

Guidelines:

- Always think about the end user who will use the tool
- Suggest practical, useful input fields
- Keep forms simple but powerful
- Provide clear examples
- Be encouraging and collaborative

Remember: You're building a prompt that will become a standalone AI tool!

```
`;
```

## Key Success Factors

1. **Conversation Memory:** Store entire conversation for context
2. **Phase Management:** Clear progression through builder phases
3. **Smart Extraction:** Claude extracts structure from natural conversation
4. **Live Testing:** Admin can test before deployment
5. **Template Library:** Reuse successful patterns

This is what makes Prompt Machine special - turning conversations into deployed AI tools!