

Enterprise Project Plan: [Project Name]

Planning Philosophy: This document follows the principle that design decisions should be explicit, not implicit. All code implementation follows this plan to avoid "AI slop" and ensure maintainable, testable, and understandable solutions.

Author: [Your Name]

Date: [YYYY-MM-DD]

Version: 0.1.0

Status: Draft | In Review | Approved | In Progress | Complete

How to Use This Template

1. **Start with Executive Summary:** Fill in the high-level what/why/who
2. **Define Scope Clearly:** Be explicit about goals AND non-goals
3. **Map Use Cases:** Real scenarios before technical solutions
4. **Document Constraints:** All assumptions and limitations up front
5. **Design Components:** Break down into clear, responsible parts
6. **Plan Testing:** How you'll validate it works
7. **Review Before Coding:** Get feedback on the plan, not the code
8. **Reference During Implementation:** Keep the plan updated as source of truth

Remember: The goal is to make design decisions explicit before implementation begins. This plan should be detailed enough that someone could understand the entire system without seeing the code.

Executive Summary

One-Sentence Description: [Describe what this project does in one clear sentence]

Problem Statement: [What problem are you solving? Why does it matter?]

Target Audience: [Who will use this? IT admins, end-users, developers, etc.]

Primary Value Proposition: [What is the main benefit this provides?]

Project Overview

Purpose

[Detailed explanation of why this project exists and what gap it fills]

Goals

1. [Primary goal]

2. [Secondary goal]
3. [Tertiary goal]

Non-Goals

[Explicitly state what this project will NOT do. This prevents scope creep.]

1. [What you're not building]
2. [Features you're intentionally excluding]
3. [Use cases you're not supporting]

Success Criteria

How will we know this project is successful?

- ☐ [Measurable success criterion 1]
 - ☐ [Measurable success criterion 2]
 - ☐ [Measurable success criterion 3]
-

Use Cases

Primary Use Cases

Use Case 1: [Name]

Actor: [Who is performing this action?]

Context: [When/where does this happen?]

Goal: [What are they trying to accomplish?]

Steps:

1. [Step 1]
2. [Step 2]
3. [Step 3]

Expected Outcome: [What should happen?]

Use Case 2: [Name]

[Repeat structure as above]

Secondary Use Cases

[Less common but still important scenarios]

Anti-Use Cases

[Scenarios that should NOT be supported, to clarify boundaries]

Technical Constraints

Must Have

- [Required technology/platform]
- [Required integration]
- [Required compatibility]

Assumptions

- [Assumption about the environment]
- [Assumption about user capabilities]
- [Assumption about existing infrastructure]

Limitations

- [Known limitation 1]
- [Known limitation 2]
- [Known limitation 3]

Dependencies

External Dependencies

- [External tool/service name] - [Why it's needed]
- [External library name] - [Why it's needed]

Internal Dependencies

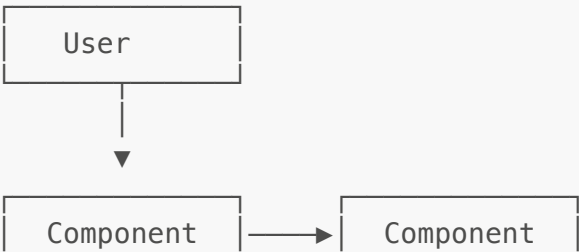
- [Internal system name] - [Why it's needed]
- [Internal tool name] - [Why it's needed]

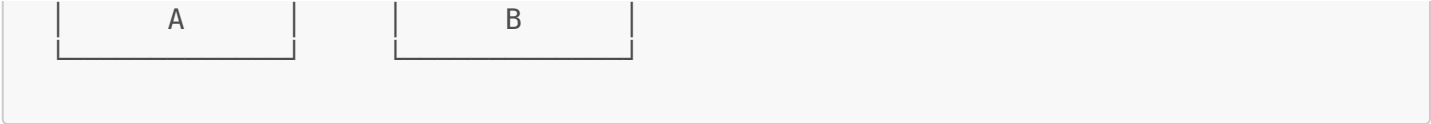
Architecture & Design

High-Level Architecture

[Describe the overall architecture in prose. What are the major components?]

[Optional: ASCII diagram or description of component relationships]





Core Components

Component 1: [Name]

Purpose: [What this component does]

Responsibilities:

- [Responsibility 1]
- [Responsibility 2]

Key Functions/Methods:

- `functionName()` - [What it does]
- `anotherFunction()` - [What it does]

Inputs: [What data/parameters does it receive?]

Outputs: [What does it produce/return?]

Component 2: [Name]

[Repeat structure as above]

Data Model

Key Data Structures

[Structure Name]:

```
{
  "field1": "type/purpose",
  "field2": "type/purpose",
  "nested": {
    "field3": "type/purpose"
  }
}
```

Data Flow

1. [Data enters the system from...]
2. [It gets processed by...]
3. [Results are stored/displayed in...]

State Management

[How does the system maintain state? Files, memory, database?]

- **Configuration State:** [How/where configuration is stored]
 - **Runtime State:** [What state is maintained during execution]
 - **Persistent State:** [What state survives restarts]
-

User Interface Design

User Experience Flow

1. [User does X]
2. [System responds with Y]
3. [User sees/does Z]

Interface Elements

Primary Interface: [Name/Type]

Type: [CLI, GUI, Web Interface, etc.]

Key Elements:

- [Element 1] - [Purpose and behavior]
- [Element 2] - [Purpose and behavior]

Sample Output/Mockup:

[Example of what the user will see]

Error Handling & User Feedback

Error States:

- [Error condition] → [How it's displayed to user]
- [Error condition] → [How it's displayed to user]

Progress Indicators:

- [When/how progress is shown]

Help/Documentation Access:

- [How users get help]
-

Implementation Approach

Technology Stack

- **Primary Language:** [Language and version]
- **Key Libraries/Frameworks:**
 - [Library 1] - [Purpose]
 - [Library 2] - [Purpose]
- **Build Tools:** [If applicable]
- **Testing Framework:** [If applicable]

File Structure

```
project-root/  
├── README.md  
├── [main-script].[ext]  
├── config/  
│   └── [config-files]  
├── resources/  
│   └── [assets/helpers]  
├── tests/  
│   └── [test-files]  
└── docs/  
    └── [documentation]
```

Core Algorithms/Logic

Algorithm 1: [Name]

Purpose: [What problem this solves]

Approach:

1. [Step 1 in plain language]
2. [Step 2 in plain language]
3. [Step 3 in plain language]

Edge Cases:

- [Edge case 1] → [How it's handled]
- [Edge case 2] → [How it's handled]

Algorithm 2: [Name]

[Repeat structure as above]

Configuration Strategy

User-Configurable Settings:

- **settingName** - [Default value] - [Purpose and valid values]
- **anotherSetting** - [Default value] - [Purpose and valid values]

Configuration Method: [File, CLI flags, environment variables, etc.]

Configuration Validation: [How invalid configs are detected and handled]

Testing Strategy

Test Modes

- **Development Mode:** [How it works, what it enables]
- **Test Mode:** [Default behavior for validation]
- **Production Mode:** [Final deployment behavior]

Test Scenarios

1. **Happy Path:** [Test the main success scenario]
2. **Edge Cases:**
 - [Edge case 1]
 - [Edge case 2]
3. **Error Conditions:**
 - [Error condition 1]
 - [Error condition 2]
4. **Performance:** [What performance characteristics to validate]

Validation Checklist

- ☐ [Check 1]
 - ☐ [Check 2]
 - ☐ [Check 3]
-

Error Handling

Error Categories

1. **User Errors:** [How we handle user mistakes]
2. **System Errors:** [How we handle system failures]
3. **Configuration Errors:** [How we handle misconfigurations]
4. **Network Errors:** [How we handle connectivity issues]

Error Response Strategy

Fatal Errors (stop execution):

- [Condition that's fatal]

- [Action taken]

Recoverable Errors (continue with warning):

- [Condition that's recoverable]
- [Action taken]

Silent Failures (log but continue):

- [Condition that's silent]
- [Action taken]

Logging Strategy

Log Levels:

- ERROR: [What gets logged as ERROR]
- WARNING: [What gets logged as WARNING]
- NOTICE: [What gets logged as NOTICE]
- INFO: [What gets logged as INFO]
- DEBUG: [What gets logged as DEBUG]

Log Location: [Where logs are stored]

Log Format: [Structure of log entries]

Security Considerations

Authentication/Authorization

[How is access controlled? Does this run with elevated privileges?]

Sensitive Data Handling

[How is sensitive data (passwords, tokens, etc.) managed?]

Input Validation

[What inputs are validated and how?]

Privilege Requirements

[What permissions/privileges are required to run this?]

Deployment Strategy

Prerequisites

System Requirements:

- OS: [Supported operating systems and versions]
- Dependencies: [Required software/tools]
- Permissions: [Required access levels]

Preparation Steps:

1. [Step 1 to prepare environment]
2. [Step 2 to prepare environment]
3. [Step 3 to prepare environment]

Installation Process

1. [Installation step 1]
2. [Installation step 2]
3. [Installation step 3]

Configuration Process

1. [Configuration step 1]
2. [Configuration step 2]
3. [Configuration step 3]

Verification Process

[How to verify the deployment was successful]

- ☐ [Verification check 1]
- ☐ [Verification check 2]
- ☐ [Verification check 3]

Maintenance & Updates**Version Strategy**

Version Format: [Semantic versioning? Other?]

Version Increments:

- Major: [When to increment major version]
- Minor: [When to increment minor version]
- Patch: [When to increment patch version]

Update Process

[How updates will be distributed and applied]

Backward Compatibility

[What compatibility guarantees are made?]

Deprecation Policy

[How will deprecated features be handled?]

Documentation Plan

User Documentation

- ☐ README with quick start guide
- ☐ Detailed usage instructions
- ☐ Configuration reference
- ☐ Troubleshooting guide
- ☐ FAQ

Developer Documentation

- ☐ Architecture overview
- ☐ Code comments and inline documentation
- ☐ API reference (if applicable)
- ☐ Contributing guidelines

Operational Documentation

- ☐ Deployment guide
 - ☐ Runbook for common issues
 - ☐ Change log
-

Support & Community

Support Channels

- **Primary Support:** [Where users should go first]
- **Community Support:** [Slack, forums, etc.]
- **Issue Tracking:** [GitHub issues, etc.]

Response Expectations

[What level of support is provided? Response time expectations?]

Contributing

[Is this open source? How can others contribute?]

Metrics & Monitoring

Success Metrics

[What metrics will you track to measure success?]

- [Metric 1]: [Target value]
- [Metric 2]: [Target value]
- [Metric 3]: [Target value]

Monitoring Plan

[How will you monitor the system in production?]

- [What will be monitored]
- [How alerts will be triggered]
- [Who responds to alerts]

Timeline & Milestones

Phase 1: [Name] (Target: [Date])

- ☐ [Deliverable 1]
- ☐ [Deliverable 2]
- ☐ [Deliverable 3]

Phase 2: [Name] (Target: [Date])

- ☐ [Deliverable 1]
- ☐ [Deliverable 2]
- ☐ [Deliverable 3]

Phase 3: [Name] (Target: [Date])

- ☐ [Deliverable 1]
- ☐ [Deliverable 2]
- ☐ [Deliverable 3]

Known Issues & Future Enhancements

Known Limitations

1. [Limitation 1] - [Impact and any workaround]
2. [Limitation 2] - [Impact and any workaround]

Future Enhancement Ideas

- 1. [Enhancement 1] - [Rationale]
- 2. [Enhancement 2] - [Rationale]
- 3. [Enhancement 3] - [Rationale]

Technical Debt

[Are there any shortcuts taken that should be addressed later?]

Questions & Decisions

Open Questions

- 1. [Question that needs to be answered before implementation]
- 2. [Question that needs to be answered before implementation]

Key Decisions Made

Decision	Rationale	Date	Decided By
[Decision 1]	[Why this was chosen]	[Date]	[Name]
[Decision 2]	[Why this was chosen]	[Date]	[Name]

Rejected Alternatives

Alternative	Why It Was Rejected
[Alternative approach 1]	[Reason for rejection]
[Alternative approach 2]	[Reason for rejection]

References & Resources

Inspiration

- [Project/Article that inspired this]
- [Related work]

Documentation

- [Relevant documentation link]
- [API documentation]

Related Projects

- [Similar project 1]
 - [Similar project 2]
-

Approval & Sign-off

Review Process

- ☐ Technical review by: [Name/Role]
- ☐ Security review by: [Name/Role]
- ☐ Stakeholder approval by: [Name/Role]

Sign-off

Plan Approved By: _____

Date: _____

Ready for Implementation: Yes / No

Implementation Notes

Once this plan is approved, use this section to track implementation progress and capture any deviations from the plan.

Implementation Log

- [Date]: [What was implemented]
- [Date]: [Any changes to the plan and why]
- [Date]: [Progress update]

Lessons Learned

[After implementation, what did you learn that would improve this plan for next time?]

End of Project Plan
