Factory Bootstrap Specification

Version: 1.0.0
Status: Draft
Last Updated: 2026-01-07
Owner: Code-Factory Core Team

Executive Summary

This specification defines the complete onboarding and initialization flow for the Spec-Driven Software Factory system. The goal is to achieve **zero-friction setup** where a developer can go from discovery to productive use in under 2 minutes with a single command.

---

1. One-Click Installation Flow

1.1 Installation Command

```
curl -sSL https://raw.githubusercontent.com/ssdajoker/Code-Factory/main/scripts/in-
stall.sh | sh
```

1.2 Installation Script Behavior

The `install.sh` script performs the following operations:

1. **Platform Detection**
   - Detect OS: Linux, macOS, Windows (WSL/Git Bash)
   - Detect architecture: amd64, arm64
   - Set appropriate binary name: `factory-{os}-{arch}`

2. **Binary Download**
   - Fetch latest release from GitHub Releases API
   - Download appropriate binary for platform
   - Verify checksum (SHA256)
   - Install to `/usr/local/bin/factory` (or `~/.local/bin/factory` if no sudo)
   - Make executable: `chmod +x`

3. **Verification**
   - Run `factory --version` to confirm installation
   - Display success banner with next steps

4. **Fallback Behavior**
   - If GitHub is unreachable: display manual installation instructions
   - If no binary for platform: offer Docker alternative
   - If checksum fails: abort with security warning

1.3 Alternative Installation Methods

Homebrew (macOS/Linux):

```
brew tap ssdajoker/factory
brew install factory
```

Winget (Windows):

```
winget install ssdajoker.factory
```

Docker:

```
docker pull ghcr.io/ssdajoker/factory:latest
docker run -it -v $(pwd):/workspace ghcr.io/ssdajoker/factory init
```

Nix:

```
nix-env -iA nixpkgs.factory
```

---

2. First-Run Initialization: `factory init`

2.1 Command Invocation

```
factory init
```

2.2 Initialization Flow

Phase 1: Welcome & Context Detection (0-5 seconds)

1. **Display Welcome Banner**

```
╔══════════════════════════════════════════════════════════╗
║ ║
║ 🏭 SPEC-DRIVEN SOFTWARE FACTORY 🏭 ║
║ ║
║ Turning specifications into reality ║
║ ║
╚══════════════════════════════════════════════════════════╝

Let's get you set up in under 2 minutes...
```

1. **Detect Current Context**
   - Check if inside a git repository: `git rev-parse --git-dir`
   - Check if `.factory/` directory exists
   - Check if `~/.factory/config.toml` exists
   - Determine if this is first-time setup or project-specific setup

2. **Prompt for Setup Type**
   ```
   What would you like to do?

[1] 🚀 Quick Start (GitHub integration + LLM auto-detect)
[2] 🔧 Manual Setup (configure everything yourself)
[3] 👥 Join Team (clone existing team configuration)
[4] 🆘 Help (learn more about Factory)

Choice [1-4]:
```

Phase 2: GitHub Integration Setup (5-30 seconds)

For Option 1 (Quick Start):

1. **GitHub OAuth Flow Initiation**

```
┌──────────────────────────────────────────────────────────┐
   │ 🔐 GitHub Integration                                     │
   ├──────────────────────────────────────────────────────────┤
   │                                                          │
   │ Factory needs GitHub access to:                          │
   │  ✓ Read repository metadata                              │
   │  ✓ Create and manage issues                              │
   │  ✓ Read and write pull requests                          │
   │  ✓ Access repository contents                            │
   │                                                          │
   │ Opening browser for GitHub authorization...              │
   │                                                          │
   │ 🌐 https://github.com/login/oauth/authorize?...          │
   │                                                          │
   │ [Press Enter to open browser, or Ctrl+C to skip]         │
   └──────────────────────────────────────────────────────────┘
```

2. **OAuth Device Flow (Fallback for Headless)**

```
┌──────────────────────────────────────────────────────────┐
   │ 🔐 GitHub Device Authorization                            │
   ├──────────────────────────────────────────────────────────┤
   │                                                          │
   │ Visit: https://github.com/login/device                   │
   │                                                          │
   │ Enter code: ABCD-1234                                    │
   │                                                          │
   │ Waiting for authorization... ⏳                           │
   │                                                          │
   └──────────────────────────────────────────────────────────┘
```

3. **Local HTTP Callback Server**
   - Start temporary HTTP server on `http://localhost:8765/callback`
   - Listen for OAuth callback with authorization code
   - Exchange code for access token
   - Shutdown server after successful exchange

4. **Token Storage**
   - Create `~/.factory/` directory with `0700` permissions
   - Store token in `~/.factory/github_token` with `0600` permissions
   - Encrypt token using system keyring if available (macOS Keychain, Windows Credential Manager, Linux Secret Service)

Phase 3: GitHub App Installation (30-60 seconds)

1. **Check for Existing Installation**
   - Query GitHub API: `GET /user/installations`
   - Look for “Code-Factory” app installation
   - If found, skip to repository selection

2. **Prompt for App Installation**

```
┌─────────────────────────────────────────────────────────┐
│ 📦 GitHub App Installation Required                     │
├─────────────────────────────────────────────────────────┤
│                                                         │
│ Factory uses a GitHub App for enhanced integration.     │
│                                                         │
│ The app needs these permissions:                        │
│  • Contents: Read & Write                               │
│  • Issues: Read & Write                                 │
│  • Pull Requests: Read & Write                          │
│  • Metadata: Read-only                                  │
│  • Webhooks: Read & Write (optional)                    │
│                                                         │
│ Opening installation page...                            │
│                                                         │
│ 🌐 https://github.com/apps/code-factory/installations/new │
│                                                         │
│ [Press Enter when installation is complete]            │
└─────────────────────────────────────────────────────────┘
```

3. **Repository Selection**
   - After app installation, fetch accessible repositories
   - If in git repo, auto-detect and confirm
   - Otherwise, present interactive list:
   ```

   Select repository to initialize Factory:

↓ ssdajoker/Code-Factory (current directory)
ssdajoker/my-app
ssdajoker/another-project
myorg/team-project

[↑/↓ to navigate, Enter to select, / to search]
```

1. **Installation Verification**
   - Test API access: `GET /repos/{owner}/{repo}`
   - Verify write permissions
   - Display confirmation:

```
✅ GitHub integration complete!

Repository: ssdajoker/Code-Factory
Access: Read & Write
App: Installed
```

Phase 4: LLM Configuration (60-90 seconds)

1. **Auto-Detection**

   ```
   ┌────────────────────────────────────────────────────────┐
   │ 🤖 LLM Configuration                                    │
   ├────────────────────────────────────────────────────────┤
   │                                                        │
   │ Detecting available LLM providers...                   │
   │                                                        │
   │ ⏳ Checking Ollama (localhost:11434)...                 │
   └────────────────────────────────────────────────────────┘
   ```

2. **Ollama Detection**
   - Attempt connection to `http://localhost:11434/api/tags`
   - If successful, list available models
   - Recommend models: `codellama`, `mistral`, `llama2`
   - If no models, offer to pull one:

```
✅ Ollama detected!

Available models:
• codellama:7b (recommended for code)
• mistral:7b (fast and capable)

Select default model:
↓ codellama:7b
mistral:7b
[Download another model...]
```

1. **BYOK (Bring Your Own Key) Flow**
   - If Ollama not detected, prompt for API keys:

   ```
   ┌────────────────────────────────────────────────────────┐
   │ 🔑 LLM API Configuration                                │
   ├────────────────────────────────────────────────────────┤
   │                                                        │
   │ Choose your LLM provider:                              │
   │                                                        │
   │ [1] OpenAI (GPT-4, GPT-3.5)                            │
   │ [2] Anthropic (Claude 3)                               │
   │ [3] Google (Gemini)                                    │
   │ [4] Azure OpenAI                                       │
   │ [5] Skip (configure later)                             │
   │                                                        │
   ```

```
│ Choice [1-5]:                                         │
└───────────────────────────────────────────────────────┘
```

2. **API Key Input**
   ```
   Enter your OpenAI API key:
   (Input will be hidden)

sk-********

Testing connection... ✅

Select default model:
↓ gpt-4-turbo-preview (recommended)
gpt-4
gpt-3.5-turbo
```

1. **Configuration Storage**
   - Store in `~/.factory/config.toml`:
   ```toml
   [llm]
   provider = "ollama" # or "openai", "anthropic", etc.
   model = "codellama:7b"
   endpoint = "http://localhost:11434"

# For BYOK providers
# api_key is stored in system keyring, not in this file
```

Phase 5: Project Initialization (90-120 seconds)

1. **Create Project Structure**

```
 ┌───────────────────────────────────────────────────────┐
   │ 📁 Initializing Project Structure                    │
   ├───────────────────────────────────────────────────────┤
   │                                                       │
   │ Creating directories...                               │
   │  ✓ /contracts/                                        │
   │  ✓ /reports/                                          │
   │  ✓ /.factory/                                         │
   │                                                       │
   │ Creating initial files...                             │
   │  ✓ contracts/README.md                                │
   │  ✓ .factory/config.toml                               │
   │  ✓ .gitignore (updated)                               │
   │                                                       │
   └───────────────────────────────────────────────────────┘
```

2. **Directory Structure Created**

```
project-root/
   ├── contracts/           # Specification documents
   │   ├── README.md
   │   └── .gitkeep
```

```
   ├── reports/              # Generated reports
   │   ├── README.md
   │   └── .gitkeep
   └── .factory/             # Project-specific config
       └── config.toml
```

3. **Update .gitignore**
   - Add Factory-specific ignores:

```
# Factory
   .factory/cache/
   .factory/temp/
   reports/*.tmp
```

4. **Create Initial Contract**
   - Generate `contracts/README.md` with template
   - Optionally create first spec from project README

Phase 6: Confirmation & Next Steps (120 seconds)

```
╔══════════════════════════════════════════════════════════════╗
║                                                              ║
║               ✅  SETUP COMPLETE!  ✅                        ║
║                                                              ║
╚══════════════════════════════════════════════════════════════╝

Configuration Summary:
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

📦 Repository:     ssdajoker/Code-Factory
🔐 GitHub:         Connected (OAuth)
🤖 LLM Provider:   Ollama (codellama:7b)
📁 Project:        Initialized

Next Steps:
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

1. Start the TUI:
   $ factory

2. Create your first specification:
   $ factory intake

3. Review existing code against specs:
   $ factory review

4. Learn more:
   $ factory help

Happy building! 🏭
```

3. GitHub OAuth & App Integration

3.1 OAuth Application Setup

Application Details:
- **Name:** Code-Factory
- **Homepage URL:** https://github.com/ssdajoker/Code-Factory
- **Authorization callback URL:** http://localhost:8765/callback
- **Device Flow:** Enabled (for headless environments)

OAuth Scopes Required:
- `repo` - Full control of private repositories
- `read:user` - Read user profile data
- `read:org` - Read organization membership (for team features)

3.2 GitHub App Setup

App Details:
- **Name:** Code-Factory
- **Description:** Spec-Driven Software Factory - Turn specifications into reality
- **Homepage URL:** https://github.com/ssdajoker/Code-Factory
- **Callback URL:** http://localhost:8765/callback
- **Webhook URL:** (optional) https://factory.example.com/webhooks
- **Webhook Secret:** (generated per installation)

Required Permissions:

Permission	Access	Reason
Contents	Read & Write	Read code, create/update spec files
Issues	Read & Write	Track spec changes, create change orders
Pull Requests	Read & Write	Review PRs against specs, suggest changes
Metadata	Read-only	Access repository metadata
Webhooks	Read & Write	(Optional) Real-time notifications

Webhook Events (Optional):
- `push` - Trigger automatic spec review
- `pull_request` - Review PR against specs
- `issues` - Track spec-related issues

3.3 API Calls for Automated Setup

3.3.1 OAuth Token Exchange

```
POST https://github.com/login/oauth/access_token
Content-Type: application/json

{
  "client_id": "Iv1.xxxxxxxxxxxxx",
  "client_secret": "xxxxxxxxxxxxxxxxxxxxxx",
  "code": "authorization_code_from_callback",
  "redirect_uri": "http://localhost:8765/callback"
}
```

Response:

```json
{
  "access_token": "gho_xxxxxxxxxxxxxxxxxxxx",
  "token_type": "bearer",
  "scope": "repo,read:user,read:org"
}
```

3.3.2 Verify Token & Get User Info

```
GET https://api.github.com/user
Authorization: Bearer gho_xxxxxxxxxxxxxxxxxxxx
```

Response:

```json
{
  "login": "ssdajoker",
  "id": 13389148,
  "name": "User Name",
  "email": "user@example.com"
}
```

3.3.3 List User Installations

```
GET https://api.github.com/user/installations
Authorization: Bearer gho_xxxxxxxxxxxxxxxxxxxx
```

Response:

```
{
  "total_count": 1,
  "installations": [
    {
      "id": 12345678,
      "app_id": 123456,
      "target_type": "User",
      "account": {
        "login": "ssdajoker"
      }
    }
  ]
}
```

3.3.4 List Installation Repositories

```
GET https://api.github.com/user/installations/12345678/repositories
Authorization: Bearer gho_xxxxxxxxxxxxxxxxxxxx
```

Response:

```
{
  "total_count": 5,
  "repositories": [
    {
      "id": 1125111279,
      "name": "Code-Factory",
      "full_name": "ssdajoker/Code-Factory",
      "private": false
    }
  ]
}
```

3.3.5 Create Installation Access Token

```
POST https://api.github.com/app/installations/12345678/access_tokens
Authorization: Bearer JWT_TOKEN
```

Response:

```
{
  "token": "ghs_xxxxxxxxxxxxxxxxxxxx",
  "expires_at": "2026-01-07T12:00:00Z",
  "permissions": {
    "contents": "write",
    "issues": "write",
    "pull_requests": "write"
  }
}
```

3.4 Device Flow (Headless Environments)

For SSH sessions, Docker containers, or CI/CD environments:

Step 1: Request Device Code

```
POST https://github.com/login/device/code
Content-Type: application/json

{
  "client_id": "Iv1.xxxxxxxxxxxxx",
  "scope": "repo read:user read:org"
}
```

Response:

```json
{
  "device_code": "3584d83530557fdd1f46af8289938c8ef79f9dc5",
  "user_code": "ABCD-1234",
  "verification_uri": "https://github.com/login/device",
  "expires_in": 900,
  "interval": 5
}
```

Step 2: Poll for Authorization

```
POST https://github.com/login/oauth/access_token
Content-Type: application/json

{
  "client_id": "Iv1.xxxxxxxxxxxxx",
  "device_code": "3584d83530557fdd1f46af8289938c8ef79f9dc5",
  "grant_type": "urn:ietf:params:oauth:grant-type:device_code"
}
```

Response (pending):

```json
{
  "error": "authorization_pending"
}
```

Response (success):

```json
{
  "access_token": "gho_xxxxxxxxxxxxxxxxxxxx",
  "token_type": "bearer",
  "scope": "repo,read:user,read:org"
}
```

---

4. Secret Storage Strategy

4.1 Storage Locations

Global Configuration:
- Path: `~/.factory/config.toml`

- Permissions: `0600` (read/write for owner only)
- Contents: Non-sensitive configuration (LLM provider, model, preferences)

GitHub Token:
- Path: `~/.factory/github_token`
- Permissions: `0600`
- Contents: OAuth access token (encrypted if keyring available)

Project Configuration:
- Path: `{project}/.factory/config.toml`
- Permissions: `0644` (readable by team)
- Contents: Project-specific settings (no secrets)

4.2 Encryption Strategy

Tier 1: System Keyring (Preferred)
- macOS: Keychain Access
- Windows: Credential Manager
- Linux: Secret Service API (GNOME Keyring, KWallet)

Tier 2: File Encryption (Fallback)
- Use AES-256-GCM encryption
- Derive key using Argon2id KDF with:
- User-provided password (prompted during setup), OR
- High-entropy random key stored securely in `~/.factory/master.key` (0600 permissions)
- Argon2id parameters: time=1, memory=64MB, threads=4, keyLen=32
- Store encrypted secrets in `~/.factory/secrets/*.enc`
- Each encrypted file contains: salt (16 bytes) + nonce (12 bytes) + ciphertext

Security Note: Never derive encryption keys from predictable values like machine ID or user ID alone. Always use proper KDF with high-entropy input.

Tier 3: Plain File (Last Resort)
- Store token in `~/.factory/github_token`
- Warn user about security implications
- Recommend using environment variable instead

4.3 Configuration File Format

~/.factory/config.toml:

```toml
[user]
name = "ssdajoker"
email = "user@example.com"

[github]
# Token stored separately in keyring or github_token file
token_storage = "keyring"  # or "file", "env"
default_org = "ssdajoker"

[llm]
provider = "ollama"
model = "codellama:7b"
endpoint = "http://localhost:11434"
temperature = 0.7
max_tokens = 4096

[llm.fallback]
provider = "openai"
model = "gpt-3.5-turbo"
# API key stored in keyring

[ui]
theme = "auto"  # auto, light, dark
editor = "vim"  # for editing specs
browser = "default"  # for OAuth flow

[modes]
default = "intake"  # Default mode on startup

[modes.review]
auto_fix = false
strict_mode = true

[modes.change_order]
auto_create_issue = true
```

{project}/.factory/config.toml:

```toml
[project]
name = "Code-Factory"
repository = "ssdajoker/Code-Factory"
initialized_at = "2026-01-07T12:00:00Z"

[contracts]
directory = "contracts"
format = "markdown"  # or "yaml", "json"

[reports]
directory = "reports"
format = "markdown"
auto_commit = false

[integrations]
github_app_installed = true
installation_id = 12345678

[team]
# Team members can clone and use without re-auth
shared_config = true
```

4.4 Environment Variable Override

Users can override token storage with environment variables:

```
export FACTORY_GITHUB_TOKEN="gho_xxxxxxxxxxxxxxxxxxxx"
export FACTORY_LLM_API_KEY="sk-xxxxxxxxxxxxxxxxxxxx"
export FACTORY_LLM_PROVIDER="openai"
export FACTORY_LLM_MODEL="gpt-4-turbo-preview"
```

---

5. User Experience & Feedback

5.1 Terminal Banners

Success Banner:

```
╔══════════════════════════════════════════════════════════╗
║                                                          ║
║              ✅  OPERATION SUCCESSFUL  ✅                ║
║                                                          ║
╚══════════════════════════════════════════════════════════╝
```

Error Banner:

```
╔══════════════════════════════════════════════════════════╗
║                                                          ║
║              ❌  OPERATION FAILED  ❌                     ║
║                                                          ║
╚══════════════════════════════════════════════════════════╝
```

Warning Banner:

```
┌──────────────────────────────────────────────────────────┐
│ ⚠️  WARNING                                               │
├──────────────────────────────────────────────────────────┤
│ This operation may have security implications.           │
└──────────────────────────────────────────────────────────┘
```

5.2 Progress Indicators

Spinner (for quick operations):

```
⠋ Connecting to GitHub...
⠙ Connecting to GitHub...
⠹ Connecting to GitHub...
⠸ Connecting to GitHub...
⠼ Connecting to GitHub...
```

Progress Bar (for longer operations):

```
Downloading binary...
[██████████████████████████████░░░░░░░░░░] 75% (15.2 MB / 20.0 MB)
```

Step Indicator:

```
[1/5] ✓ GitHub authentication
[2/5] ⏳ Installing GitHub App...
[3/5] ⏸️  LLM configuration
[4/5] ⏸️  Project initialization
[5/5] ⏸️  Verification
```

5.3 Error Messages

Clear and Actionable:

```
❌ Error: GitHub authentication failed

Reason: Invalid OAuth token

What to do:
1. Check your internet connection
2. Try running: factory auth reset
3. Re-run: factory init

Need help? Visit: https://github.com/ssdajoker/Code-Factory/issues
```

With Recovery Options:

```
❌ Error: Ollama not detected

Factory can work with:
• Ollama (local, free, private)
• OpenAI (cloud, paid, powerful)
• Anthropic Claude (cloud, paid, powerful)

What would you like to do?
[1] Install Ollama (recommended)
[2] Use OpenAI (enter API key)
[3] Use Anthropic Claude (enter API key)
[4] Skip for now (configure later)

Choice [1-4]:
```

---

6. Fallback Behavior

6.1 Offline / No GitHub Scenarios

Detection:
- Attempt to connect to `https://api.github.com`
- Timeout after 5 seconds
- Display offline mode banner

Offline Mode:

```
┌───────────────────────────────────────────────────────────┐
│ 🔌 Offline Mode                                           │
├───────────────────────────────────────────────────────────┤
│                                                           │
│ GitHub integration is unavailable.                        │
│                                                           │
│ You can still use Factory in local mode:                 │
│  ✓ Create and edit specifications                         │
│  ✓ Review local code against specs                        │
│  ✓ Generate reports                                       │
│                                                           │
│ GitHub features will be available when online.            │
│                                                           │
└───────────────────────────────────────────────────────────┘

Continue in offline mode? [Y/n]:
```

Local-Only Features:
- Spec creation and editing
- Code review against local specs
- Report generation
- LLM integration (if Ollama is available)

Disabled Features:
- GitHub issue creation
- PR review
- Remote spec synchronization
- Team collaboration

6.2 No LLM Available

Detection:
- Check for Ollama: `http://localhost:11434/api/tags`
- Check for API keys in config
- If both fail, enter manual mode

Manual Mode:

```
┌───────────────────────────────────────────────────────────┐
│ 🤖 No LLM Detected                                        │
├───────────────────────────────────────────────────────────┤
│                                                           │
│ Factory works best with an LLM, but you can still:       │
│  ✓ Create specifications manually                         │
│  ✓ Use templates for common patterns                      │
│  ✓ Review code with rule-based checks                     │
│                                                           │
│ To enable AI features:                                    │
│  • Install Ollama: https://ollama.ai                      │
│  • Or configure API key: factory config llm               │
│                                                           │
└───────────────────────────────────────────────────────────┘

Continue without LLM? [Y/n]:
```

Degraded Features:
- Spec generation: Use templates instead of AI
- Code review: Basic pattern matching instead of semantic analysis
- Change detection: Diff-based instead of intent-based

6.3 Insufficient Permissions

GitHub App Not Installed:

```
❌ Error: GitHub App not installed

Factory needs the GitHub App installed to access your repository.

Install now:
https://github.com/apps/code-factory/installations/new

After installation, run: factory init --reconnect
```

Missing Permissions:

```
⚠️  Warning: Limited GitHub access

Factory has read-only access to your repository.
Some features will be disabled:

Disabled:
  ❌ Creating/updating specs in repo
  ❌ Creating issues for change orders
  ❌ Commenting on PRs

Available:
  ✓ Reading existing specs
  ✓ Reviewing local code
  ✓ Generating local reports

To enable all features, grant write access:
https://github.com/apps/code-factory/installations/12345678
```

7. Team Setup Flow

7.1 First Team Member (Admin)

Setup:
1. Run `factory init` (full setup as described above)
2. Commit `.factory/config.toml` to repository
3. Share repository with team

Committed Configuration:

```toml
# .factory/config.toml (committed to repo)
[project]
name = "Code-Factory"
repository = "ssdajoker/Code-Factory"
team_mode = true

[contracts]
directory = "contracts"
format = "markdown"

[reports]
directory = "reports"
format = "markdown"

# Note: No secrets in this file!
# Team members will authenticate individually
```

7.2 Additional Team Members

Setup:
1. Clone repository: `git clone https://github.com/ssdajoker/Code-Factory.git`
2. Run `factory init --team`

Team Init Flow:

```
╔═══════════════════════════════════════════════════════════╗
║                                                           ║
║        🏭  JOINING TEAM PROJECT  🏭                        ║
║                                                           ║
╚═══════════════════════════════════════════════════════════╝

Detected existing Factory configuration!

Project: Code-Factory
Owner: ssdajoker
Team Mode: Enabled

To join this project, you need to:
1. Authenticate with GitHub (your own account)
2. Configure your LLM preferences

This will take about 1 minute...

[Press Enter to continue]
```

Simplified Flow:
- Skip project initialization (already configured)
- Only authenticate GitHub (personal token)
- Only configure LLM (personal preference)
- Inherit project settings from `.factory/config.toml`

Result:

```
✅ Team setup complete!

You're now connected to: Code-Factory

Your personal settings:
  GitHub: authenticated as @teammember
  LLM: ollama (codellama:7b)

Project settings (shared):
  Contracts: /contracts
  Reports: /reports

Start working: factory
```

7.3 Team Synchronization

Automatic Sync:
- Pull latest specs: `git pull origin main`
- Factory detects changes automatically
- No manual sync needed

Conflict Resolution:
- If specs conflict, Factory shows diff
- User chooses: keep local, use remote, or merge
- Changes tracked in change order log

---

8. Security Considerations

8.1 Token Security

Best Practices:
- Never commit tokens to repository
- Use system keyring when available
- Rotate tokens regularly (prompt user every 90 days)
- Revoke tokens on `factory auth logout`

Token Scopes:
- Request minimum necessary scopes
- Explain each scope to user during auth
- Allow user to decline optional scopes

8.2 Data Privacy

Local-First:
- All specs and reports stored locally
- GitHub used only for synchronization
- LLM queries can be local (Ollama) or cloud (BYOK)

Cloud LLM Privacy:
- Warn user when using cloud LLMs
- Option to redact sensitive data before sending
- Option to use local Ollama for sensitive projects

8.3 Audit Trail

Logging:
- Log all GitHub API calls to `~/.factory/logs/github.log`
- Log all LLM queries to `~/.factory/logs/llm.log`
- Rotate logs daily, keep 30 days

User Control:
- `factory logs show` - View recent activity
- `factory logs clear` - Clear all logs
- `factory privacy` - Review privacy settings

---

9. Testing & Validation

9.1 Installation Testing

Test Matrix:
- OS: Linux (Ubuntu, Fedora), macOS (Intel, ARM), Windows (WSL, Git Bash)
- Architecture: amd64, arm64
- Network: Online, offline, slow connection
- Permissions: sudo, non-sudo

Validation:
- Binary downloads correctly
- Checksum verification works
- Installation to correct path
- Executable permissions set
- Version command works

9.2 OAuth Flow Testing

Test Scenarios:
- Browser available (desktop)
- No browser (SSH session)
- Callback server port blocked
- User cancels authorization
- Token exchange fails
- Invalid token

Validation:
- Graceful fallback to device flow
- Clear error messages
- Recovery instructions provided
- No hanging processes

9.3 LLM Detection Testing

Test Scenarios:
- Ollama running with models
- Ollama running without models
- Ollama not installed
- API key provided

- Invalid API key
- No LLM available

Validation:
- Correct provider detected
- Model selection works
- Fallback to manual mode
- Clear instructions for setup

---

10. Success Metrics

10.1 Time to First Value

Target: < 2 minutes from `curl` to first spec created

Measurement:
- Track time from installation start to `factory init` completion
- Track time from init to first spec creation
- Log metrics to `~/.factory/metrics.log` (opt-in)

10.2 Setup Success Rate

Target: > 95% successful first-time setup

Measurement:
- Track completion of each setup phase
- Track fallback usage (offline, no LLM, etc.)
- Track error rates and types

10.3 User Satisfaction

Target: > 4.5/5 stars for setup experience

Measurement:
- Optional feedback prompt after setup
- GitHub issue sentiment analysis
- Community feedback

---

11. Future Enhancements

11.1 Phase 2 Features

- **Auto-update:** `factory update` to update binary
- **Plugin system:** Extend Factory with custom modes
- **Cloud sync:** Optional cloud backup of specs
- **Web UI:** Browser-based interface (localhost:3333)

11.2 Phase 3 Features

- **CI/CD integration:** GitHub Actions, GitLab CI
- **Slack/Discord notifications:** Real-time alerts
- **Multi-repo support:** Manage multiple projects

- **Spec marketplace:** Share and discover spec templates

Appendix A: Command Reference

```
# Installation
curl -sSL https://raw.githubusercontent.com/ssdajoker/Code-Factory/main/scripts/in-
stall.sh | sh

# Initialization
factory init                    # Full setup
factory init --team             # Join existing team project
factory init --offline          # Skip GitHub integration
factory init --no-llm           # Skip LLM configuration

# Authentication
factory auth login              # Authenticate with GitHub
factory auth logout             # Revoke token and logout
factory auth status             # Check authentication status
factory auth reset              # Reset and re-authenticate

# Configuration
factory config show             # Show current configuration
factory config edit             # Edit configuration file
factory config llm              # Configure LLM provider
factory config github           # Configure GitHub integration

# Modes
factory                         # Start TUI (default mode)
factory intake                  # Start in INTAKE mode
factory review                  # Start in REVIEW mode
factory change-order            # Start in CHANGE_ORDER mode
factory rescue                  # Start in RESCUE mode

# Utilities
factory version                 # Show version
factory help                    # Show help
factory doctor                  # Diagnose issues
factory logs                    # View logs
```

Appendix B: File Structure Reference

```
~/.factory/                      # Global configuration
├── config.toml                  # User preferences
├── github_token                 # OAuth token (encrypted)
├── logs/                        # Activity logs
│   ├── github.log
│   └── llm.log
└── cache/                       # Temporary cache

{project}/.factory/              # Project configuration
├── config.toml                  # Project settings (committed)
├── cache/                       # Local cache (gitignored)
└── temp/                        # Temporary files (gitignored)

{project}/contracts/             # Specifications (committed)
├── README.md
├── system_architecture.md
└── feature_specs/

{project}/reports/               # Generated reports (gitignored)
├── README.md
└── review_2026-01-07.md
```

End of Specification