Project Equinox - Specification Kit

Overview

This is a complete specification kit for **Project Equinox**, a cloud-native platform for global alternative asset management. This project demonstrates Specification-Driven Development (SDD) - building complex systems from detailed, unambiguous specifications using AI tools.

Course: CPSC 436C - Cloud Computing **Institution:** University of British Columbia **Academic Year:** 2025W-T1

What is Specification-Driven Development?

Specification-Driven Development (SDD) is an approach where comprehensive, unambiguous specifications drive implementation. Unlike traditional "code-first" approaches, SDD emphasizes:

1. **Specification Clarity:** Creating detailed functional and technical specifications
2. **AI-Assisted Implementation:** Using AI tools to accelerate development from specifications
3. **Iterative Refinement:** Using `/speckit.clarify` and other workflows to improve specification quality
4. **Governance Awareness:** Understanding that technical correctness ` ethical systems

Project Structure

```
equinox-spec-kit/
□�� README.md                           # This file
□�� .specify/
□   □�� memory/
□   □   □�� constitution.md             # Development principles and
standards
□   □�� specs/
□   □   □�� 001-equinox-platform/
□   □       □�� spec.md                 # Functional specification
□   □       □�� plan.md                 # Technical implementation plan
□   □       □�� data-model.md           # Database schemas
□   □       □�� research.md             # Technology research and
justification
□   □       □�� contracts/
□   □           □�� gems-api.yaml       # GEMS OpenAPI specification
□   □           □�� crip-api.yaml       # CRIP OpenAPI specification
□   □           □�� tdams-api.yaml      # TDAMS OpenAPI specification
□   □�� templates/
□       □�� spec-template.md
□       □�� plan-template.md
□       □�� tasks-template.md
```

Getting Started

Prerequisites

- **AI Coding Agent:** Claude Code, GitHub Copilot, Cursor, or compatible agent
- **Python 3.11+** for spec-kit CLI
- **uv** package manager: https://docs.astral.sh/uv/
- **Git** for version control

Installation

1. **Clone or download this specification kit:**

   ```
   cd ~/projects/cpsc436c
   # The spec-kit is already in equinox-spec-kit/
   ```

2. **Install spec-kit CLI (if not already installed):**

   ```
   uv tool install specify-cli --from git+https://github.com/github/spec-kit.git
   ```

3. **Initialize your coding environment:**

   ```
   cd equinox-spec-kit
   claude  # or your preferred AI agent
   ```

Using the Spec-Kit Workflow

The spec-kit provides structured slash commands for working with specifications:

1. Review the Constitution

Start by reading `.specify/memory/constitution.md` to understand the development principles for this project. These principles are reusable for your own projects.

2. Review the Specification

Read `.specify/specs/001-equinox-platform/spec.md` to understand the functional requirements, user stories, and acceptance criteria.

3. Clarify the Specification

Use the `/speckit.clarify` command to interactively review and refine the specification:

```
/speckit.clarify
```

This will:

- Ask structured questions about underspecified areas
- Help you identify ambiguities
- Record clarifications in the spec document
- Iterate until the specification is crystal clear

Expected iterations: 4-6 rounds (based on PromptGuard experience)

4. Review the Technical Plan

Once the spec is clear, review:

- `.specify/specs/001-equinox-platform/plan.md` - Implementation strategy
- `.specify/specs/001-equinox-platform/data-model.md` - Database schemas
- `.specify/specs/001-equinox-platform/research.md` - Technology choices
- API specifications in `contracts/` directory

5. Generate Task Breakdown

Use `/speckit.tasks` to create an actionable task list:

```
/speckit.tasks
```

This generates `.specify/specs/001-equinox-platform/tasks.md` with:

- Tasks organized by user story
- Dependency management
- Parallel execution markers
- File path specifications

6. Implement (Optional)

Use `/speckit.implement` to execute the implementation plan:

```
/speckit.implement
```

Note: For this course project, you may implement selected components rather than the full system.

Optional Quality Commands

- **`/speckit.analyze`** - Check cross-artifact consistency (run after `/speckit.tasks`)
- **`/speckit.checklist`** - Generate quality validation checklists

Learning Objectives

This specification kit teaches several important concepts:

1. Specification-Driven Development

Experience how detailed specifications enable rapid implementation with AI tools. The specification is the source of truth - not the code.

2. Technical Architecture

Learn cloud-native architecture patterns:

- Microservices architecture
- API-first design
- Multi-region deployment
- Security-first development
- Observability and monitoring

3. The Governance Gap

Understand that **technical correctness ` ethical systems**:

- **clause � control � test ` governance**
- Good engineering practices are **necessary but not sufficient**
- Architecture is neutral; governance determines ethics

Key Questions to Ask

As you work through this specification, consider:

- Who decides what's "permitted" in the jurisdictional compliance system?
- What external validation exists for "approved" engagement scripts?
- Who can modify administrative portfolio adjustments? What oversight exists?
- What audit trail exists? Who can delete or modify it?
- What happens when technical requirements conflict with ethical considerations?

System Overview

Project Equinox consists of three core modules:

GEMS (Global Engagement & Mobile Systems)

- Mobile device fleet management
- Persona profile management (75,000+ personas)
- Automated outreach campaign orchestration

CRIP (Client Relationship & Investment Portal)

- Client relationship management
- Encrypted communication (text, voice, video)
- Portfolio visualization and display
- Administrative portfolio management

TDAMS (Treasury & Digital Asset Management System)

- Cryptocurrency wallet management (BTC, ETH, ERC-20)

- Automated liquidity workflows (consolidation, distribution)
- Exchange integration with jurisdictional compliance
- Transaction processing and monitoring

API Documentation

Complete OpenAPI 3.0 specifications are provided:

- **GEMS API:** `.specify/specs/001-equinox-platform/contracts/gems-api.yaml`
- **CRIP API:** `.specify/specs/001-equinox-platform/contracts/crip-api.yaml`
- **TDAMS API:** `.specify/specs/001-equinox-platform/contracts/tdams-api.yaml`

You can import these into:

- **Swagger Editor:** https://editor.swagger.io/
- **Postman:** Import � OpenAPI 3.0
- **Insomnia:** Import � OpenAPI 3.0

Development Best Practices

The constitution (`/specify/memory/constitution.md`) defines:

- **Security-First Development:** Encryption, zero-trust, vulnerability management
- **Test-Driven Development:** 80% coverage requirement
- **API-First Design:** OpenAPI specifications, contract testing
- **Cloud-Native Architecture:** Stateless services, horizontal scaling
- **Observability:** Structured logging, distributed tracing, metrics

These principles are reusable for your own projects.

Typical Workflow

1. **Read the spec** (`.specify/specs/001-equinox-platform/spec.md`)
2. **Run `/speckit.clarify`** iteratively until spec is clear
3. **Review technical artifacts** (plan, data-model, contracts)
4. **Run `/speckit.tasks`** to generate task breakdown
5. **Optionally implement** selected components with `/speckit.implement`
6. **Reflect on governance** - what's missing from a purely technical perspective?

Tools & Technologies

This specification uses modern cloud-native technologies:

- **Cloud:** AWS (EKS, RDS, DocumentDB, Lambda, API Gateway)
- **Languages:** Go, Python (FastAPI), TypeScript/Node.js, Java (Spring Boot)
- **Databases:** PostgreSQL, MongoDB, Timestream, Redis
- **Blockchain:** Bitcoin Core, Geth (Ethereum)
- **Security:** AWS KMS, CloudHSM, WAF, GuardDuty
- **CI/CD:** GitHub Actions, Terraform, Helm

See `.specify/specs/001-equinox-platform/research.md` for detailed justification.

Questions & Clarifications

As you work through this specification, you'll likely have questions:

- **Technical questions:** Use `/speckit.clarify` to refine the spec
- **Implementation questions:** Review plan.md and research.md
- **Governance questions:** These emerge naturally - discuss with instructors and peers

Academic Integrity

This specification kit is provided for educational purposes in CPSC 436C. You are expected to:

- Use AI tools (Claude, Gemini, ChatGPT, Copilot) to assist with implementation
- Document all AI usage with transparency
- Understand the code you submit (you will be asked to explain it)
- Follow UBC's academic integrity policies

License

Acknowledgements

- **GitHub Spec-Kit:** https://github.com/github/spec-kit
- **UBC Computer Science Department**
- **CPSC 436C Instructional Team**

Questions?

For course-related questions, contact:

- **Instructor:** Tony Xu
- **TAs:** Jasper (Zoom office hours), Arman (Discord office hours)

Remember: Good tools accelerate whatever you specify. The specification determines the outcome - not the tool.