Day 1: Foundation Exercises - HX-Infrastructure Knowledge Base

Overview

These exercises provide hands-on practice with the HX-Infrastructure Knowledge Base project, applying Day 1 foundation concepts in a real-world context.

Exercise 1: Repository Analysis and Setup (60 minutes)

Objective

Analyze the current state of the HX-Infrastructure Knowledge Base and establish the foundation for integration work.

Tasks

1.1 Repository Structure Analysis (20 minutes)

```
cd /home/ubuntu/github_spec_training/HX-Infrastructure-Knowledge-Base

# Analyze current structure
find . -type f -name "*.md" | sort
find . -type d | sort

# Document findings
mkdir -p docs/analysis
```

Deliverable: Create `docs/analysis/repository-structure.md` documenting:
- Current directory structure
- Existing files and their purposes
- Identified gaps and opportunities
- Recommended structure enhancements

1.2 Content Gap Analysis (20 minutes)

Review the README.md file and identify placeholder sections that need content:

```
# Review README content
cat README.md | grep -n "placeholder"
cat README.md | grep -n "TODO"
cat README.md | grep -n "docs/"
```

Deliverable: Create `docs/analysis/content-gaps.md` listing:
- All placeholder sections identified
- Priority levels for content creation
- Estimated effort for each section
- Dependencies between sections

1.3 Workflow Assessment (20 minutes)

Analyze the existing GitHub Actions workflow:

```
# Review existing workflow
cat .github/workflows/connectivity-check.yml

# Test workflow functionality
git add docs/analysis/
git commit -m "Add initial analysis documentation"
git push origin main
```

Deliverable: Create `docs/analysis/workflow-assessment.md` documenting:
- Current workflow capabilities
- Enhancement opportunities
- Additional workflows needed
- Integration with training program

Exercise 2: Specification Creation (90 minutes)

Objective

Create comprehensive specifications for knowledge base integration using GitHub Spec Kit methodology.

Tasks

2.1 Integration Specification (30 minutes)

Use GitHub Spec Kit to create a specification for the overall integration project:

```
# Initialize specification project
uvx --from git+https://github.com/github/spec-kit.git specify init hx_kb_integration --ai copilot

cd hx_kb_integration
```

In your AI agent, use `/specify` to create a specification including:
- **Project Scope:** What will be integrated and why
- **Success Criteria:** Measurable outcomes for integration
- **Constraints:** Time, resource, and technical limitations
- **Stakeholders:** Who will use and maintain the knowledge base
- **Dependencies:** External requirements and prerequisites

Deliverable: Complete specification document with validation from AI agent

2.2 Content Integration Specification (30 minutes)

Create detailed specifications for content integration:

```
# Create content-specific specification
uvx --from git+https://github.com/github/spec-kit.git specify init content_integration --ai copilot
```

Specify requirements for:
- Sprint documentation integration
- Architecture documentation creation
- Operational runbook development
- Template and example creation
- Quality assurance processes

Deliverable: Detailed content integration specification with task breakdown

2.3 Validation Workflow Specification (30 minutes)

Specify enhanced validation workflows for the knowledge base:

```
# Create workflow specification
uvx --from git+https://github.com/github/spec-kit.git specify init valida-
tion_workflows --ai copilot
```

Include specifications for:
- Content quality validation
- Link checking and reference validation
- Code example testing
- Documentation completeness checks
- Automated content generation

Deliverable: Comprehensive workflow specification with implementation plan

Exercise 3: Initial Implementation (90 minutes)

Objective

Begin implementing the foundation structure and initial content for the knowledge base.

Tasks

3.1 Directory Structure Creation (30 minutes)

Create the complete directory structure for the integrated knowledge base:

```
cd /home/ubuntu/github_spec_training/HX-Infrastructure-Knowledge-Base

# Create comprehensive directory structure
mkdir -p docs/{adrs,architecture,history/sprints,operations/{run-
books,monitoring,backup-recovery},security,troubleshooting,integration}
mkdir -p templates/{ansible,terraform,cicd,documentation}
mkdir -p examples/{playbooks,modules,configs,scripts}
mkdir -p metrics/{training,project}
mkdir -p tests/{content,workflows}

# Create placeholder files with basic structure
touch docs/adrs/README.md
touch docs/architecture/README.md
touch docs/history/sprints/README.md
touch docs/operations/README.md
touch templates/README.md
touch examples/README.md
touch metrics/README.md
touch tests/README.md
```

Deliverable: Complete directory structure with placeholder files

3.2 First ADR Creation (30 minutes)

Create the first Architecture Decision Record for the training integration:

```
# Create ADR template and first ADR
mkdir -p docs/adrs
```

Create `docs/adrs/ADR-0001-training-integration.md` with:
- **Status:** Proposed
- **Context:** Integration of HX-KB into GitHub Spec Kit training program
- **Decision:** Use progressive content development approach across 5 training days
- **Consequences:** Enhanced practical learning, real project outcomes, measurable training effectiveness

Deliverable: Complete ADR-0001 following standard ADR format

3.3 Initial Documentation Templates (30 minutes)

Create templates for consistent documentation:

```
# Create template directory and files
mkdir -p templates/documentation
```

Create the following templates:
- `templates/documentation/sprint-summary-template.md`
- `templates/documentation/runbook-template.md`
- `templates/documentation/adr-template.md`
- `templates/documentation/architecture-doc-template.md`

Deliverable: Complete set of documentation templates with examples

Exercise 4: Quality Validation (60 minutes)

Objective

Implement quality validation processes and test the initial implementation.

Tasks

4.1 Content Validation Setup (30 minutes)

Create basic content validation processes:

```
# Create validation scripts directory
mkdir -p scripts/validation

# Create basic validation script
cat > scripts/validation/validate-structure.sh << 'EOF'
#!/bin/bash
# Basic structure validation script

echo "Validating HX-KB structure..."

# Check required directories
required_dirs=("docs/adrs" "docs/architecture" "docs/history/sprints" "docs/
operations" "templates" "examples")

for dir in "${required_dirs[@]}"; do
    if [ -d "$dir" ]; then
        echo "✓ $dir exists"
    else
        echo "✗ $dir missing"
        exit 1
    fi
done

# Check required files
required_files=("README.md" "docs/adrs/ADR-0001-training-integration.md")

for file in "${required_files[@]}"; do
    if [ -f "$file" ]; then
        echo "✓ $file exists"
    else
        echo "✗ $file missing"
        exit 1
    fi
done

echo "Structure validation passed!"
EOF

chmod +x scripts/validation/validate-structure.sh
```

Deliverable: Working validation script with comprehensive checks

4.2 Documentation Quality Check (30 minutes)

Validate the quality of created documentation:

```
# Run structure validation
./scripts/validation/validate-structure.sh

# Check markdown formatting
find docs -name "*.md" -exec echo "Checking {}" \; -exec head -5 {} \;

# Validate ADR format
cat docs/adrs/ADR-0001-training-integration.md
```

Review and improve:
- Consistent markdown formatting
- Complete ADR structure

- Clear and actionable content
- Proper cross-references

Deliverable: Quality-validated documentation with improvements implemented

Exercise 5: Integration Testing (45 minutes)

Objective

Test the integration between the knowledge base work and the training program.

Tasks

5.1 Workflow Integration Test (20 minutes)

Test the integration with existing workflows:

```
# Commit all changes
git add .
git status
git commit -m "Day 1: Complete foundation structure and initial content

- Add comprehensive directory structure
- Create ADR-0001 for training integration
- Add documentation templates
- Implement basic validation processes
- Complete initial analysis and specifications"

# Push changes and test workflow
git push origin main
```

Deliverable: Successful commit and push with workflow validation

5.2 Specification Validation (15 minutes)

Validate specifications against actual implementation:

```
# Review specifications vs. implementation
cd hx_kb_integration
# Use AI agent to validate specification against actual work completed
```

Check alignment between:
- Specified outcomes vs. actual deliverables
- Planned tasks vs. completed work
- Success criteria vs. achieved results

Deliverable: Specification validation report with any necessary updates

5.3 Training Integration Check (10 minutes)

Verify integration with training program objectives:

Review Day 1 success criteria:
- [ ] Environment fully validated and working
- [ ] Spec Kit installation verified
- [ ] HX-KB repository analyzed and documented
- [ ] Initial specifications created and validated

- [ ] Implementation plan developed
- [ ] Day 1 deliverables committed to repository

Deliverable: Completed success criteria checklist with evidence

Success Metrics

Quantitative Measures

- **Structure Completeness:** 100% of planned directories created
- **Documentation Coverage:** All required templates and ADRs created
- **Validation Success:** All validation scripts pass
- **Specification Quality:** AI agent validation score 85%+

Qualitative Measures

- **Clarity:** Documentation is clear and actionable
- **Consistency:** Formatting and structure are consistent
- **Completeness:** All exercise objectives met
- **Integration:** Work aligns with training program goals

Troubleshooting Guide

Common Issues

Issue: Git push fails

Symptoms: Permission denied or authentication errors
Solution:

```
# Check git configuration
git config --list | grep user
git config --list | grep remote

# Reconfigure if needed
git config user.name "Your Name"
git config user.email "your.email@example.com"
```

Issue: Spec Kit commands not working

Symptoms: Command not found or installation errors
Solution:

```
# Reinstall Spec Kit
uvx --from git+https://github.com/github/spec-kit.git specify --help

# Check UV installation
uv --version
```

Issue: Directory creation fails

Symptoms: Permission errors or path issues
Solution:

```
# Check current directory and permissions
pwd
ls -la
mkdir -p test_dir && rmdir test_dir  # Test mkdir permissions
```

Next Steps

Upon completion of Day 1 exercises:

1. **Review and Validate:** Ensure all deliverables meet quality standards
2. **Prepare for Day 2:** Review Day 2 objectives and requirements
3. **Document Lessons:** Note any challenges or insights for improvement
4. **Team Coordination:** Share progress and coordinate with other participants

Resources

- GitHub Spec Kit Documentation (https://github.com/github/spec-kit)
- ADR Template (../templates/adr-template.md)
- Markdown Style Guide (../guides/markdown-style-guide.md)
- HX-Infrastructure Integration Plan (../../docs/IntegrationPlan.md)

Continue to Day 2 Exercises (day2-intermediate-exercises.md) →