Shellock: Adaptive Terminal Environment Orchestrator

CS7IS5 Adaptive Applications - Project Outline

Team Members: [Name 1], [Name 2], [Name 3], [Name 4], [Name 5]

---

1. Goal

Shellock is an intelligent terminal assistant that helps developers set up virtual environments and resolve dependency issues. The system reasons over errors, adapts its strategy, and learns from past failures to handle environment setup across npm, pip, cargo, and other package managers.

Shellock automatically adapts to the user's preferred tech stack, project requirements, hardware constraints, and operating system—eliminating hours of Stack Overflow searches and manual debugging.

---

2. Overview

How Shellock Works

Simple User Journey:

1. User types: `shellock setup "npm Next.js + Tailwind"`
2. Shellock shows a JSON configuration (user can edit)
3. User approves → Shellock creates the environment
4. If errors occur → Shellock suggests fixes automatically
5. User approves fix → Shellock retries and succeeds

Key Features

Smart Setup: Understands natural language requests like "I need Python with PyTorch for CUDA"

Error Recovery: When setup fails, Shellock analyzes logs and suggests specific fixes

Learning: Remembers past errors and solutions to prevent future issues

Multi-Environment: Run multiple setups in parallel (e.g., production vs staging)

Cross-Platform: Works across npm, pip, cargo, Docker, and Go

---

3. Proposed Solution

Technology Stack

- **Language**: Python (easy to understand and extend)
- **LLM**: Ollama with Llama 3 (runs locally, free)
- **Interface**: Terminal commands with live progress display
- **Execution**: Isolated containers (safe testing without breaking system)

Data Storage (JSON Files)

File 1: Package Database (`shellock_packages.json`)

- Current versions of npm, pip, cargo packages
- Dependency requirements
- System information (OS, installed tools)
- Updated automatically on every run

File 2: Learning Memory (`shellock_history.json`)

- Previous errors encountered
- Fixes that worked
- User preferences
- Success rates

How It Works

Setting Up an Environment:

Step 1: User Request
"shellock setup npm Next.js + Tailwind + Prisma"

Step 2: Shellock Thinks (Ollama LLM)
Query: "Find compatible versions for Next.js, Tailwind, Prisma from database"
Output: JSON configuration with exact versions

Step 3: User Reviews
Shows JSON → User can edit or approve

Step 4: Shellock Executes
Generates bash script → Runs in isolated container → Shows live logs

Step 5: Success!
Environment ready in ~15 seconds

Fixing Errors:

Step 1: Error Detected
npm install fails with "network timeout"

Step 2: Shellock Analyzes (Ollama LLM)
Query: "Search database for solutions to npm network timeout"
Output: JSON fix with command to use faster registry

Step 3: User Reviews Fix
Shows solution → User approves

Step 4: Shellock Retries
Applies fix → Runs setup again → Success!

Step 5: Learns for Future
Saves fix to history → Next time auto-suggests this solution

Adaptive Intelligence

Shellock adapts in three ways:

1. **User Preferences**: If you always approve setups without editing → Shellock stops asking
2. **Error Patterns**: Same error 3 times → Shellock auto-applies known fix
3. **System Context**: ARM Mac detected → Uses ARM-compatible packages automatically

---

4. Management Approach

Agile Development: 2-week sprints with clear goals

Tools:

- GitHub for code
- Trello for task tracking
- Discord for daily check-ins
- Google Drive for documentation

Process:

- Daily 15-minute team check-ins
- Weekly demos of working features
- Code reviews before merging

---

5. Role Assignment

Team Member	Responsibility	Deliverables
Person 1	Core system setup	CLI interface, JSON schemas, Ollama integration
Person 2	Package managers	npm, pip handlers + process management
Person 3	Error recovery	Log analysis, fix suggestions, retry logic
Person 4	User interface	Live progress display, multi-environment support
Person 5	Testing & delivery	Test suite, documentation, final report + video

---

6. Application Use Cases

Use Case 1: Student Starting New Project

Scenario: Computer science student needs React environment for assignment

Steps:

1. shellock setup "npm React + TypeScript"
2. Shellock shows configuration → Student approves
3. Environment ready in 12 seconds

Benefit: No googling "how to install React", no version conflicts

Use Case 2: Developer Hitting Error

Scenario: Professional developer gets npm timeout error

Steps:

1. npm install fails
2. shellock analyze myproject
3. Shellock suggests: "Use faster registry? [y/n]"
4. Developer approves → Retry succeeds

Benefit: Saves 30+ minutes of Stack Overflow searching

Use Case 3: Team Collaboration

Scenario: Team needs identical environments across 5 developers

Steps:

1. Lead developer: shellock setup "pip PyTorch + CUDA"
2. Shellock generates config.json
3. Team members use same JSON → Identical environments

Benefit: "Works on my machine" problems eliminated

Use Case 4: Multi-Environment Testing

Scenario: Developer testing app in production vs staging setups

Steps:

1. shellock group prod staging
2. shellock prod setup "npm Next.js 15"
3. shellock staging setup "npm Next.js 14"
4. Both run in parallel → Compare results

Benefit: Easy A/B testing without conflicts

---

7. Architecture

System Layers

USER
↓
Terminal Interface (Python + Typer)
↓
Ollama LLM (converts text → JSON)
↓
JSON Validation (checks correctness)
↓
Bash Script Generator (creates setup commands)
↓
Isolated Container (runs safely)

↓
Live Log Display (shows progress)
↓
Error Analyzer (if failure occurs)
↓
JSON Learning (saves for future)

Technology Choices

Component	Technology	Why This Choice
Programming	Python	Easy to learn, great libraries
AI Brain	Ollama Llama 3	Free, runs locally, no internet needed
Commands	Typer	Clean command-line interface
Display	Rich	Beautiful terminal output
Containers	Podman	Safe isolated testing
Sessions	libtmux	Multiple environments at once

All technologies are **free and open-source** (MIT/Apache licenses).

---

8. Time Plan

12-Week Development Schedule

Weeks	Phase	Goals	Team Focus
1-2	Foundation	CLI working, JSON parsing, Ollama connected	Person 1 leads
3-4	Core Features	npm/pip setup working, basic errors handled	Person 2, 3
5-6	Error Recovery	Log analysis, auto-fixes, retry logic	Person 3 leads
7-8	Multi-Environment	Parallel setups, environment comparison	Person 4 leads
9-10	Polish	Beautiful interface, comprehensive testing	Person 4, 5
11-12	Delivery	Final report, 1-minute demo video, submission	Person 5 leads

Sprint Structure

Every 2 Weeks:

- Sprint planning (Monday)
- Daily 15-min check-ins
- Working demo (Friday)
- Team retrospective (Friday)

Milestones

- **Week 2**: `shellock setup "npm react"` works
- **Week 6**: Error recovery functioning
- **Week 10**: All features complete
- **Week 12**: Report + video submitted

---

9. Success Criteria

By Week 12, Shellock is successful if:

✅ Sets up npm, pip, cargo environments from simple commands
✅ Detects and fixes common errors automatically
✅ Learns from past failures to prevent future issues
✅ Works on Mac, Linux, Windows
✅ Has 80%+ test coverage
✅ Complete documentation and demo video
✅ Installable via `pip install shellock`

---

10. Risk Management

Risk	How We Handle It
Ollama service unavailable	Use backup cloud LLM API
Complex errors hard to parse	Start with 10 common patterns, expand gradually
Team member unavailable	Pair programming, shared documentation
Feature creep	Lock features at Week 8, focus on polish
OS compatibility issues	Test on GitHub Actions (Mac/Linux/Windows)

Appendix: Example Session

```
$ shellock setup "npm Next.js + Tailwind"

 Shellock: Analyzing request...

{
"ecosystem": "npm",
"packages": ["next@15.0.1", "tailwindcss@3.4.0"],
"node_version": ">=20"
}

Edit configuration? [y/n]: n

▯ Creating environment...
[████████████████████] 100% Complete (14.2s)

✅ Environment ready!
▯ Location: ./my-nextjs-app
▶ Start: cd my-nextjs-app && npm run dev

$ npm run dev
```

App fails with network error

```
$ shellock analyze myapp

▯ Analyzing logs...

Error: npm ERR! network timeout
Fix: Use faster registry

{
"error": "network_timeout",
"solution": "npm config set registry https://registry.npmmirror.com"
}

Apply fix? [y/n]: y

▯ Retrying with fix...
✅ Success! Environment now working.

▯ Saved to learning history.
```

Document Version: 1.0
Date: February 19, 2026
Total Pages: 5
Status: Ready for submission