20251119-demo-notes

Demo Planning Meeting Notes

Date: November 18, 2025
Participants: Shawn Wallace, Michael Collier
Branch: `demo/coi`
GitHub Issue: #12 - Upgrade Task Repository to Cosmos DB

> **Note:** AI-generated meeting notes. Reviewed and formatted for clarity.

Demo Scenario Overview

Objective: Upgrade the in-memory task processor to use Azure Cosmos DB, demonstrating parallel development of C# code and Bicep infrastructure.

Key Decision: Use existing workshop infrastructure project with legacy in-memory task processor as the demo scenario to showcase both code and infrastructure changes.

Demo Scenario Planning and Task Division

Scenario Selection

- Use existing infrastructure project with legacy `InMemoryTaskRepository`
- Upgrade to Cosmos DB as the backend database
- Chosen for relevance and opportunity to demonstrate both code and infrastructure changes

Branching Strategy

- Created dedicated `demo/coi` branch for all demo work
- Both developers branch off from `demo/coi` to avoid impacting main codebase
- Pull requests will target `demo/coi` (not `main`)

Task Assignment

Shawn (@shawnewallace) - C# Code Track

- Update C# code to support Cosmos DB
- Implement `CosmosDbTaskRepository`
- Update dependency injection and configuration
- Add proper error handling and logging
- Write unit and integration tests

Michael (@mcollier) - Infrastructure Track

- Set up `infra/` directory with Bicep files
- Create `main.bicep` and `cosmos.bicep` modules
- Add parameter files for multi-environment support (dev, staging, prod)
- Provision Cosmos DB account and App Service for API hosting
- Consider adding build pipeline if time permits

Demonstrating Agent and Copilot Customizations

Agent Workflow Demonstration

1. Fetch GitHub issue using agent mode
2. Generate step-by-step implementation plan
3. Create feature branches
4. Implement solution with AI assistance
5. Perform automated code reviews using custom prompts
6. Submit pull requests with Copilot-generated descriptions

Copilot Instructions File

- Repository-level instructions: `.github/copilot-instructions.md`
- Links to additional instruction files:
    - C# coding style standards
    - Commit message conventions (Conventional Commits)
    - Follow-up question handling
- Demonstrate how to customize and link files for project-specific needs

File-Specific Instructions

- Add front matter to instruction files
- Apply to specific file types (e.g., `*.cs`, `*.bicep`)
- Enable Copilot to automatically use correct conventions per file type

Slash Commands and Check Mode

- Use `/check` command for deep code reviews
- Set up as reusable prompts for common tasks
- Improve code quality and consistency

Team Onboarding and Best Practices

- Documented conventions accelerate new developer onboarding
- Ensure consistent code quality across team
- Facilitate rapid adoption of best practices

Model Selection and Multi-Agent Capabilities

Model Switching in VS Code

- Demonstrate switching between AI models (GPT-4.1, Claude Sonnet, etc.)
- Different models yield different results based on task and time of day
- Usage throttling can affect availability

Multi-Agent Parallel Work

- Experiment with multiple agents working simultaneously
- Demonstrate agent sessions in VS Code
- Visualize concurrent work on different tasks

Guidance on Model Selection

- Recommend most cost-effective model that meets requirements
- Show auto-select vs. manual model choice
- Demonstrate how to evaluate results and switch models

Project Status and Team Updates

Sprint and Deployment Timeline

- Final feature sprint with December product launch target
- Increased urgency for code completion and testing
- Focus on shipping production-ready features

Testing and Observability Enhancements

- Adding logging and observability foundations

- Integrating Winston and Application Insights
- Ensuring traceability through the stack
- Gaps remain in API layer (opportunity for demo)

Coordination with QA and PM

- Balance untested items with pending pull requests
- Clear communication with QA team and PM (Varun)
- Align priorities across teams

AI-Assisted Requirements and Testing

AI for Requirements and Testing (Pete's Insights)

- Many requirements are not testable as written
- Leverage AI to improve requirement quality
- Generate better tests from improved requirements
- Pete presenting at upcoming QA conference

Action Items

Demo Preparation - Shawn

- [x] Create `demo/coi` branch
- [x] Create GitHub issue #12 for Cosmos DB upgrade
- [ ] Add front matter to instruction files
- [ ] Install VS Code Insiders and test plan mode
- [ ] Run through dry run of planning process
- [ ] Share summary with Piyush and Steve in group chat

Demo Preparation - Michael

- [ ] Set up `infra/` directory structure
- [ ] Create `cosmos.bicep` and `main.bicep` files
- [ ] Add parameter files for multiple environments
- [ ] Create App Service Bicep module

Shared Tasks

- [ ] Test parallel development workflow
- [ ] Validate pull request process into `demo/coi`
- [ ] Document lessons learned for demo talking points

Demo Highlights to Showcase

✅ **Agent Planning** - Fetch issue, generate plan, assign tasks
✅ **Parallel Development** - Two developers, separate concerns, no conflicts
✅ **Copilot Customizations** - Instructions, file-specific rules, consistency
✅ **Code Review Automation** - `/check` command, automated reviews
✅ **Model Selection** - Switch models, compare results, cost-effectiveness
✅ **Infrastructure as Code** - Bicep best practices, multi-environment support
✅ **Team Collaboration** - PRs, code review, integration workflow

---

References

- **GitHub Issue:** https://github.com/centricconsulting/ai-coding-workshop/issues/12
- **Work Item Tracking:** `./demo-work-item-tracking.md`
- **Copilot Instructions:** `../../.github/copilot-instructions.md`
- **Demo Branch:** `demo/coi` Demonstrating Agent and Copilot Customizations: Michael and Shawn discussed showcasing advanced GitHub Copilot and agent customizations in VS Code, including planning, code review prompts, and file-specific instructions, to highlight best practices and team conventions during the demo. Agent Workflow Demonstration: They outlined a workflow where an agent fetches a GitHub issue, creates a plan, implements the solution, and performs code reviews using custom prompts and agent modes in VS Code. This includes demonstrating planning mode, branch creation, and automated code review before pull requests. Copilot Instructions File: Shawn described the use of a Copilot instructions file that links to additional instruction files for tasks like code style, commit message conventions, and follow-up question handling. They agreed to highlight how these files can be customized and linked for project-specific needs. File-Specific Instructions: They discussed adding front matter to instruction files to apply them to specific file types (e.g., .cs, .bicep), enabling Copilot to automatically use the correct conventions and coding styles for each file type. Slash Commands and Check Mode: Shawn explained the use of slash commands (e.g., /check) to trigger deep code reviews by the agent, and how these can be set up as reusable prompts for common tasks, improving code quality and consistency. Team Onboarding and Best Practices: They emphasized the value of documented conventions and Copilot customizations for onboarding new developers, ensuring consistent code quality, and facilitating rapid team adoption of best practices. Model Selection and Multi-Agent Capabilities: Michael and Shawn explored the use of different AI models (e.g., GPT-4.1, Claude Sonnet, OpenAI Codex) and discussed demonstrating model selection and multi-agent parallel work in VS Code to address common questions about model effectiveness. Model Switching in VS Code: They planned to show how VS Code allows switching between various AI models for coding tasks, noting that different models may yield better results depending on the task and time of day due to usage throttling. Multi-Agent Parallel Work: Shawn described experimenting with

cloud code to enable multiple agents to work on several tasks simultaneously, and they considered demonstrating agent sessions in VS Code to visualize concurrent work. Guidance on Model Selection: They agreed to address the frequent question of which model to use by recommending the most cost-effective model that meets the task requirements, and to demonstrate how to set models to auto-select or manually choose based on results. Project Status and Team Updates: Michael updated Shawn on the current project status, including the final feature sprint, deployment pressures, and the need for improved testing and observability, while mentioning coordination with team members like Ian and the Centric QA team. Sprint and Deployment Timeline: Michael explained that the team was informed at the start of the sprint that it would be the last feature sprint, with a product launch targeted for December, leading to increased urgency for code completion and testing. Testing and Observability Enhancements: Michael described efforts to add logging, observability, and unit test foundations to the project, integrating tools like Winston and App Insights, and ensuring traceability through the stack, though noting gaps in the API layer. Coordination with QA and PM: He highlighted the challenge of balancing untested items and pending pull requests, and the need for clear communication with the QA team and project manager Varun to align priorities. AI-Assisted Requirements and Testing: Shawn shared insights from a colleague, Pete, about using AI to write more testable requirements and generate better tests, and mentioned Pete's upcoming conference presentation on this topic. AI for Requirements and Testing: Shawn relayed Pete's assertion that requirements are often not testable, and proposed leveraging AI to improve both the writing of requirements and the generation of corresponding tests, which Pete plans to present at a QA conference. Follow-up tasks: Demo Preparation and Branch Setup: Create a demo/COI branch in the repository for collaborative work and ensure all demo-related changes are made there to avoid impacting the main project. (Shawn) Demo Preparation and Branch Setup: Create and assign an issue to add a Cosmos DB task repository and related infrastructure to the demo/COI branch for the upcoming demo. (Shawn) Demo Preparation and Branch Setup: Add front matter to the instructions files to facilitate easier customization and demonstrate this during the demo. (Shawn) Demo Preparation and Branch Setup: Tweak the instructions and set up a work item to upgrade the project to use a real database context with Cosmos DB, and plan the process including branch management and planning steps. (Shawn) Demo Preparation and Branch Setup: Set up an infra directory in the project and add a Bicep file (e.g., cosmos.bicep) to provision Cosmos DB and an app service for the API, ensuring support for multiple environments. (Michael) Demo Preparation and Branch Setup: Install VS Code Insiders and check out the plan mode to prepare for the demo. (Shawn) Demo Preparation and Branch Setup: Run through a dry run of the planning process to identify any tweaks needed before the demo. (Shawn) Demo Preparation and Branch Setup: Summarize the meeting notes and ensure all identified to-do items are checked off, then share the summary with the group chat including Piyush and Steve to keep everyone aligned. (Shawn)