Sprint 1 Progress

GitHub PR Status IoT Display

- Course: AI-Driven IoT System Development
- Focus: local Swift watcher + IoT display path

Problem and Goal

Problem

- Too much context switching to check PR/review updates in GitHub
- Important status changes are easy to miss during focused work

Goal

- Build passive, glanceable awareness via IoT output
- Poll GitHub PR/review signals and surface meaningful changes

What I Completed So Far

- Defined project scope and architecture in:
  - `GitHub-Status-IoT-App/README.md`
  - `GitHub-Status-IoT-App/PPP.md`
- Added initial workflow placeholders for status-check experiments:
  - `.github/workflows/check-pr-status.yml`
  - `.github/workflows/workflow-testing`
- Created Swift executable project:
  - `GitHub-Status-IoT-App/GitHubWatcher/Package.swift`

Software Progress (Implemented)

- Swift CLI scaffolded with `swift-argument-parser`
- Async entry command in:
  - `GitHub-Status-IoT-App/GitHubWatcher/Sources/GitHubWatcher/GitHubWatcher.swift`
- Runtime options added:
  - `--wait` (poll interval, minutes)
  - `--watchTimeout` (session limit, hours)
- Core models added:
  - `Credentials`
  - `WatcherManager`
  - typed `GitHubUser` response model

GitHub API Progress

- Implemented authenticated user fetch in:
  - `GitHub-Status-IoT-App/GitHubWatcher/Sources/GitHubWatcher/Endpoints/GitHubGetters.swift`
- Current call:
  - `GET https://api.github.com/users/{username}`
- Decoding now uses a typed model (`GitHubUser`) instead of raw JSON
- Verified end-to-end path:
  - credentials -> request -> decode -> watcher manager state update

Learning With AI Progress

Topic 1: `gh api` in a background-thread app (Software)

- Established architecture direction: local background Swift process while actively working
- Built first end-to-end API call and CLI scaffold

Topic 2: External screen display (Hardware)

- Goal and output strategy defined
- Implementation pending after software polling loop is stable

Current Gaps and Next Milestones

Not done yet

- PR and review status endpoints
- Poll -> compare -> notify loop with persisted state
- External screen integration and display rendering

Next milestones

1. Implement PR/review fetch + state snapshot comparison
2. Add stable background run behavior and logging
3. Integrate first external screen output for change alerts