Training Curricula

Expert-Driven AI-Assisted Development

Teaching Marines to Build Software with AI

Document Version: 3.0
Effective Date: 26 January 2026
Classification: UNCLASSIFIED // Distribution Unlimited

Prepared by:
SSgt Jesse C. Morgan
Marine Corps Detachment, Presidio of Monterey
Defense Language Institute Foreign Language Center

The only skill that matters:

Can you explain what you want to a very smart, very literal assistant?

Contents

1. Training Overview

1.1 Three Courses, One Goal

These three courses teach Marines how to use AI to build software. Not theory—practice.

Course	Duration	Outcome
AI-Fluent Development Orientation	2 hours	Write effective prompts, build a working prototype
Platform Training	4 hours	Build 3 complete tools on Power Platform
Advanced Workshop	4 hours	Complex builds, debugging, teaching others

1.2 Prerequisites

Training 1: None. Come with a problem you want to solve.
Training 2: Complete Training 1. Have M365 account with Power Platform access.
Training 3: Have built at least one working tool.

1.3 Required Access (Verify Before Training)

Before attending training, participants must verify they have:

Requirement	How to Verify	If You Don't Have It
M365 GCC Account	Can log into outlook.office365.us	Contact unit S-6/help desk
Power Apps access	Can log into make.powerapps.com	Request license from S-6
SharePoint access	Can access your unit SharePoint site	Request from SharePoint admin
AI tool access	See below	See below

1.3.1 AI Tool for Training

The prompts in this training require conversational AI that can generate code. Options:

- **M365 Copilot in Power Apps** (preferred)—built into Power Apps if your unit has Copilot licenses
- **Azure OpenAI**—request via MSS Data Portal (`mss.data.mil`)
- **Standalone AI** (Claude, ChatGPT, Gemini)—usable with synthetic data only for learning

No AI access? Contact your S-6 to request M365 Copilot, or coordinate with the Program Coordinator. Training can proceed with instructor demonstrations, but hands-on practice requires AI access.

Instructor responsibility: Verify all participants have required access at least 48 hours before training. Coordinate with S-6 to resolve access issues before class.

1.3.2 Equipment for Training

Participants must bring to hands-on sessions:

- Government laptop/workstation with CAC reader
- CAC/PIV card
- Access to network (NIPR for GCC environments)

Browser: Microsoft Edge or Chrome recommended for Power Platform. Some features may not work in Firefox.

2. Instructor Certification

2.1 Who Can Teach

Course	Instructor Requirements
Training 1 (Orientation)	Completed all three training courses AND deployed at least one tool
Training 2 (Platform)	Same as Training 1, plus demonstrated proficiency with Power Platform
Training 3 (Advanced)	Same as Training 2, plus served as QA reviewer for at least 2 tools
Supervisor Orientation	Any qualified Training 1 instructor

2.2 Instructor Certification Process

1. Complete all three training courses as a student
2. Build and deploy at least one tool under this SOP
3. Shadow an existing instructor for one delivery of the course
4. Deliver the course with an existing instructor present (co-teach)
5. Receive certification from Program Coordinator

2.3 Train-the-Trainer Guidance

For new units without certified instructors:

- Request support from an established unit's instructor
- Use video recordings of course delivery (if available)
- Program Coordinator may certify based on equivalent experience

Succession planning: Each unit should maintain at least 2 certified instructors. When an instructor PCSs, ensure replacement is certified before departure.

2.4 Maintaining Certification

Instructors remain certified as long as they:

- Deliver at least one course per year
- Stay current with platform and AI tool changes
- Maintain active involvement in the developer community

3. Training 1: AI-Fluent Development Orientation

Required — 2 Hours — All Developers

Purpose: Teach people how to talk to AI to build things.

3.1 Agenda

Time	Topic
0:00–0:10	The Mindset Shift (why this works)
0:10–0:40	How to Talk to AI (prompt basics)
0:40–0:55	Live Build #1: Simple tool from scratch
0:55–1:05	Break
1:05–1:35	The Iteration Loop (when AI gets it wrong)
1:35–1:55	Live Build #2: Fix, extend, improve
1:55–2:00	Your assignment: Pick your problem

3.2 Module 1: The Mindset Shift (10 min)

You're not learning to code. You're learning to describe what you need clearly.

Bad prompt:
"Make me an app for tracking stuff"

Good prompt:
"I need a simple app where tutors can log their sessions. Each session should capture: tutor name, student name, date, duration, and notes. I want to see a list of all sessions and be able to add new ones."

The clearer you describe it, the better the output.

3.3 Module 2: How to Talk to AI (30 min)

3.3.1 The Prompt Formula

Context + Task + Constraints + Format = Good Prompt

Example:
"I'm a language instructor *(context)*. I need a way to track which students have completed their tutoring sessions this week *(task)*. It needs to work in Power Apps connected to a SharePoint list *(constraints)*. Show me the list structure first, then the app screens *(format)*."

3.3.2 Practice Exercise

Participants write prompts for these scenarios:

1. Tracking equipment checkout
2. Scheduling conference room bookings
3. Collecting feedback after training events

Instructor reviews and improves prompts together.

3.4 Module 3: Live Build #1 (15 min)

Instructor builds in real-time, talking through the process.

Problem: "I want to track who's submitted their leave requests"

Step 1 — Data structure:

> "I need a SharePoint list to track leave requests. Columns: Marine name, type of leave, start date, end date, status (pending/approved/denied), approver notes. Give me the column setup."

Step 2 — Build the app:

> "Now build me a Power App canvas app that connects to this list. I need: a form to submit new requests, a gallery showing my requests, and a way for approvers to see all pending requests."

Step 3: Test it, find what's wrong, ask AI to fix it.

> **Key point:** This is messy. AI won't get it perfect. That's normal.

3.5 Module 4: The Iteration Loop (30 min)

When AI gives you garbage (it will):

1. **Be specific about what's wrong**
   - Bad: "This doesn't work"
   - Good: "The submit button isn't saving to SharePoint. I get this error: [paste error]"
2. **Ask for explanations**
   - "Explain what this formula does so I can verify it's right"
   - "Why did you use a Gallery instead of a Form here?"
3. **Break it into smaller pieces**
   - If a big request fails, ask for one piece at a time
   - "Just show me how to connect to SharePoint first"
4. **Give it your failed attempt**
   - "I tried this: [paste code]. It gives this error: [paste error]. Fix it."

3.5.1 Practice Exercise

Instructor gives participants a broken Power Apps formula. They practice writing prompts to debug it with AI.

3.6 Module 5: Live Build #2 (20 min)

Continue the leave tracker:

- Add a feature (email notification when approved)
- Fix something that doesn't work
- Make it look better

> "The gallery is showing all requests. I only want to show requests where Status = 'Pending'. How do I filter it?"
>
> "Now I want to send an email when someone approves a request. This needs Power Automate, right? Walk me through setting up that flow."

3.7 Wrap-Up (5 min)

Your assignment:

1. Think of ONE real problem in your section
2. Write a prompt describing what you need
3. Try building it (even if you fail, that's learning)
4. Bring questions to Platform Training

> **Compliance note:** If your tool will handle PII or health data, check with Privacy Officer first. For most tools, you're good to go.

4. Training 2: Platform Training (Power Platform)

Recommended — 4 Hours — Hands-On Building

Purpose: Build three working tools, learning the platform as you go.

4.1 Agenda

Time	Topic
0:00–0:15	Platform orientation (where things live)
0:15–1:15	Build #1: Equipment tracker (lists + app)
1:15–1:30	Break
1:30–2:30	Build #2: Request workflow (add automation)
2:30–2:45	Break
2:45–3:45	Build #3: Dashboard (add reporting)
3:45–4:00	Troubleshooting patterns + Q&A

4.2 Platform Orientation (15 min)

Where things live:

- **SharePoint** = your database (lists store data)
- **Power Apps** = your user interface (screens, forms, buttons)
- **Power Automate** = your automation (when X happens, do Y)
- **Power BI** = your dashboards (charts, reports)

How to access:

- make.powerapps.com (for apps)
- make.powerautomate.com (for flows)
- Your SharePoint site (for lists)

That's all the orientation you need. Let's build.

4.3 Build #1: Equipment Tracker (60 min)

What we're building: An app where people can check out equipment and you can see what's available.

4.3.1 Phase 1 — Data (15 min)

> "I need a SharePoint list for equipment checkout. Columns: equipment name, serial number, checked out by (person), checkout date, return date, status (available/checked out). Also a second list for the equipment inventory: name, serial, category, location."

Build the lists together.

4.3.2 Phase 2 — App (30 min)

> "Build a Power Apps canvas app for equipment checkout. Screen 1: show available equipment in a gallery, tap to check out. Screen 2: form to confirm checkout (auto-fills equipment, asks for expected return date). Screen 3: show my current checkouts with a return button."

Build it, test it, fix what breaks.

4.3.3 Phase 3 — Polish (15 min)

> "The gallery looks ugly. Make it show equipment name in bold, serial number smaller below, and a colored icon showing status (green=available, red=checked out)."

4.4 Build #2: Request Workflow (60 min)

What we're building: Automation for a request process—when someone submits, notify the approver; when approved, notify the requester.

4.4.1 Phase 1 — The trigger (20 min)

> "Create a Power Automate flow that triggers when a new item is added to my Leave Requests SharePoint list. It should send an email to the person in the 'Approver' column with the request details."

4.4.2 Phase 2 — The approval action (20 min)

> "Now create another flow: when the Status column changes to 'Approved', send an email to the person who submitted the request telling them it's approved. Include the dates."

4.4.3 Phase 3 — Error handling (20 min)

> "What happens if the Approver field is empty? Add a condition to check, and if empty, send to a default approver email instead."

> **Key learning:** Automation is just "when X happens, do Y." AI can write the logic; you just describe the business rules.

4.5 Build #3: Leadership Dashboard (60 min)

What we're building: A Power BI dashboard showing request status, equipment utilization, or whatever data you've been collecting.

4.5.1 Phase 1 — Connect data (15 min)

> "How do I connect Power BI to my SharePoint list called 'Leave Requests'?"

4.5.2 Phase 2 — Basic visuals (25 min)

> "Create these visuals:
>
> 1. Card showing total requests this month
> 2. Pie chart showing requests by status (pending/approved/denied)
> 3. Bar chart showing requests by person
> 4. Table showing all pending requests with requester name and dates"

4.5.3 Phase 3 — Make it useful (20 min)

> "Add a date filter so leadership can look at different time periods. Also add a filter by department if that column exists."

4.6 Troubleshooting Patterns (15 min)

When things break, try these prompts:

"Delegation warning" error:

> "I'm getting a delegation warning on this Filter formula: [paste]. How do I fix it or work around it?"

App won't save to SharePoint:

> "My Patch formula isn't saving. Here's the formula: [paste]. Here's the error: [paste]. The list columns are: [list them]."

Flow failing:

> "My Power Automate flow fails at this step: [description]. The error says: [paste]. What's wrong?"

5. Training 3: Advanced Development Workshop

Optional — 4 Hours — Complex Builds + Teaching Others

Purpose: Build something complex, debug hard problems, prepare to help others.

5.1 Agenda

Time	Topic
0:00–0:30	Architecting bigger solutions
0:30–1:30	Build: Multi-screen app with roles
1:30–1:45	Break
1:45–2:45	Advanced automation patterns
2:45–3:00	Break
3:00–3:45	Debugging hard problems (group workshop)
3:45–4:00	Teaching others: how to answer questions

5.2 Architecting Bigger Solutions (30 min)

Sometimes you need multiple apps:

- A user-facing app (submit requests)
- An admin app (manage settings, see everything)
- A dashboard (leadership view)

> "I'm building a tutoring management system. Users are: students (book sessions), tutors (log sessions, see their schedule), chiefs (see all data, run reports), admins (manage tutor assignments). Help me think through: should this be one app or multiple? What data do I need? What are the main screens for each user type?"

AI helps you architect, not just build.

5.3 Build: Multi-Screen App with Roles (60 min)

What we're building: A task management app where users see their tasks, supervisors assign tasks, and admins manage categories.

5.3.1 Phase 1 — Role detection (15 min)

> "In Power Apps, how do I check if the current user is in a SharePoint group called 'Supervisors'? I want to show different screens based on their role."

5.3.2 Phase 2 — Conditional navigation (20 min)

> "On app start, check the user's role. If they're a supervisor, go to SupervisorHome screen. If they're an admin, go to AdminHome. Otherwise go to UserHome."

5.3.3 Phase 3 — Role-specific screens (25 min)

Build each home screen with appropriate functionality:

- **User:** My tasks, mark complete
- **Supervisor:** Team tasks, assign new, progress chart
- **Admin:** Manage categories, manage users

5.4 Advanced Automation Patterns (60 min)

5.4.1 Pattern 1: Scheduled flows (20 min)

> "Create a flow that runs every Monday at 8am, checks for overdue tasks, and sends a summary email to each person with their overdue items."

5.4.2 Pattern 2: Approval workflows (20 min)

> "Create a flow with a real approval action. When a purchase request is submitted, send an approval to the supervisor. If approved, update status and notify requester. If rejected, notify with the rejection reason."

5.4.3 Pattern 3: Error handling (20 min)

> "My flow sometimes fails because the data is missing. Add try-catch style error handling: if the main action fails, log the error to a SharePoint list and send me an alert email, but don't stop the whole flow."

5.5 Debugging Hard Problems (45 min)

Group workshop format: Participants bring their own stuck problems. Group works through them together.

For complex formula errors:

> "Break down this formula step by step and tell me what each part does. Then identify where it might fail: [paste formula]"

For performance issues:

> "This app takes 10 seconds to load. Here's what happens on OnStart: [paste]. Here's my main gallery's Items property: [paste]. What's making it slow and how do I fix it?"

For "it works sometimes" bugs:

> "This flow works sometimes but fails randomly. Here are three run histories—one success, two failures. [paste details]. What's different between them?"

5.6 Teaching Others (15 min)

When someone asks you for help:

1. **Don't take over.** Guide them to find the answer.
   - "What have you tried?"
   - "What error are you seeing?"
   - "Let's ask AI together—how would you describe this problem?"
2. **Teach the debugging process, not just the fix.**
   - "Here's how I'd approach this..."
   - "The error message is telling us X, which usually means Y"
3. **Point to resources, don't be the resource.**
   - "This is in the SOP under Phase 3"
   - "Try asking AI: [suggest prompt]"

> Your job isn't to build everyone's tools. It's to help them build their own.

6. Supervisor Orientation

Required for Supervisors — 30 Minutes

Purpose: Ensure supervisors can effectively approve, support, and track developer efforts.

6.1 Agenda

Time	Topic
0:00–0:10	What EDD is (and isn't)
0:10–0:15	Time allocation expectations
0:15–0:20	Evaluating Problem Definition Documents
0:20–0:25	Your role in the workflow
0:25–0:30	Where to find help

6.2 What EDD Is (and Isn't)

EDD IS:

- Marines building permanent tools using AI assistance
- Creating institutional capability that survives PCS
- Investing time upfront to save time permanently

EDD IS NOT:

- Marines using AI for daily tasks
- A way to avoid proper IT support
- Quick fixes that create technical debt

6.3 Time Allocation Expectations

Phase	First Tool	Subsequent Tools
Training	10 hours	0 hours (already trained)
Development	20–40 hours	10–20 hours
Documentation	20–40 hours	8–16 hours
Total	**50–90 hours**	**18–36 hours**

First tools take longer. This is an investment. Subsequent tools are faster, and the capability compounds across the unit.

6.4 Evaluating Problem Definitions

When approving a Problem Definition Document, verify:

- The problem is real (not hypothetical)
- The scope is achievable (start small)
- Success criteria are measurable
- Data considerations are addressed (PII/PHI)
- Time estimate is realistic

Red flags:

- "Build a system that does everything"
- No clear users identified
- Requires classified data handling
- Duplicates existing enterprise solutions

6.5 Your Role in the Workflow

1. **Approve Problem Definitions** — Ensure feasibility and alignment
2. **Allocate Time** — Development needs dedicated hours, not spare moments
3. **Track Progress** — Weekly check-ins during development
4. **Serve as QA Reviewer** (if no qualified peer available) — Use the checklist
5. **Ensure Turnover** — Don't let tools die at PCS

6.6 Where to Find Help

- **SOP:** Complete methodology and compliance requirements
- **Training Curricula:** This document—course content and instructor guidance
- **Program Coordinator:** For questions about standards, registry, metrics
- **Privacy/Cybersecurity Office:** For compliance questions

7. Quick Reference Card

7.1 The Prompt Formula

Context → **Task** → **Constraints** → **Format**

7.2 When AI Gets It Wrong

1. Be specific about what's wrong
2. Paste the error message
3. Break it into smaller requests
4. Ask it to explain its thinking

7.3 Universal Debugging Prompt

> "I'm trying to [goal]. I did [action]. Expected [result]. Got [actual]. Error: [message]. Fix it."

7.4 The Build Loop

Describe → **Generate** → **Test** → **Fix** → **Repeat**

A. Instructor Notes

A.1 For Training 1 (Orientation)

- Keep it inspiring, not bureaucratic
- Emphasize: "You can do this"
- Live demo should be messy/real, not polished
- End with concrete next step: "Pick your problem"

A.2 For Training 2 (Platform)

- Hands-on > lecture (70/30 ratio)
- Have backup exercises if people finish early
- Expect varied skill levels; pair stronger with newer
- AI prompts are training wheels, not crutches

A.3 For Training 3 (Advanced)

- This is peer learning; instructor facilitates
- Real problems from participants are best content
- Documentation workshop is where most people struggle—spend time here
- Peer review builds community of practice

A.4 For Supervisor Orientation

- Keep it brief—supervisors are busy
- Focus on their specific role, not the whole methodology
- Address time allocation concerns directly
- Provide the SOP for reference, don't try to cover everything