ADMI 2026 Workshop

Friday, March 27, 2026
Morehouse College

Vibe Coding from Classroom to Industry

AI Utilization in Higher Education Settings for Faculty/Staff

Workshop Focus

- Why AI matters now
- Hands-on lesson planning
- Industry coding workflows
- Guardrails for higher ed

Dr. Dewayne A. Dixon
Faculty Fellow | AUC DSI
Visiting Assistant Professor, Morehouse College

Prof. Cory Brooks
ex SWE (Big Tech, Wall Street)
CodePath Instructor, Morehouse College

Vibe Coding – INDUSTRY

AI-Assisted development with human verification – ACADEMIA

Workshop roadmap

Today's journey from theory to practice

Total duration: 110–120 minutes

1	**Why AI Matters** Data-driven urgency, adoption stats, and the trust paradox	**15 min**
2	**Lesson Plan Creation** Hands-on activity: build a complete unit with AI tools	**45 min**
☕	**Break & Setup** Stretch, hydrate, and check VS Code + GitHub access	**05 min**
3	**Industry Vibe Coding** VS Code, Copilot, agents, and prompt engineering	**50 min**
🏁	**Wrap-Up & Resources** Key takeaways, Q&A, and the resource pack	**Closing**

Audience check-in

Quick poll: where are you today with AI?

Raise your hand for the option that best describes your current usage.

1. Never used

I haven't tried AI tools yet, or I have not found the time.

2. Dabbled a bit

I have used ChatGPT once or twice, but it is not part of my routine.

3. Use weekly

I use AI for a few targeted tasks such as drafting, outlining, or feedback.

4. Use daily

AI is integrated into my workflow and changes how I work.

What You’re Probably Thinking Right Now

Before we dive into adoption statistics, let’s acknowledge the elephant in the room.

Many of you are thinking:

1. “This is just going to make cheating easier!”

2. “Students will stop thinking and just copy-paste.”

3. “AI hallucinations!!! How do I stop it from misleading students?”

4. “I don’t have time to police AI use with 60+ students!”

These concerns aren’t obstacles. They’re exactly why **intentional pedagogical design is required**.
Today’s goal: Show you concrete classroom strategies that make AI use strengthen critical thinking instead of replacing it.

What This Workshop is *NOT*

Not a mandate to allow AI in your course.

Not pressure to redesign your syllabus this semester.

Noe a lowering of academic standards.

Not outsourcing thinking to machines.

Today is about protecting rigor while acknowledging reality.

Why This Session Exists

This is **NOT** about replacing instructors and **NOT** about chasing trends.

This session *is* about:

1. Preserving academic rigor in an AI-present world

2. Keeping grading defensible and transparent.

3. Designing assignments students cannot "fake".

4. Ensuring human judgment stays in control.

Whether we teach it or not, AI is already here!!!
Our choice is supervised use… or unsupervised use.

Industry reality: AI adoption is universal

Three survey signals that frame the urgency for educators

84%

Developers use or plan to use AI tools

Stack Overflow 2025

51%

Professional developers use AI daily

Stack Overflow 2025

91%

Adoption in active developer cohorts

DX sample (135k+ devs)

What this means for higher ed

AI is no longer a niche skill. Faculty are preparing students for a work environment where AI-assisted workflows are already normal. The question is not whether students will encounter these tools — it is whether they will learn to use them critically.

Enterprise signals you cannot ignore

The market has moved from experimentation to standardization

~90%

Fortune 500 adoption

Microsoft FY26 Q1 Earnings

20M+

Total GitHub Copilot users

Mid-2025

4.7M

Paid subscribers

Jan 2026
(75% YoY)

~26%

Projected CAGR

AI code assistant market

Reading the signal

Enterprise adoption, user scale, and paid growth all suggest AI-assisted coding is becoming infrastructure — not a novelty.

AI is writing a meaningful share of code

Students need practice collaborating with AI-generated output

22%

Merged code that is AI-authored

DX Analytics

26.9%

All production code is AI-authored

DX Research, Q1 2026 (4.2 million developers analyzed)

Implication for teaching

Students must learn to read, test, critique, and improve AI-produced code — not just write code from scratch.

The Faculty Time Reality

Common fear: “This sounds like more work than I can manage.”

What typically happens instead:

1. Fewer grading disputes (criteria are explicit).

2. Fewer plagiarism investigations.

3. Short oral checks replace hours of forensic review.

4. Better explainations implies fewer clarification emails.

Key shift:
Time moves *earlier* (design) instead of *later* (policing).

Productivity gains come with nuance

The upside is real, but it requires judgment and review

3.6h/week

Average time saved

DX Analytics 2025

+60%

PR volume increase

Daily AI users vs light users

10–30%

Overall productivity lift

Across varied tasks

Daily users vs. light users

Teams that use AI more consistently often ship more pull requests, but gains depend on task type, developer skill, and verification habits.

AI Output is Convincing and Often Wrong

Common failure modes we see in student work:

1. Incorrect complexity explanations

2. Silent edge-case failures.

3. Hallucinated APIs or citations.

4. Confident but false reasoning.

Design response:
Assignments must *force discovery of errors*. If students **cannot** explain what failed **they do not pass.**

The trust paradox

Adoption is high, but confidence remains limited

1.7×

More issues in unverified AI code

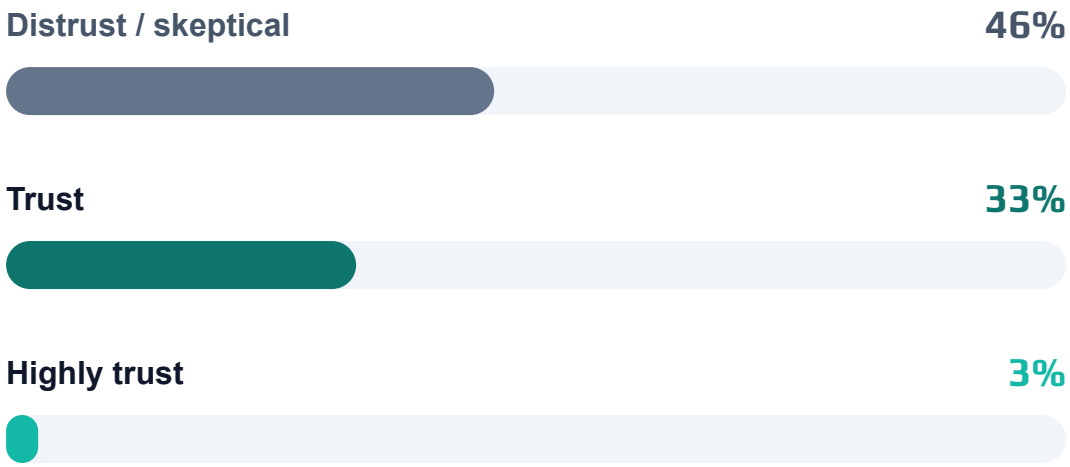
CodeRabbit Dec 2025

75%

Developers still turn to humans when they doubt AI

Stack Overflow 2025

Trust levels in AI accuracy



Critical takeaway

AI literacy does not mean blind trust. It means better prompting, stronger review, and routine verification.

Education gap → student outcomes

Institutional competitiveness increasingly depends on AI literacy

70%

CS teachers already include AI topics

2025 teacher landscape

Hiring reality

Employers increasingly expect AI literacy across the full development lifecycle.

Funding pressures

Graduate outcomes are being watched more closely, AI skill-readiness is part of that story.

Institutional implication

Curriculum adaptation affects enrollment appeal, graduate preparedness, and institutional relevance.

Critical takeaway

Integrating AI topics into the curriculum is a strategic necessity that directly impacts student employability and institutional standing.

Classroom-Tested Strategies That Actually Work

Five concerns that usually come up first

		Core Strategy	Scalable Implementation
1	**Academic integrity**	Grade process documentation, not just output	Require AI conversation logs + **1-page reflection on what failed**. Weight: 35%
2	**Critical Thinking Loss**	Make AI the starting point, not the finish line.	Assignment: "*Use AI to generate quicksort. Then test on 5 edge cases, identify 2 AI explanation errors, refactor for a given constraint.*" HIgher-order analysis required.
3	**Hallucination Risk**	Teach verification as a graded skill	Give students AI-generated code containing 3 **subtle bugs**. Grade **debugging logic**, not speed.
4	**Grading Scalability**	5-minute rotating oral spot-checks	Randomly select 20% of students per assignment. Ask 3 questions: Walk me through this. Why this approach? What breaks if I change X?
5	**Equity & access**	Institutional baseline + transparent alternatives	Course use free ChatGPT as baseline. Premium tools allowed under same rules. Paper-based alternative with adjusted rubric available.

Key insight: When students must **test**, **critique**, and **improve AI output**, they engage in **deeper analysis** than when writing code from scratch.

Educator use cases

Practical ways faculty can use AI right away

📋 Lesson planning

Generate unit outlines, objectives, pacing guides

📝 Grading & rubrics

Draft clear criteria and speed up first-pass feedback

📊 Exams & homework

Produce diverse questions, variants, coding challenges

✍️ Grant writing

Draft proposals, summaries, alternative abstracts

📚 Study plans

Create personalized practice paths or remedial sets

🔜 Next step

We will explore several of these workflows in the hands-on segment

Minimal Adoption Paths (pick one)

You do **NOT** need to redesign your course.

Start with just ONE option:

01 Add an explanation + verification section to one assignment.

02 Require AI disclosure + 1-page reflection (no grading overhaul).

03 Use AI only for instructor prep this semester.

All there protect rigor and are opt-in. None require new tools or approvals.

This Does Not Replace Expertise

AI Handles drafting.
Faculty expertise becomes more visible and not less.

Your role expands:

- Designing constraints
- Evaluating reasoning
- Interpreting edge cases
- Judging trade-offs
- Defending academic standards

AI cannot explain *why* something is wrong.
That remains human work.

SEGMENT 2

Hands-on lesson planning

AI is a supervised assistant, not an authority. Let's build a lesson plan that uses AI productively — with human judgment at every step.

We are moving from insight to practice with a simpler, live-demo-friendly layout.

Build a lesson plan with AI assistance

45-minute collaborative workshop structure

45 minutes

Format

Guided Build

Section-by-section guided build: prompt, generate, critique, refine in real time.

Collaborative Work

Work in clusters by course type (intro CS, data structures, AI/ML, software engineering).

Deliverables

- Learning objectives
- Lecture outline
- Practice problem set
- Grading rubric
- **AI-use & verification checklist**

Workflow

1. **Prompt**
2. **Generate**
3. **Inspect**
4. **Revise**
5. **Verify**

Use your preferred AI tool (e.g., GenSpark, ChatGPT, Claude) — the workflow matters more than the brand.

Risk zones: Which AI uses are safe in teaching?

A faculty guide to low-risk, medium-risk, and high-risk AI use

Green — Low Risk

- Brainstorming
- Outlining
- Rubric drafting
- Test-question drafting
- Example generation
- Debugging hints
- Code explanation

Yellow — Use with Caution

- Code generation
- Lesson materials with factual claims
- Custom GPTs / personalized agents
- Auto-generated references
- Feedback drafts (require instructor review)

Red — High Risk Without Safeguards

- Final grading without human review
- Unverified citations or URLs
- Uploading sensitive student data to public tools
- Merging agent work without review
- Letting AI replace student explanation of code

Privacy & data risks

Protect what you share with AI tools

Prefer institution-managed tools

They have negotiated data agreements and governance.

Custom GPTs have different boundaries

Check what data persists, is shared, or leaves the platform.

External apps/APIs route data outside your control

Avoid entering protected or sensitive information.

Never upload sensitive data to public tools

Student records (FERPA), unpublished research, private course materials, institutional data.

Rule of thumb: When in doubt, assume it's public.

The faculty role: require thinking, not ban tools

Four non-negotiable expectations for students using AI in CS courses

Explain

Students must explain code logic in their own words, not just paste output.

Test

Students must run, test, and verify all AI-generated output before submission.

Justify

Students must justify design decisions and trade-offs — not just accept defaults.

Demonstrate

Students must show understanding without AI in some assessments.

AI handles the drafting. Humans handle the thinking. That boundary is your job to enforce.

Assignment Redesign Example: From Vulnerable to AI-Integrated

Before (AI-Vulnerable)

Assignment: *Implement binary search in Python. Submit working code.*

Result: AI solves in 10 seconds. Students copy-paste. Little learning.

After (AI-Resistant)

- **Part 1**: Use AI to generate binary search implementation.
- **Part 2**: Test on **5 provided edge cases** and document what fails.
- **Part 3**: Compare AI's complexity explanation to textbook and identify **2 inaccuracies**.
- **Part 4**:: Refactor for readability **with justification**.
- **Part 5**: **5-minute oral defense** (no AI access)

Grade Weights

- **30%** code correctness
- **40%** verification & testing process
- **30%** explanation quality

Notice

AI becomes the drafting tool. Students must understand code deeply enough to **critique, test, and modify it** which is a higher-order skill than writing from scratch.

Step 1. Learning objectives

5 min

Start with pedagogy. Define clear, measurable outcomes before generating content.

Copy & adapt prompt

```
I teach [COURSE NAME].

Propose 3–5 measurable learning objectives for a
unit on [TOPIC].
Align them to Bloom’s Taxonomy and use specific
action verbs.
Avoid vague verbs like “understand” or “Know”.

Include a brief justification for why each
objective is accessible in a CS course.
```

Refinement tips

Check measurability: avoid vague verbs like "understand" or "know."

Align to assessment: if you cannot imagine an exam question, make the objective more concrete.

Mix Bloom's levels: include lower-order recall and higher-order creation/evaluation.

Prompting is iterative: ask for a more rigorous or more introductory version.

Faculty checkpoint

Mark which objective is too vague and rewrite it yourself. If you can't imagine an exam question for it, make it more concrete.

Students must be able to explain what each objective means — not just copy it.

Treat all output as a first draft. You are expected to modify, correct, and contextualize before classroom use.

Step 2. Lecture outline

Turn objectives into a paced, engaging 50-minute lesson flow.

7 min

Copy & adapt prompt

```
Using these objectives, draft a 50-minute lecture
outline.

Include timeboxes for each section.

Add concrete examples and engagement checks to
verify understanding.

Flag any parts of the outline that may be
unrealistic for a 50-minute class and explain why.
```

Refinement tips

Adjust pacing: AI often underestimates time for demos and discussion.

Personalize examples: swap generic placeholders for your students' context.

Verify engagement: use polls, mini tasks, or checks — not only 'Any questions?'

Try: 'Act as a skeptical student and show me the boring parts.'

Faculty checkpoint

Identify one part that is unrealistic, too generic, or poorly paced — and fix it before using.

AI often underestimates discussion time and overestimates content coverage.

Treat all output as a first draft. You are expected to modify, correct, and contextualize before classroom use.

Step 3. Practice problems

5 min

Generate varied problems with clear inputs, outputs, and edge cases.

Copy & adapt prompt

```
Generate 5 practice problems at varied difficulty
for [CONCEPT].

Include at least 2 with edge cases and 1 with a
real-world context.
Format each problem with a clear input/output
example.

For each problem, include one common student
mistake or misconception.
```

Refinement tips

Calibrate difficulty: aim for a smooth ramp from trivial to challenging.

Add constraints: ask for time complexity limits or banned shortcuts when useful.

Make it real: change 'sort numbers' into a context like transactions, logs, or sensors.

Generate extra volume: then keep the best ones.

Faculty checkpoint

Check for hidden assumptions, wrong outputs, weak edge cases, or misleading wording. Solve each problem yourself before assigning.

Students must show their reasoning — not just the final answer.

Treat all output as a first draft. You are expected to modify, correct, and contextualize before classroom use.

Step 4. Assessment rubric

5 min

Define success criteria before students submit the work.

Copy & adapt prompt

```
Create a rubric for [ASSIGNMENT TYPE] that assesses:
- Correctness
- Style & readability
- Testing coverage
- Explanation & comments
Use 4 performance levels with specific descriptors.

Include one criterion that penalizes plausible but incorrect explanations.
```

Refinement tips

Weight criteria explicitly if you want stronger emphasis on correctness or testing.

Add integrity checks such as attribution and citation requirements.

Request a paste-ready format such as Markdown table or CSV.

Bonus: ask for a student-facing checklist version too.

Faculty checkpoint

Add one criterion that rewards explanation, verification, or debugging — not just the final answer.

This makes the rubric enforce understanding, not just output.

Treat all output as a first draft. You are expected to modify, correct, and contextualize before classroom use.

Step 5. Supplemental materials

Generate the supporting assets that usually consume prep time.

5 min

Copy & adapt prompt

```
1) Draft slide bullets for the outline above, with
speaker notes.

2) Produce commented code examples for [CONCEPT].

3) List 5 commonly cited references or
documentation sources that instructors should
manually verify before sharing.
Do not invent URLs. If unsure, say so.
```

Refinement tips

Reduce slide density: ask for 3–4 bullets per slide instead of full paragraphs.

Specify code style: include language version and style guide if it matters.

Ask the model to 'explain this like a beginner' for student-friendly comments.

Generate extra volume, then curate the best results.

Faculty checkpoint — verify everything

Verify every citation, URL, and factual claim before sharing with students. AI frequently hallucinates references that look real but don't exist.

Cross-check all links. If a URL doesn't resolve, drop it.

Treat all output as a first draft. You are expected to modify, correct, and contextualize before classroom use.

Group activity debrief

Share outputs, compare prompts, and reflect on what verification caught.

01

Present outputs

Select 2–3 groups to share one component. Focus question: Where did AI help most? Where did you have to override it?

02

Discuss verification habits

What phrasing produced the best result? What verification step caught the biggest error? Where did the AI struggle?

03

Capture best practices

Compile the strongest prompts AND the best verification habits into a shared template faculty can reuse.

Use → Verify → Explain

A faculty-ready framework for responsible AI integration

USE AI may help with brainstorming, drafting, debugging hints, and first-pass materials.

VERIFY All AI-assisted work must be checked with tests, source verification, and human review.

EXPLAIN Students must be able to explain the logic, decisions, and corrections in their own words.

Sample syllabus statement — adapt for your course

"You may use AI for brainstorming, outlining, debugging hints, and draft generation. You may not submit AI output unchanged. You must disclose meaningful AI use and be able to explain, verify, and defend your final work."

Break & Setup

Take a breather

Stretch, hydrate, and get ready for the coding segment

Break time

05:00

Resume shortly for Segment 3.

Setup checklist

Install VS Code (must be locally installed)

Sign in to GitHub.com or create a free account

Install the Copilot extension within VS Code, and sign into your GitHub account.

Make sure you are using the free subscription tier. Students can register for an upgraded free student account.

From there you are ready live hands-on demo!

Segment 3

Industry vibe coding

Bridging classroom theory and modern software engineering practice with live demos.

We are moving from insight to practice with building a personal artifact for you to keep after today.

Term of the day: vibe coding

A useful definition plus the classroom implication

Definition

The process of generating software primarily through iterative LLM prompting and review rather than purely manual line-by-line coding.

Industry reality

Professional developers increasingly work in a hybrid mode: AI for boilerplate and acceleration, humans for judgment, logic, and review.

Student risk

The risk is not simply 'AI replacing students.' A new risk moving forward is students with strong AI literacy outpacing those without it.

Software development lifecycle (industry)

AI is not replacing the process — it is embedded across it. Here are some examples of where AI is used throughout the SDLC.

1 Requirements

- Clarify ambiguous specs
- Generate user stories

2 Architecture

- Propose design patterns
- Analyze trade-offs

3 Implementation

- Scaffold boilerplate
- Suggest functions
- Refactor syntax

4 Documentation

- Summarize modules
- Generate comments

5 Testing

- Generate unit tests
- Create edge cases

6 Maintenance

- Triage bug reports
- Explain legacy code

Environment setup check

Making sure everyone can follow the live coding segment

VS Code installed

Confirm that Visual Studio Code launches correctly on your machine.

GitHub account ready

Sign in to GitHub and make sure you can access repositories you need.

Copilot extension

Install the GitHub Copilot extension within VS Code. You will not need a paid subscription for today.

If you are unable to install VS Code today, please complete the non-coding portions individually. You will be able to implement your project from the browser towards the end.

Github Copilot: What exactly is it?

The AI pair programmer reshaping daily development workflows

What it does

- Context-aware suggestions from your files
- Generate unit tests, documentation, boilerplate
- Chat-based explanation and refactoring
- Define agent instructions and use custom agents

20M+ **Users**
Mid-2025

4.7M **Paid subs**
Jan 2026

~90% **Fortune 100 use**
Original deck

Critical caveat

AI output is a prediction, not a fact. It can hallucinate APIs, introduce security flaws, or suggest deprecated patterns.

Best practices

Always review line by line, pair with tests, and always start with a good prompt and instruction set.

Prompt Engineering: Getting more out of AI

Specificity is the difference between toy output and production-ready output

Weak prompt

```
"Make a website grading assistant for my
assignments."
```

Likely outcome:

Misaligned grading strategies

Multiple iterations required

Harsh and random rules without guidance.

Strong prompt

```
 "I am an instructor working to create a website
to be used as a grading tool. Please review the
below design before asking a series of at least 10
follow up questions..."

 Requirements...

 UI UX...

 Rubric..."
```

Lesson: Define requirements before asking for work (Similar to manager → employee relationship).

Pro tip: Always specify language, return type, constraints, and verification method.

Prompt creation workflow (5min)

Think before we prompt. Let’s put as much detail into our ideal product. We will use this in a few slides.

Workshop Goal:

Implement a grading assistance tool (website) using Copilot through VS Code. Type these responses out into a blank txt file or notepad for now.

1. **Consider UI/UX questions:**
   a. Color palette?
   b. Animations and layout?
   c. Text and labels.
   d. Background images (need to be downloaded and saved locally).

2. **What kind of insights would be helpful for you to see?**
   a. Commonly missed concepts
   b. Generate specified study questions per student
   c. Dashboards or charts showing trends
   d. Calendar view over time

3. **IMPORTANT**: You can ask Copilot to create sample submissions of your class type for today’s demo.

1 **Design your website mentally**
Be explicit about requirements.

2 **Draft Notes**
Write rough specification ideas step by step.

3 **Prompt**
Translate notes into a solid prompt.

4 **Review & Iterate**
Have the LLM ask you follow-up questions.

Copilot Agent: Roles and Instructions

Users should define agent “roles” to fine tune their behavior. This is accomplished by providing agent instructions.

The task:

Open the Copilot Panel in VS Code (top right of the window)

Click the settings icon for the Copilot Panel

Click Generate Instructions Drop down → Instructions (Workspace)

Paste your full prompt at the bottom of this file

Example Role: “You are a veteran web designer who focuses on data analytics and designing technical products for educators.”

1

1. Identify the “real world” persona

Think about who you would want to build this in real life.

2

2. Preventing “free reign”

Be specific about things that the agent absolutely can never do.

3

3. Lasting guardrails

Instruction files are read before every user prompt is executed. For example: “confirm that any photos found online are copyright friendly”

4

Paste into Copilot instructions file

Now you have a local file that you can edit at anytime for immediate behavioral changes that last.

Hands-on challenge

Now that Copilot is setup and ready to go, provide any required files into the workspace and try starting a chat in the Copilot panel to begin implementation.

The assignment:

15:00 remaining

Using methods that we discussed today, try to work with Copilot to implement your website.

1. Play around with the different models available (Claude, Sonnet, GPT, etc.)
2. Always Start in Plan mode — research and outline before writing a line of code
3. Iteratively work through the (Design → Prompt → Implement) cycle.
4.

1 Requirement 1

The website launches and renders successfully.

2 Requirement 2

You were able to add one unique element that we did not discuss together.

3 Requirement 3

You are proud of what you made today! There is so much to learn in AI and the tools keep evolving every day.

★ Pro Tip

In the Copilot chat, you can scroll up to 'Restore Checkpoint' at any point.

GitHub agents preview

Beyond VS Code: Organizing large scale product development using Copilot.

Example workflow

1. Initialize a new Repository (enable ReadMe option)
2. Open an issue with a clear request (prompt tips)
3. Assign it to Copilot
4. Agent implements changes
5. Human reviews the branch or accepting their changes

High-value use cases

Issue triage

Boilerplate & scaffolding

Test generation for existing code

The human loop

The human loop: By default, GitHub will not allow Copilot to merge changes without your approval.

YOU ARE STILL IN CONTROL

Faculty Control Is Non-Negotiable

\

You choose if, where, **and** how **AI appears.**

You may opt out of any student use.

You may require non-AI assessments.

You may revise policy at any time.

Authority stays with you, standards stay intact, and this is guidance (NOT a mandate).

Debrief & reflection

Move from passive observation to active pedagogical strategy

01

What surprised you?

Reflect on capabilities, limitations, and moments when the tools behaved differently than expected.

02

Where did AI help or hinder?

Identify moments of acceleration versus friction or confusion in today's workflow.

03

What would you try next week?

Name the first small classroom change: one policy tweak, one assignment change, or one lab activity.

CONCLUSION

Key takeaways

Three things to remember from today

1. AI literacy is table stakes

Students graduating without AI collaboration skills face a real disadvantage in modern software work.

Action: integrate it now, not later.

2. Use + verify

Do not only teach prompting. Teach the verification habits that make AI-generated output trustworthy.

Action: make verification a graded skill.

3. Start small

You do not need a full curriculum overhaul. One lesson plan, one policy update, or one Copilot lab is a strong beginning.

Action: try the lesson plan built today.

Resources & next steps

Keep the vibe going with tools, contacts, and cited source labels from the original deck

Contacts

Dr. Dewayne Dixon
ddixon@aucenter.edu

Prof. Cory Brooks
cory.brooks@morehouse.edu

Essential tools

GenSpark — persistent teaching workflows

Base44 — semester-stable AI tools

GitHub Copilot — industry-standard assistant

EdStem — LMS with integrity features

VS Code — Industry standard IDE

Original source labels retained

Stack Overflow Developer Survey 2025

DX Analytics Report (Q4 2025)

CodeRabbit Analysis (Dec 2025)

Microsoft FY26 Q2 / Gartner 2025