Applied Agentic AI

Proposed course syllabus · graduate and undergraduate versions

John E. Benhart IV

Course description

The interdisciplinary nature of this course provides students the opportunity to bring a real-world problem from their field and apply AI using methods learned in class to scaffold research and performance.

In this course, students learn to work as the "human-in-the-loop," first by learning about agentic AI and how it functions. Then, the process of verifying, iterative improvement, and documentation are instructed to improve the performance of AI tools/agents. This all culminates in a term project that, using the practices from this course, allows for students to delve deeper into specific topics than they previously could in a semester-long course.

Bring a real problem from your field, and learn to solve it with a team of AI agents that you direct. Students come from engineering and from business, and each one brings a problem from industry or research.

You start with work that almost every project needs: statistical analysis, project planning and scheduling. You do each one with agents, and you learn where they help and where they go wrong. Next you scope a problem with an AI-simulated client, the way you would with a real one. A simulated client lets every student practice working with a client, even students who have not worked in industry. Then you build your own project. You set the goal, assign the work and check every result; the agents do the building.

You stay the human in the loop. AI should scaffold your work, not replace your judgment. Agents are fast and fluent, and a fluent wrong answer looks just like a right one. So the course teaches you to check what an agent produces, to write the standards its work must meet as you iterate, and to plan work it can carry out. These skills carry over to any tool and any field.

This course comes from my own experience researching AI, teaching with AI, and building a toolkit of AI workflows that I improved each time I tested it.

Target Audience

- **Graduate version:** master's and early doctoral students. Prerequisite: one course in quantitative methods, statistics or analytics.
- **Undergraduate version:** juniors and seniors. Prerequisite: one course in statistics or quantitative methods.
- No programming course or prior AI experience is required. Students learn to read and test the short scripts the agent drafts.
- Students use any AI assistant or coding agent, such as Claude or Claude Code, ChatGPT or Codex, Google Gemini, or GitHub Copilot. No outcome depends on one product.

Learning outcomes

Students who complete the course will be able to:

1. Direct AI agents on everyday project work, such as statistical analysis, planning and scheduling, and scope a problem with a client.
2. Explain how an AI agent produces its output, and why it can get things wrong.
3. Verify results from AI through specific techniques and behaviors taught in the course.
4. Build and iterate on standards agents must meet (rules and regulations, with finished work as exemplars) for specific projects.
5. Improve an agent's work step by step, and measure whether each change helped.
6. Plan work for a team of agents, knowing what each agent contributes.
7. Interact with real-world problems and use AI as a scaffold to make operational decisions and optimize about a goal.

Modules

1. **Everyday project work with agents**
   - Statistical analysis: the agent runs it, and you check it
   - Basic project management: a plan, milestones and a status report the agent drafts

- Scheduling: the agent builds a schedule, and you test it against the constraints
- Small project 1: scope a problem with an AI-simulated client and agree a problem statement
- Small project 2: one of the three tasks above, done with agents, on data you choose
- The same three tasks return as worked examples in later modules
- *Verification test:* a flawed statistic or schedule, found on paper

2. How an agent reads, and why it gets things wrong

- An overview of AI as a stochastic parrot
- The basics of AI context, memory, pointers and retrieval-augmented generation (RAG)
- What an agent reads and what it processes
- Context rot
- Correcting a deliberately flawed analysis
- End-of-session review: what the agent got wrong, and why
- *Verification test:* the missing context behind a wrong answer

3. Checking an agent's work before you use it (verification)

- Why the work ahead is less about repeating routine tasks and more about checking AI results
- Checking results before they are used or signed
- A verdict on every result the agent returns
- The agent runs the check; a person signs the verdict
- PACE (Parallel Agent Consensus Engine): two agents work the same problem independently, each reviewed by its own coach; a cross-reviewer then compares the two and reports where they agree and where they differ
- *Verification test:* planted errors, the check that finds each, and a verdict

4. Iterative improvement and documentation

- The improvement loop: experiment, check, record, revise
- Work that is not finished adds to the rules and regulations (Module 5); finished work becomes an exemplar
- Documenting each iteration: what changed, why, and what it fixed
- The correction log as a working record that later sessions read
- Measuring whether a change improved efficiency and performance
- *Verification test:* whether a documented change actually improved the result

5. Writing the standards an agent must meet (rules, regulations and exemplars)

- Why standards keep fast, fluent agents in check and their work consistent
- Rules: scripts the work must pass, trusted only after they fail
- Regulations: guidance for judgment, given at every session start
- Exemplars: finished work to follow
- The learning loop: until the work is finished, each correction becomes a rule, regulation or exception; finished work becomes an exemplar
- *Verification test:* a rule against a planted case

6. Working from the real source (ground truth)

- A knowledge base the agent works from, so you know where each fact came from
- The source itself on file, not the agent's account of it
- Extracted text checked as real text
- A working link before any citation
- *Verification test:* citations whose source does not say what is claimed

7. Planning work an agent can carry out (planning guidance and oversight)

- Why a plan for an agent must state the domain knowledge and assumptions a person would leave unsaid
- The charge, and an inventory of what already exists
- Socratic questioning, one question at a time
- A plan, a critic, and a person's approval before building
- Build, then verify against the plan
- *Verification test:* where built work departs from its plan

8. Many agents, one question: the Council of Agents (agentic performance amplification)

- Agents that build, audit and critique work, ideas and models
- The Council of Agents: distinct roles, one question, a chair's report
- Different roles give different answers, even from one model
- Module 5 rules as gates a person must approve
- *Verification test:* what a council's agreement does and does not show

Term project

The course is built around the term project. Each student finds a real-world problem or research topic to address. By providing them the opportunity to delve into a passion project in detail, students are able to gain experience while simultaneously learning about pressure points in the workplace/research when working with applied problems.

Each module adds one piece to the project: (Module 1) a problem statement agreed with the client, (2) a correction log, (3) a verdict on every result the agent returns, (4) a documented record of each iteration, (5) the standards file, built from each iteration that is not finished, (6) a source audit, (7) the approved plan, and (8) a map of the gates and who may override each one.

The project is built on its rules: each time the agent gets something wrong, the student records the fix in the standards file as a rule, a regulation or a one-off exception, and adds the finished work as an exemplar; by the end the agent's work must pass every rule, each one shown to fail on a planted case. The project closes with a three-part defense, in which students (1) find the defects planted in a copy of their own work, on paper and without AI; (2) run one of their own rules and show it refusing a planted case; and (3) explain why a gate sits where it does.

How the versions differ

Both versions cover the same eight modules and use the same grading. They differ in three places.

Part of the course	Graduate	Undergraduate
Red team (finding planted defects)	Finds the defects planted in a classmate's project; without a client, also builds a flawed project, with an answer key, for a classmate	Finds the defects planted in a classmate's project
Council of Agents	Designs the agents' roles	Uses roles the instructor supplies, run in class
Source audit (checking citations)	Audits five citations, one of them found by the student	Audits three citations the instructor supplies

Grading

AI is allowed, and expected, on all work outside the room. The term project is the largest part of the grade. Forty percent comes from a midterm and a final, taken in class, on paper, without AI. The midterm covers Modules 1 to 5; the final covers all eight. Both versions use the same weights.

Component	Weight	Take-home, AI allowed	In class, AI-free, on paper
Examinations	40%	–	40% (midterm 15%, final 25%)
Module work	15%	5% small projects and completion	10% module checks
Term project and its rules	45%	20% graded submission	25% defense
Total	**100%**	**25%**	**75%**

Client data

Each real client signs a one-page scope letter that says what the student may see and share. An AI-simulated client needs no letter. Projects use public, synthetic or client-approved data, and none collects data about people. A client is named only with written consent, never on a public page.