ISSN: 2473-4901
v12 n6798

From Prompt to Prototype: Teaching Vibe Coding to Professional MBA Students

Mark Frydenberg
mfrydenberg@bentley.edu

David J. Yates
dyates@bentley.edu

Department of Computer Information Systems
Bentley University
Waltham, MA 02452, USA

Abstract

Vibe coding enables users to guide artificial intelligence (AI) development agents through natural-language prompts rather than writing code directly. This paper describes the design, implementation, and evaluation of a vibe coding project completed by six students in a Professional Master of Business Administration (MBA) technology lab. Using Google Antigravity, students defined business or professional problems, translated them into functional requirements, guided an AI development agent through iterative prototype development, tested and revised the resulting applications, documented errors, and reflected on lessons learned. The project followed a learn–apply–reflect cycle grounded in experiential learning, reflective practice, and project-based learning. Exploratory evidence from student prototypes, process documentation, presentations, reflections, and an end-of-lab survey suggests that students developed practical skill and judgment while gaining a more realistic understanding of AI-assisted development. The implementation also exposed prototype fragility, scope-control challenges, incomplete data integration, and the continuing need for human oversight. Taken together, these findings illustrate how carefully scaffolded vibe coding can support business and information systems education by positioning students as managers of AI-assisted development.

Keywords: Vibe coding, AI-assisted software development, generative AI, project-based learning, information systems education, prompt engineering.

From Prompt to Prototype: Teaching Vibe Coding to Professional MBA Students

Mark Frydenberg and David J. Yates

1. INTRODUCTION

Generative artificial intelligence (AI) tools are reshaping how students, professionals, and organizations think about software development. Building applications has often been framed as an activity for computer science students, who first had to learn programming syntax, development environments, debugging practices, and deployment workflows before creating even modest working prototypes. Recent AI-assisted development tools change that starting point. In vibe coding, users describe a desired application, feature, interface, or workflow in natural language, and an AI system generates and revises code in response. This shifts some of the user's work from writing code directly to defining intent, specifying requirements, evaluating outputs, and guiding iterative refinement (Bamil, 2025; Crowson & Celi, 2025).

This shift is especially relevant for business students. Many business students will not become professional software developers, but they will increasingly be expected to evaluate, guide, and govern AI-assisted development work. They may need to assess whether an AI-generated dashboard, workflow tool, decision aid, or prototype is useful, reliable, explainable, or ready for further development. At the same time, generative AI use among college students has expanded rapidly, creating a need for targeted educational strategies that help students use AI tools effectively and responsibly (Frydenberg et al., 2026). Vibe coding therefore creates both an opportunity and a challenge for business education: it lowers the barrier to creating software prototypes, but it also requires new forms of judgment about prompting, requirements, testing, data quality, user experience, and technical fragility.

Prior work on AI-assisted programming suggests that these tools create distinctive user experiences, dependencies, and challenges, including difficulties evaluating generated code, understanding system behavior, and managing the interaction between human intent and machine-generated output (Sarkar et al., 2022). These issues are particularly important for novices because AI code-generation tools can support learning while also creating risks of overreliance, incomplete understanding, and misplaced confidence unless they are embedded in careful instructional design (Kazemitabaar et al., 2023). For business students with limited technical experience, the educational goal, therefore, is not simply to "make an app," but to learn how to frame a problem, guide an AI development agent, test what is produced, identify limitations, and reflect on the relationship between apparent functionality and actual usefulness.

This paper describes the design, implementation, and evaluation of a vibe coding project in a Professional Master of Business Administration (MBA) technology lab titled "Mastering Artificial Intelligence for Professional Excellence" at a business-focused university in New England. Appendix A provides the course description and brief syllabus. Students used Google Antigravity to create proof-of-concept prototypes addressing business or professional problems of their choosing. The assignment positioned students as managers of AI-assisted development rather than as traditional programmers. They defined users and value propositions, translated business needs into functional requirements, guided an AI development agent, tested and revised the resulting prototypes, documented errors, and reflected on lessons learned.

The paper makes three contributions to information systems education. First, it frames vibe coding as an activity for business and information systems education rather than only as a programming activity. Second, it presents a learn–apply–reflect instructional design that draws on experiential learning, reflective practice, and project-based learning (Barron et al., 1998; Kolb, 1984; Schön, 1983). Third, it reports exploratory evidence from six student projects, process documentation, presentation and reflection artifacts, and end-of-lab survey responses, illustrating both the promise and the limitations of using vibe coding with Professional MBA students.

2. VIBE CODING OVERVIEW

Vibe coding is an emerging form of AI-assisted software development in which a user describes an intended application, feature, interface, or workflow in natural language, and an AI system generates, revises, and often helps test the resulting code. Rather than writing every line of code directly, the user works at the level of intent, requirements, constraints, feedback, and iteration. For business students, this shift is important because application prototyping becomes accessible to learners who understand business problems, data needs, and user workflows, even if they do not have extensive programming backgrounds (Bamil, 2025; Crowson & Celi, 2025).

Vibe coding also builds on a longer trajectory of end-user development and low-code/no-code approaches intended to enable people without extensive programming expertise to create or configure software at higher levels of abstraction (Barricelli et al., 2019; Bock & Frank, 2021). Vibe coding extends this trajectory by using natural-language interaction with generative and agentic AI systems as a primary mechanism for specifying and revising software. This places particular importance on the user's ability to frame problems, specify requirements, evaluate generated outputs, and direct iterative improvement.

2.1 Vibe Coding in the Business Curriculum

For Professional MBA students, the educational opportunity was not simply to learn coding in the traditional sense, but to learn how AI-assisted development can be specified, guided, tested, and revised. This aligns with the managerial role many business students are likely to seek or occupy — evaluating, guiding, and governing AI-assisted knowledge work. This need is consistent with evidence that generative AI awareness and use have risen rapidly among college students, creating a need for targeted educational strategies that help learners use AI tools effectively and responsibly (Frydenberg et al., 2026).

Much of the emerging research on AI-assisted coding has examined professional programmers or students learning programming. Sarkar et al. (2022), for example, examine programming with AI as a distinct form of software-development work, while Kazemitabaar et al. (2023) study AI code generators in introductory programming education. Our context differs in an important respect: the students were business professionals rather than aspiring software developers, and the instructional goal was not primarily to teach programming. Instead, students were positioned as managers of AI-assisted development who needed to translate business needs into requirements, guide an AI development agent, evaluate what it produced, and decide when a prototype was useful, fragile, incomplete, or in need of revision.

The most obvious affordance of vibe coding is rapid prototyping. A learner can describe a dashboard, workflow tool, data-entry interface, or decision-support application and receive a working or partially working prototype much faster than would usually be possible through conventional software development. This enables students to explore business-process experimentation. For example, what would a better procurement dashboard look like? How might a stock-monitoring tool signal when review is needed? What would a student-service application need to include to be useful? Vibe coding also makes user-interface generation more tangible. Students can see how changes in their prompts affect layout, navigation, visual hierarchy, and user experience. In this sense, vibe coding can help students connect business analysis, design thinking, and technical implementation in a single activity.

At the same time, vibe coding should not be presented as a substitute for software expertise. Research on programming with AI suggests that AI-assisted programming creates new experiences and challenges around understanding generated code, debugging, evaluating correctness, and managing the interaction between human intent and machine-generated output (Sarkar et al., 2022). These challenges are especially important for novices because AI code-generation tools can support novice learners while also raising concerns about overreliance, incomplete understanding, and misplaced confidence (Kazemitabaar et al., 2023). In our context, these limitations appeared as prototype fragility, setup friction, mock or incomplete data integrations, broken features after new prompts, and uncertainty about whether an application was truly functional or merely appeared polished.

In our Professional MBA technology lab, students used Google Antigravity as the primary vibe coding

environment. Google describes Antigravity as an agentic development platform, and practitioner accounts emphasize its agent-first design, browser interaction, planning artifacts, and ability to support asynchronous or semi-autonomous coding workflows (David, 2025; Google Antigravity Team, 2025; Preston, 2025). This tool choice was pedagogically useful because it made the role of the AI development agent concrete. Our students did not ask a chatbot for code snippets. They interacted with an environment that could propose a technical stack, create files, run commands, modify an interface, and attempt to verify its own work. This made it possible to frame the student role as one of providing managerial guidance — defining the goal, approving or redirecting the plan, testing outputs, identifying gaps, and requesting revisions.

2.2 Conceptual Prerequisites

We paired a hands-on, in-class Antigravity activity with instruction on the architecture of web-based applications. Although vibe coding lowers the barrier to software development by translating natural-language functional descriptions into code, students still need a working vocabulary for the systems on which their prototypes depend. Accordingly, we introduced core concepts at a level suitable for newcomers, entrepreneurs, and non-coders completing their first vibe-coding project.

System Architecture. The lesson began with a high-level overview of the parts of a data-driven, web-based application, distinguishing the front end — the part of an application the user sees and interacts with, including buttons, forms, layouts, text, and graphics — from the back end, which operates behind the scenes to implement application logic, databases, and services and often runs on a remote web server.

The lesson next reviewed front-end web technologies. Students were introduced to HTML (hypertext markup language) as the means of defining structural elements such as headings, paragraphs, images, and forms; CSS (cascading style sheets) as a way to control fonts, colors, layout, and other aspects of appearance; and JavaScript as a means of implementing interactive behavior. The purpose was not to teach students to program these technologies directly, but to give them enough vocabulary to interpret an AI-generated technical plan and distinguish visible interface elements from the functionality supporting them.

Application Programming Interfaces (APIs). Students who wished to incorporate live data from external sources — such as stock prices from Yahoo Finance or weather information and news headlines from external feeds — needed to understand how applications obtain and exchange data. We explained that an API is a structured way for one program to request services or data from another and receive information in a form that can be integrated into an application. Appendix C lists potential external data sources, curated by Google Gemini and verified by the authors, for students to explore.

Deployment. The deployment lesson introduced different contexts in which applications can run, beginning with running a prototype locally on the student's own computer for development and testing and extending to deployment on an Internet-accessible server or publication through an application store for broader use. This distinction helped students recognize that a prototype functioning on a local machine is different from an application prepared for wider distribution and use.

Taken together, these conceptual prerequisites gave students a basis for evaluating the prototypes they later created: application architecture helped them distinguish visible interface behavior from underlying functionality, APIs helped them assess whether external data were actually integrated, and deployment helped them distinguish a locally functioning proof of concept from an application ready for broader use. Students then applied these concepts to a meaningful business or professional problem, tested and revised their prototypes, and reflected on what the process revealed about AI-assisted development. These steps helped shape the learn–apply–reflect cycle described in Section 3 and the project structure described in Section 4. Section 5 reports the resulting student projects, process documentation, presentation and reflection artifacts, and end-of-lab survey responses.

3. THE LEARN–APPLY–REFLECT LEARNING CYCLE

Building on this overview, we describe the instructional design used in our Professional MBA technology lab as a learn–apply–reflect learning cycle. Students first encountered core ideas about generative AI

ISSN: 2473-4901
v12 n6798

and vibe coding (Bamil, 2025; Crowson & Celi, 2025), then applied those ideas through guided and independent AI-assisted development activities and finally reflected on their own development experiences and peers' project demonstrations. Conceptually, this cycle draws on experiential learning and reflective practice. Kolb (1984) emphasizes movement among conceptualization, experimentation, experience, and reflection; Schön (1983) distinguishes reflection during and after action; and Boud, Keogh, and Walker (1985) and Gibbs (1988) emphasize that experience becomes learning through deliberate analysis and subsequent action. Together, these perspectives support the pragmatic learn–apply–reflect pattern used in this technology lab. See Figure 1 below.

3.1 Learn Phase

The "learn" phase began before students attempted their own projects. Because the Professional MBA technology lab was a short, half-credit offering, the instructional challenge was to make AI-assisted application development approachable without reducing it to a demonstration. Students were introduced to vibe coding, natural-language prompting, and the business implications of this approach. In a guided in-class activity using Google Antigravity (David, 2025; Preston, 2025), students built an executive dashboard designed to display stock information and business-news content (see Appendix B).

The exercise connected the conceptual prerequisites introduced in Section 2.2 to judgments students would later make about their own prototypes. Understanding application architecture helped students distinguish a polished interface from underlying functionality; understanding APIs helped them assess whether external data were actually integrated; and understanding deployment helped distinguish a locally functioning proof of concept from an application ready for broader use. During the guided exercise, students tested navigation, data loading, server execution, and the difference between static mockups and dynamic functionality, creating a shared vocabulary for the larger project.

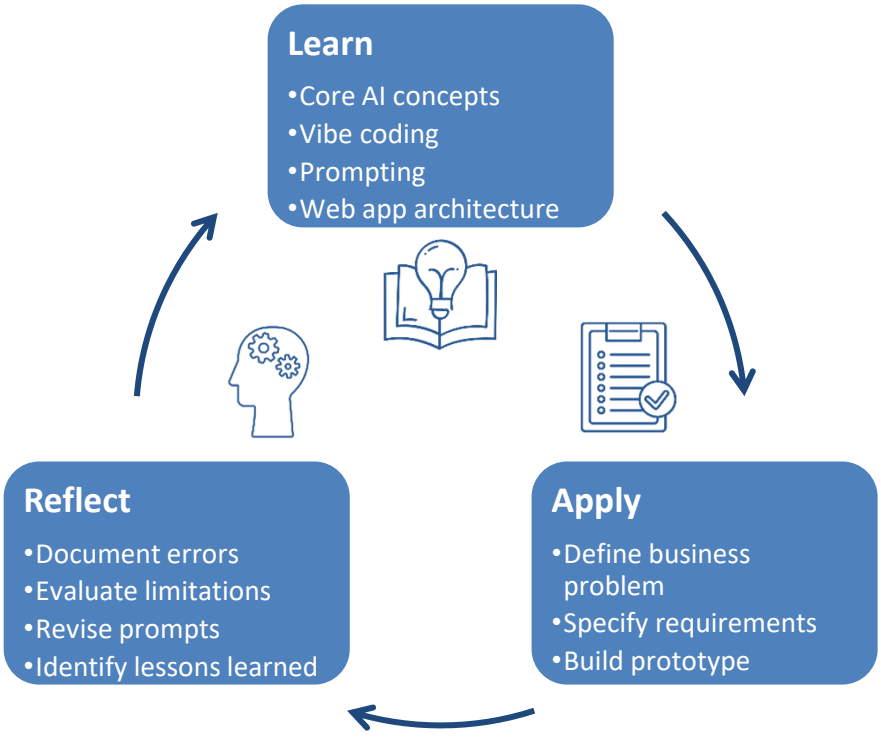

Figure 1. Learn–Apply–Reflect Learning Cycle (Source: Authors, Icons: NounProject).

3.2 Apply Phase

The "apply" phase asked students to move from following an instructor-designed example to defining and managing their own vibe-coded prototype. Each student identified a business or professional

problem, translated it into functional requirements, asked AI to recommend a technical strategy, guided an AI development agent through iterative development, validated data and user-interface functionality, and documented decisions and errors. This structure turned prompt engineering into a managerial and analytical task: students had to define users and value propositions and determine data needs, interface requirements, external data sources, and acceptance criteria. This emphasis is consistent with Ambrose et al.'s (2010) view that effective learning requires practice organized around clear goals, feedback, and opportunities to monitor one's own understanding.

Project-based learning provided the instructional mechanism for making this learning cycle visible. Rather than asking students only to discuss generative AI or complete a bounded tutorial, the project required them to produce artifacts that could be examined: a business problem statement, functional requirements, a working or partially working prototype, testing notes, an error log, a demonstration, peer feedback, and a reflection on lessons learned. This is consistent with project-based learning research emphasizing sustained inquiry, meaningful artifacts, scaffolding, and support for both "doing" and "understanding" (Barron et al., 1998; Blumenfeld et al., 1991; Kokotsaki et al., 2016). In our context, the project structure transformed the learn–apply–reflect cycle from an abstract pedagogical pattern into an observable sequence of student decisions, prototypes, breakdowns, revisions, and interpretations.

We evaluated student success using project artifacts, process documentation, presentation and reflection artifacts, and end-of-lab survey responses. For the project evidence, we examined whether students could define a meaningful problem, guide an AI development agent through development iterations, test and revise the resulting prototype, and identify and explain its limitations. We did not judge success by production readiness. These criteria organize the project summaries and outcome evidence reported in Section 5.

3.3 Reflect Phase

The "reflect" phase occurred through project and error logs, management logs, executive-summary slides, presentations, documented lessons learned, and peer evaluation. In our implementation, this phase was especially important because students' artifacts showed that vibe coding can create an illusion of progress: polished screens can appear before the underlying logic, integrations, or persistence are fully functional. Reflection therefore helped students recognize that AI-assisted development still requires significant human judgment.

Rather than treating the final prototype as the sole evidence of learning, we used reflection artifacts to examine how students interpreted breakdowns, revised prompts, evaluated limitations, and distinguished proof-of-concept functionality from production readiness. This emphasis is consistent with Moon's (1999) view of reflection as a way of moving from experience toward more calibrated professional understanding. Section 5 returns to these reflection artifacts alongside the student projects, process documentation, and end-of-lab survey responses.

3.4 Learn–Apply–Reflect as an Instructional Design

As an instructional design, the learning cycle had three principal strengths. First, it connected learning to procurement, investing, sustainability compliance, student services, personal performance, and fitness analytics. Second, it positioned students as managers of AI-assisted development who had to specify requirements, assess AI-generated plans, test results, and decide when to revise or redirect the development process. Third, it generated visible artifacts through which learning could be examined. This process is consistent with Argyris and Schön's (1978) view of learning as a theory-of-action process: students acted on assumptions, encountered mismatches between intentions and outcomes, and revised their strategies in response.

Overall, the learn–apply–reflect cycle was a strong fit for a Professional MBA technology lab because it framed AI-assisted development as a managerial, analytical, and reflective practice. In this implementation, students used AI tools to create meaningful prototypes while developing judgment about when AI-generated applications were useful, fragile, or incomplete. These outcomes do not suggest that AI-assisted development eliminates the need for software expertise. Rather, they illustrate how carefully scaffolded experience can help Professional MBA students develop informed judgment

about AI-assisted development. The next section describes the project structure that operationalized this cycle.

4. VIBE CODING PROJECT DESCRIPTION

4.1 Project Overview and Learning Goals

The vibe coding project was designed as an applied project-based learning experience in which students used Google Antigravity, an AI-assisted development environment, to create a proof-of-concept prototype addressing a relevant business or professional problem, often by combining data from multiple sources. Students were expected to define the problem and intended users, translate business needs into functional requirements, guide an AI development agent through iterative prompt-based development, validate the accuracy and functionality of the resulting prototype, and document their decision-making process. The assignment therefore emphasized the managerial, analytical, and reflective dimensions of AI-assisted software development rather than traditional coding alone.

The preparatory instruction and four project parts operationalized the learn–apply–reflect cycle described in Section 3. The conceptual instruction and guided Antigravity tutorial supported the learn phase; product strategy, development, testing, and refinement constituted the apply phase; and project and error logs, management logs, demonstrations, peer critique, executive-summary slides, and documented lessons learned supported the reflect phase.

4.2 Project Tools and Structure

Our Professional MBA technology lab included four 90-minute, in-person sessions: two during the first weekend and two approximately one month later. Students received Antigravity training during the first weekend and worked independently on their projects between the two sets of sessions. The project was organized into four major parts.

Part 1. Strategy. Students began by developing a product strategy that identified the intended users, business value, potential productivity benefits, relevant external data sources, and an appropriate technology stack. They then documented the initial prompts they planned to use in development. The written assignment did not prescribe a single mandatory prompting framework; instead, students were asked to apply their understanding of prompt-engineering frameworks discussed in class, and their prompts were assessed on how effectively they incorporated structured prompt-design principles. The course materials drew on approaches represented by Djeffal's (2025) reflexive prompt engineering framework and the prompt-engineering framework discussed by Nair et al. (2025). Students therefore had flexibility in how they structured their prompts rather than being required to use one common template.

Part 2. Development. Students used Google Antigravity to create an initial prototype of their intended application. They established a project folder, opened it in the development environment, and entered prompts describing the application's purpose, intended users, capabilities, user interface, and desired features. Students were encouraged to incorporate external sources such as financial data, news feeds, weather information, currency exchange rates, public calendars, or other business-relevant data (see Appendix C). They also specified visual and interaction-design elements, including navigation, branding, fonts, logos, and other interface features. This stage produced an initial proof of concept that could then be tested and refined.

Part 3. Testing and Refinement. Students interacted with their prototypes through the browser interface and evaluated whether navigation, interface components, data, and other functions worked as intended. They tested valid and invalid input, identified features to add, modify, or remove, and revised their prompts accordingly. Students recorded how revised prompts changed the prototype and assessed whether those changes improved its quality, functionality, or usability. Each student also maintained an error log documenting cases in which the AI produced incomplete, inaccurate, or otherwise problematic outputs and described the steps taken to address those problems.

Part 4. Executive Summary and Analysis. Students prepared an executive summary and final presentation materials. They used the Antigravity Agent Manager to summarize the prompts and

development interactions associated with their prototype and incorporated that information into presentation slides. Required presentation elements included the project title, a screenshot and description of the prototype, a video demonstration or pitch, the technologies used, an example from the error log, and lessons learned. Students also generated a management log documenting the AI-assisted development conversation.

The project culminated in an in-class demonstration and pitch. Students described the business problem, demonstrated their prototype, identified important failures or limitations encountered while working with AI, offered recommendations, and reflected on lessons learned. Peer critique and evaluation were also part of the learning experience. Assessment was distributed across five categories: prototype functionality, including relevant data use and whether the prototype worked as intended; user interface and user experience; the in-class demonstration and critiques; executive-summary slides; and the project log.

Overall, the assignment positioned students as managers of an AI-assisted development process. Their work demonstrated how they identified a business need, translated that need into requirements and prompts, guided an AI development agent as it proposed and implemented technical solutions, evaluated the resulting artifacts, corrected errors, and communicated the value and limitations of their prototypes. The grading rubric therefore reflected both prototype quality and the process used to develop, test, revise, and explain it.

5. STUDENT OUTCOMES AND REFLECTIONS

This section reports prototype outcomes and learning evidence from the student projects, process documentation, presentation and reflection artifacts, and end-of-lab survey responses. The full survey instrument is provided in Appendix D. The survey was administered at the end of the technology lab, after students completed their prototypes, and presented their projects in class. All six enrolled students completed the survey. Because the survey was used for course reflection and instructional improvement in such a small class, responses were analyzed descriptively rather than using statistical methods. Students were informed that course project examples and survey responses would remain anonymous and could be used in the aggregate to evaluate the learning process rather than assessing individual students' experiences or performance. The activity was conducted as part of normal course instruction, and the survey was administered for student reflection and to provide suggestions for improving the course.

We assessed the project evidence using the criteria established in Section 3.2: the quality of students' problem framing and prototype development, evidence of iterative testing and revision, and their ability to identify and explain the limitations of the resulting prototypes. We did not assess success in terms of production readiness. Instead, we examined whether students could define a meaningful business or professional problem, guide an AI development agent through development iterations, test and revise the resulting prototype, and articulate what the prototype could and could not do. The end-of-lab survey, completed by all six students, provided additional descriptive evidence about perceived value, self-reported learning, and students' experiences with vibe coding. Open-ended responses and self-selected adjectives were used to complement the project and reflection evidence rather than as independent measures of learning.

The six projects demonstrated the variety of business and professional problems that students attempted to address through vibe coding. The *Recalivate* project translated the idea of reusable learning and decision playbooks into a sophisticated local-first prototype with dashboards, playbooks, decision logs, guardrails, and persistence. The *ProcureCast* project addressed a concrete procurement problem by using scrap-cost and supply-chain-risk data to identify high-impact materials and monitor external disruption. The *SpinOn* project applied vibe coding to a dorm-laundry management problem, adding cycle tracking, alerts, services, payment/account features, and lost-and-found support through successive prompts. The *Smart Exit* project created a stock-monitoring and sell-signal tool that integrated market recommendations, portfolio tracking, macroeconomic data, and explainable review/sell logic. The *Realty Monitor* project focused on building operations, sustainability, and municipal compliance monitoring for biotech office spaces managed by the student's employer. *Project Recomp* was technically ambitious, using a fitness and body-composition use case to explore mobile development, predictive metrics, backend services, and visualization of body-shape change.

ISSN: 2473-4901
v12 n6798

Across these projects, the clearest evidence of learning came from the combination of project artifacts, process documentation, presentation and reflection artifacts, and end-of-lab survey responses. The prototypes showed what students built; the logs and demonstrations showed how they managed the AI-assisted development process; and the survey and reflection evidence showed how they interpreted the value, frustration, and limits of vibe coding. Taken together, these sources suggest that students moved beyond initial enthusiasm toward a more realistic understanding of AI-assisted development as a process requiring clear prompting, testing, revision, scope management, and human oversight.

5.1 Perceived Value of Technology Lab Activities

The survey items used to collect these responses are listed in Appendix D. Six students were enrolled in our Professional MBA technology lab, and all completed an end-of-lab survey focused on the perceived value of the lab activities, what they learned, and their vibe-coding experience. For the activity-value items, students responded using a five-category scale: Strongly Disagree, Disagree, Somewhat, Agree, and Strongly Agree. Figure 2 reports responses for the two vibe-coding activities, labeled in the survey as "Vibe Coding In Class" and "Vibe Coding Project." Although we estimated that students would spend 8 to 10 hours completing the project, four reported spending 10 to 12 hours, one reported more than 12 hours, and one reported 6 to 8 hours.

Five of the six students selected Agree or Strongly Agree for both the in-class vibe coding activity and the vibe coding project. For the in-class activity, two students selected Agree and three selected Strongly Agree; for the project, one selected Agree and four selected Strongly Agree. The remaining student selected Somewhat for each activity.

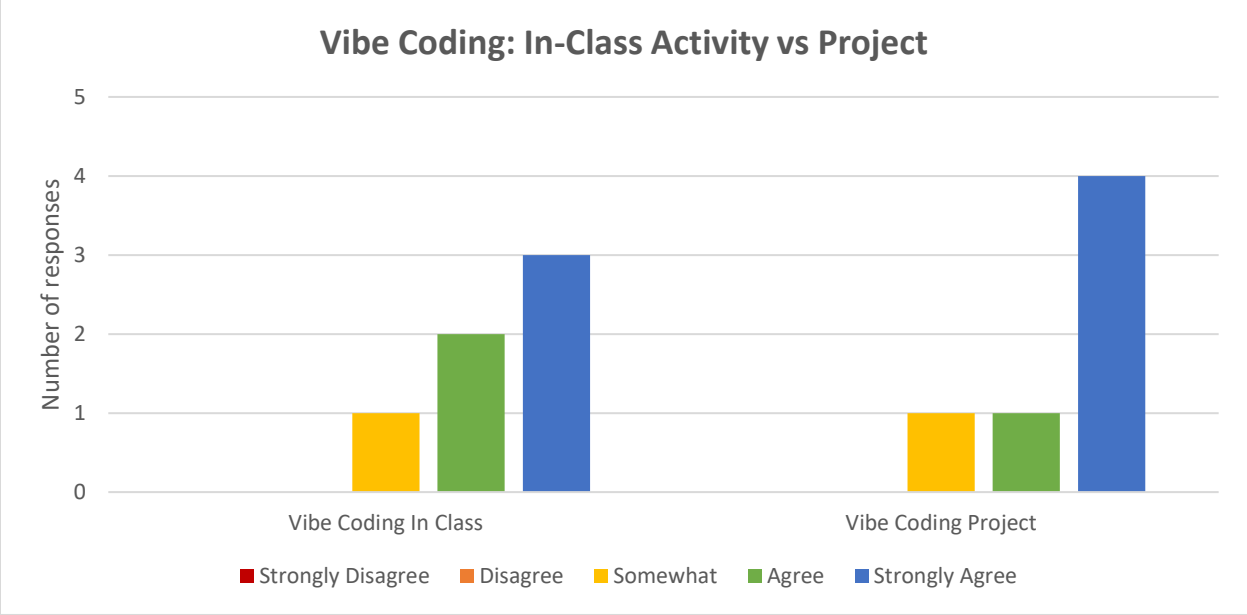

Figure 2. Perceived Value of Technology Lab Activities
(Source: Authors, based on end-of-lab survey responses).

5.2 Self-Reported Learning Outcomes from Vibe Coding

Figure 3 summarizes student responses to four survey statements related to vibe coding and AI skill-building: "The topic of vibe coding was new to me"; "The vibe coding assignment helped me to turn my idea into a potential product/service"; "I feel more confident in using AI tools"; and "I feel better prepared to use or evaluate AI tools for academic or professional contexts." Students responded using the same five-category scale: Strongly Disagree, Disagree, Somewhat, Agree, and Strongly Agree.

ISSN: 2473-4901
v12 n6798

Five of the six students agreed or strongly agreed that vibe coding was new to them, suggesting that most participants were not previously familiar with AI-assisted development. Five of the six also agreed or strongly agreed with each of the three learning and application statements: that the vibe coding assignment helped them turn an idea into a potential product or service, that they felt more confident using AI tools, and that they felt better prepared to use or evaluate AI tools in academic or professional contexts.

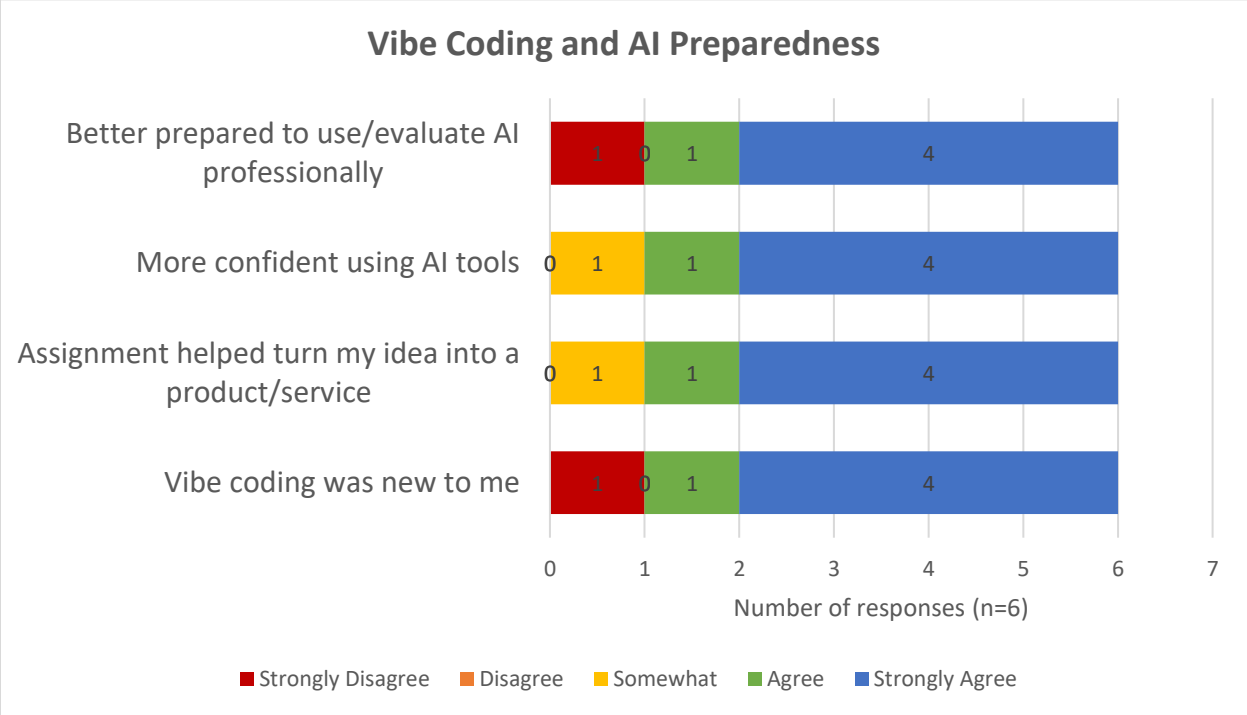

Figure 3. Vibe Coding and AI Skill-Building Outcomes (Source: Authors, based on end-of-lab survey responses).

5.3 Student Perceptions of the Vibe Coding Experience

Students were asked to supply three adjectives of their choice to describe the vibe coding project. For descriptive presentation, the authors grouped these self-selected adjectives into positive, neutral, and negative categories. Figure 4 shows the frequency of each adjective and its assigned category. These responses provide an initial view of the student experience, while the open-ended survey comments that follow complement the broader reflection evidence described in Section 3.3 by showing how students interpreted prompting difficulties, iterative development, system breakdowns, and the continuing need for human oversight.

The adjective responses suggest that students had a mixed but generally positive experience completing the vibe coding project. *Frustrating* was the most frequently reported adjective (n = 3), while *tedious* and *repetitive* were each reported once. Students also described the experience as *interesting* (n = 2) and as *beneficial, convenient, creative, efficient, fun, practical,* and *useful* (n = 1 each). Neutral descriptors — *iterative, challenging, different,* and *long* — reflected the evolving and trial-and-error nature of the AI-assisted development experience.

When asked to identify the most important thing they learned from completing the vibe coding project, students emphasized human–AI collaboration, prompt engineering, iterative development, and the need to critically evaluate AI-generated outputs. Several students emphasized the importance of communicating effectively with AI systems, noting that "[S]ometimes you need to change the way you are saying something to have AI understand it," and that "when prompting, it is important to be detailed." Other responses highlighted the value of iterative development, with one student

ISSN: 2473-4901
v12 n6798

commenting, "I learned to start small and then grow the product after different iterations. I personally started too big and think it hindered my success."

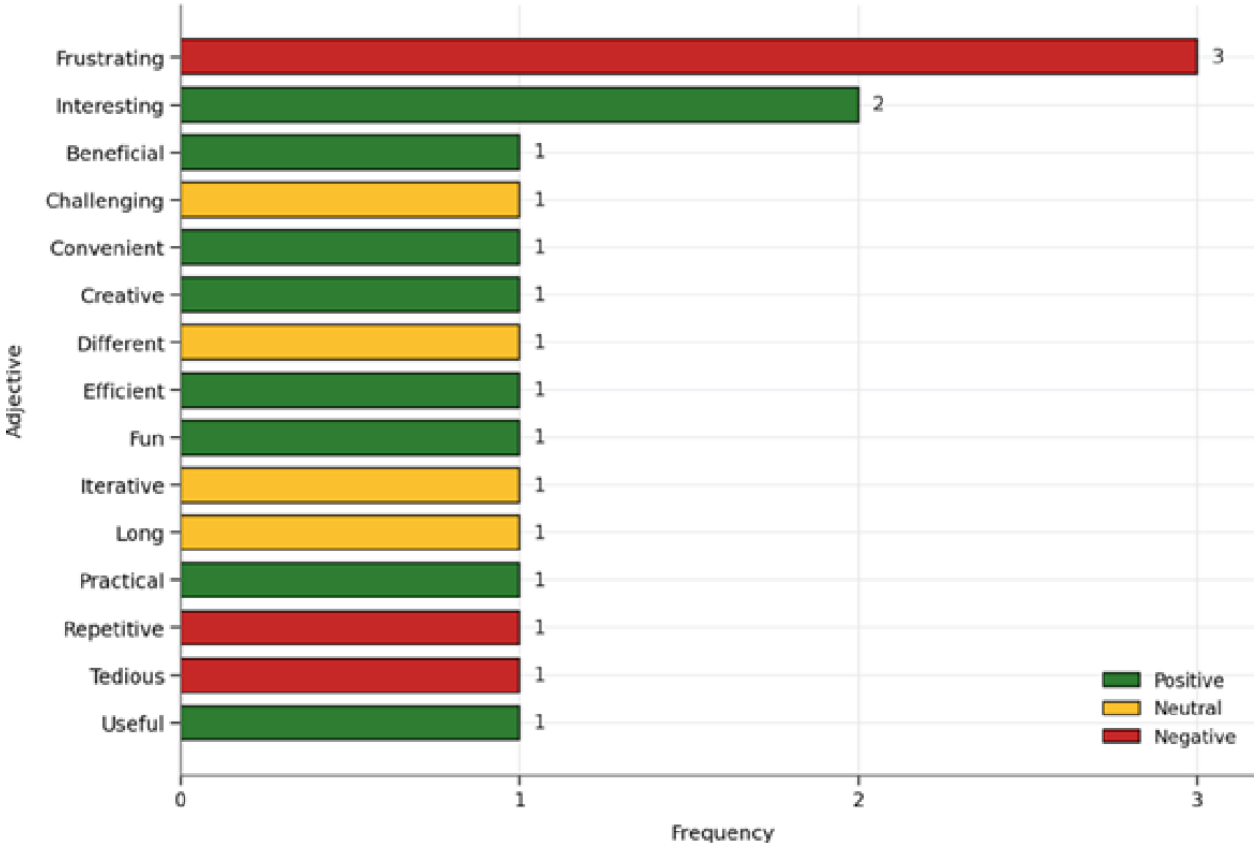

Figure 4. Frequency of Adjectives Used to Describe the Project (Source: Authors, based on end-of-lab survey responses).

Students also gained an appreciation of the role that AI can play in software development, noting that "AI is not replacing software developers" and that "AI can break things easily and set you back just as quickly as it can get you something fast." These responses suggest that students came to view vibe coding as a collaborative process requiring careful prompting, continued refinement, and human oversight and judgment.

Students' advice to future participants reinforced these learning experiences. Several emphasized iteration and precision when working with AI-assisted tools, advising future students to take time to iterate, pay close attention to prompts, start early, and learn from the process. Others emphasized experimentation and creativity, suggesting that the value of the project was not only in building the prototype but also in learning how to ask better questions, recover from unsatisfactory outputs, and manage the AI-assisted development process.

6. CONCLUSION

Several lessons emerged from this six-student implementation. First, vibe coding appeared to lower the barrier to creating working prototypes while giving students opportunities to develop practical skill and judgment in AI-assisted development through requirements definition, prompt design, testing, and evaluation. Second, students benefited from conceptual scaffolding before using AI-assisted development tools. In this technology lab, instruction on front-end/back-end architecture, APIs, data sources, and deployment helped students understand what their prototypes were doing and why they sometimes failed. Third, scope control was important. Students were most successful when they began with a focused proof of concept and expanded through iteration rather than attempting production-ready systems. Finally, reflection was an important part of the learning process because logs,

presentations, and demonstrations required students to diagnose errors, revise prompts, and recognize the limits of AI-generated prototypes.

Because this implementation involved six students in a single Professional MBA technology lab, these findings should be treated as exploratory rather than broadly generalizable. Implementation limitations included uneven reflection artifacts, insufficient instruction on recovery practices, and limited structure in the peer-evaluation data. Some students documented rich error logs and decision histories, while others provided stronger project plans or presentation reflections than written process evidence. A future offering of the technology lab should require a standardized reflection template at multiple checkpoints, provide more explicit instruction on saving working versions, rolling back changes, isolating features, and distinguishing mock data from live data, and capture more structured quantitative and qualitative peer-evaluation data.

Within this small implementation, vibe coding functioned as a project-based learning activity that helped students experience AI-assisted development as a managerial, analytical, and reflective process. Students built meaningful prototypes, but they also encountered ambiguity, fragility, and failure. These challenges were instructionally valuable because they required students to test outputs, revise prompts, reconsider scope, and distinguish surface-level functionality from more robust application behavior. Taken together, our experience suggests that a carefully scaffolded learn–apply–reflect design can use vibe coding to support practical skill and judgment, strengthen understanding of AI's capabilities and limitations, and reinforce the need for human oversight and judgment in AI-assisted application development.

7. ACKNOWLEDGMENTS

The authors created Figure 1 using the following icons from TheNounProject.com (CC BY 3.0), for which attribution is required: Mind and gear by Sari Braga; open book with lightbulb by jowy san; and check list by Azzam.

8. REFERENCES

Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010). *How learning works: Seven research-based principles for smart teaching*. Jossey-Bass.

Argyris, C., & Schön, D. (1978). *Organizational learning: A theory of action perspective*. Addison-Wesley.

Bamil, V. (2025). *Vibe coding: Toward an AI-native paradigm for semantic and intent-driven programming*. arXiv. https://arxiv.org/abs/2510.17842

Barricelli, B. R., Cassano, F., Fogli, D., & Piccinno, A. (2019). End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software, 149*, 101–137. https://doi.org/10.1016/j.jss.2018.11.041

Barron, B. J. S., Schwartz, D. L., Vye, N. J., Moore, A., Petrosino, A., Zech, L., & Bransford, J. D. (1998). Doing with understanding: Lessons from research on problem- and project-based learning. *Journal of the Learning Sciences*, *7*(3–4), 271–311. https://doi.org/10.1080/10508406.1998.9672056

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist*, *26*(3–4), 369–398. https://doi.org/10.1080/00461520.1991.9653139

Bock, A. C., & Frank, U. (2021). Low-code platform. *Business & Information Systems Engineering, 63*, 733–740. https://doi.org/10.1007/s12599-021-00726-8

Boud, D., Keogh, R., & Walker, D. (Eds.). (1985). *Reflection: Turning experience into learning*. Routledge.

Crowson, M. G., & Celi, L. A. (2025). *Academic vibe coding: Opportunities for accelerating research in an era of resource constraint*. arXiv. https://arxiv.org/abs/2508.00952

David, E. (2025, November 17). Google Antigravity introduces agent-first architecture for asynchronous, verifiable coding workflows. *VentureBeat*. https://venturebeat.com/orchestration/google-antigravity-introduces-agent-first-architecture-for-asynchronous

Djeffal, C. (2025). Reflexive prompt engineering: A framework for responsible prompt engineering and AI interaction design. In *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency* (pp. 1757–1768). Association for Computing Machinery. https://doi.org/10.1145/3715275.3732118

Frydenberg, M., Mentzer, K., & Patterson, A. (2026). The rapid rise of generative AI adoption among first-year college students. *Information Systems Education Journal*, *24*(1), 4–18. https://doi.org/10.62273/HVNN2048

Gibbs, G. (1988). *Learning by doing: A guide to teaching and learning methods*. Further Education Unit.

Google Antigravity Team. (2025, November 20). Build with Google Antigravity, our new agentic development platform. *Google Developers Blog*. https://developers.googleblog.com/build-with-google-antigravity-our-new-agentic-development-platform/

Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the effect of AI code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Article 455, pp. 1–23). Association for Computing Machinery. https://doi.org/10.1145/3544548.3580919

Kokotsaki, D., Menzies, V., & Wiggins, A. (2016). Project-based learning: A review of the literature. *Improving Schools*, *19*(3), 267–277. https://doi.org/10.1177/1365480216659733

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Prentice-Hall.

Moon, J. A. (1999). *Reflection in learning and professional development: Theory and practice*. Kogan Page.

Nair, A. B., Gafoor, F. A., Reddy, B. V., Sivasakthivel, R., RajaGopal, M., & Ramar, G. (2025). Strategical framework design for prompt engineering analysis - ChatGPT. In *Proceedings of the 2025 International Conference on Computing for Sustainability and Intelligent Future (IEEE COMP-SIF)* (pp. 1–8). IEEE. https://doi.org/10.1109/COMP-SIF65618.2025.10969948

Preston, D. (2025, November 18). Google Antigravity is an "agent-first" coding tool built for Gemini 3. *The Verge*. https://www.theverge.com/news/822833/google-antigravity-ide-coding-agent-gemini-3-pro

Sarkar, A., Gordon, A. D., Negreanu, C., Poelitz, C., Srinivasa Ragavan, S., & Zorn, B. (2022). What is it like to program with artificial intelligence? In *Proceedings of the 33rd Annual Workshop of the Psychology of Programming Interest Group (PPIG)* (pp. 127-153).

Schön, D. A. (1983). *The reflective practitioner: How professionals think in action*. Basic Books.

ISSN: 2473-4901
v12 n6798

APPENDIX A

Course Description and Brief Syllabus

BENTLEY UNIVERSITY Professional MBA
PML 672 — Mastering Artificial Intelligence for Professional Excellence
Spring 2026

INSTRUCTORS: Mark Frydenberg, David Yates

DESCRIPTION:

The emergence of Artificial Intelligence (AI) in business and educational environments highlights the interest in this rapidly growing technology. From educational and business uses to ethical concerns that arise from its use, companies are looking for ways to capitalize on this technology for the new opportunities it enables to create products and improve productivity.

This tech lab will equip you with the essential knowledge and skills to leverage AI tools effectively in your academic and professional endeavors. Discover how to streamline notetaking with AI-powered features of meeting platforms and create professional visuals using AI-generated images. Learn how to construct, customize, and refine prompts for professional use cases. Students will gain the knowledge and skills to harness the power of AI to help them achieve their goals.

TECH LAB LEARNING OBJECTIVES:

This 0.5 credit tech lab will address the following PMBA program learning outcomes:

Augmented PMBA Learning Objectives	How is the Learning Objective assessed?
Identify and analyze the impact of emerging technologies for business use. Critically evaluate the business applications of artificial intelligence (AI) by analyzing real-world case studies and engaging in discussions that assess AI's strategic impact, ethical considerations, and implementation challenges.	Instructors will assess student evaluations of AI applications for business through in-class discussions and applied case studies.
Argue the strategic perspective of adopting and bypassing AI technologies within an organizational context. Evaluate the strategic trade-offs of adopting AI technologies by creating, refining, and analyzing AI-generated content, assessing its implications for business decision-making, competitive advantage, and organizational risk management.	Instructors will assess student work in which students create, refine, and analyze AI-generated content to explore its implications in business decision-making.
Analyze, design, and defend taking innovative and alternative paths to meet an objective. Design and defend AI-driven solutions to address business challenges, critically evaluating the efficacy of generative AI in professional settings while considering innovation, feasibility, and strategic	Instructors will assess student work in which they design AI-driven solutions to address business challenges and evaluate the efficacy of generative AI in professional settings.
Apply design and systems thinking in developing innovative solutions to complex problems. Utilize design and systems thinking to evaluate and implement AI-enabled tools, working in teams to develop innovative solutions that optimize business operations and address complex organizational	Instructors will assess student team work in which students explore AI-enabled tools and their impact on business operations.

TECH LAB EXPECTATIONS: Each tech lab will consist of six hours of in-person time and twelve hours of asynchronous work time. The in-person time will take place during the designated on- campus weekends. Assigned asynchronous work is to be completed as per schedules indicated. During the in-person classes, students are expected to be engaged and actively participating in all course activities.

- Tech Labs will include the following requirements:
- Four 1.5-hour in-person sessions led by the instructor during on-campus weekends.
- Twelve (12) hours of asynchronous work.
- Deliverables for grade aligned with workshop learning outcomes.

GRADING SCALE:

Introduction Video and Comments (due before)	20%
Collaborative Slides	10%
AI Vibe Coding Project (written submission)	30%
AI Vibe Coding Project (in-class presentation)	20%
Participation and Attendance	20%
TOTAL	100%

AI Policy:

In addition to the required use of generative AI tools to complete work assigned for this tech lab, the use of generative AI tools is permitted in this lab for the following activities:

- Brainstorming and refining your ideas,
- Finding information on specific topics,
- Drafting an outline to organize your thoughts, and
- Checking grammar and style.

The use of generative AI tools is not permitted in this course for the following activities:

- Impersonating you in classroom contexts,
- Completing individual or group work unless generative AI use is explicitly permitted or required by the assignment,
- Writing a draft for an assignment, and
- Writing entire sentences, paragraphs, or papers to complete class assignments.

You are responsible for the information you submit based on an AI query (for instance, that does not violate intellectual property laws, or contain misinformation or unethical content). Your use of AI tools must be properly documented and cited in order to stay within university policies on academic honesty. Any assignment that is found to have used generative AI tools in an unauthorized manner will be subject to an Academic Integrity review. If you are in doubt about permitted usage, please ask for clarification.

COURSE SCHEDULE

NOTE: Each class is 1.5 hours in length.

Class	Topic	Assignments
Weekend 1, Class 1	Implications of AI for Business	Discussion
Weekend 1, Class 2	Introduction to Vibe Coding	Vibe Coding In-Class Lab
	Work on Vibe Coding Project	
Weekend 2, Class 1	Using AI to Analyze Data	In Class Lab
Weekend 2, Class 2	Project Presentations and Critiques	Project Presentations

ISSN: 2473-4901
v12 n6798

APPENDIX B

In-Class Vibe Coding Tutorial Summary

This appendix summarizes the in-class vibe coding activity used in our Professional MBA technology lab, [redacted course number]. The activity introduces students to AI-assisted software development through Google Antigravity, an agentic development environment in which users describe an intended application in natural language and work iteratively with an AI development agent to generate, test, and refine working software. Rather than emphasizing line-by-line programming, the activity positions students as managers of AI-assisted development. They define the product goal, approve or revise implementation plans, monitor the agent's work, evaluate the output, and request improvements based on observed functionality.

The activity asks students to build an executive dashboard application that displays stock information, business news headlines, and a space for logging daily wins. Students begin by installing Antigravity, creating a project folder, and configuring agent settings that determine how much autonomy the AI has when reviewing artifacts, executing terminal commands, and accessing files. They then submit a detailed prompt specifying the application name, purpose, desired visual style, core features, and expectation that the agent recommend an appropriate lightweight technology stack before implementation begins.

After the initial application is generated, students run the project locally and evaluate whether the user interface and functionality match the stated requirements. The activity intentionally emphasizes iteration. Students are prompted to identify limitations, such as static displays or non-functional navigation icons, and then use additional natural-language prompts to request revisions. In doing so, they practice translating observed software behavior into actionable feedback for an AI development agent. The activity concludes with optional feature extensions, such as adding additional stock symbols, and a required management log in which students ask the agent to summarize the decisions and

Activity phase	Student action	Learning purpose
Setup	Install Antigravity, create project folders, and configure agent permissions.	Prepare the development environment and understand how AI autonomy is managed.
Prompting	Describe the executive dashboard goal, visual style, features, and desired implementation process.	Practice translating product requirements into clear natural-language specifications.
Generation	Review the agent's recommended technology stack and implementation plan before proceeding.	Develop judgment about AI-generated plans and human oversight of automated work.
Testing	Run the dashboard locally and inspect whether the interface and functions work as expected.	Distinguish surface-level output from functioning software and verify generated results.
Revision	Identify limitations in the generated dashboard and use follow-up prompts to request improvements.	Practice iterative refinement and learn how to translate observed problems into actionable instructions for an AI development agent.
Reflection	Generate a management log summarizing prompts, decisions, errors, and revisions during development.	Connect tool use to reflective judgment about AI-generated functionality, limitations, and human oversight.

Table 1. Vibe Coding Tutorial Activity Flow and Learning Purpose (Source: Authors, based on course materials).

ISSN: 2473-4901
v12 n6798

interactions that occurred during development. Table 1 (on the previous page) summarizes the activity phases, student actions, and learning purposes.

The activity supports several learning outcomes relevant to business and information systems education. Students gain experience with prompt-based specification, management of AI-assisted development, tool configuration, iterative testing, and reflective documentation. The exercise also exposes students to the practical limits of AI-generated prototypes, including the difference between a polished interface and fully implemented functionality, the need to verify output, and the importance of revising prompts when the generated prototype does not meet expectations.

The completed executive dashboard is shown in Figure 5.

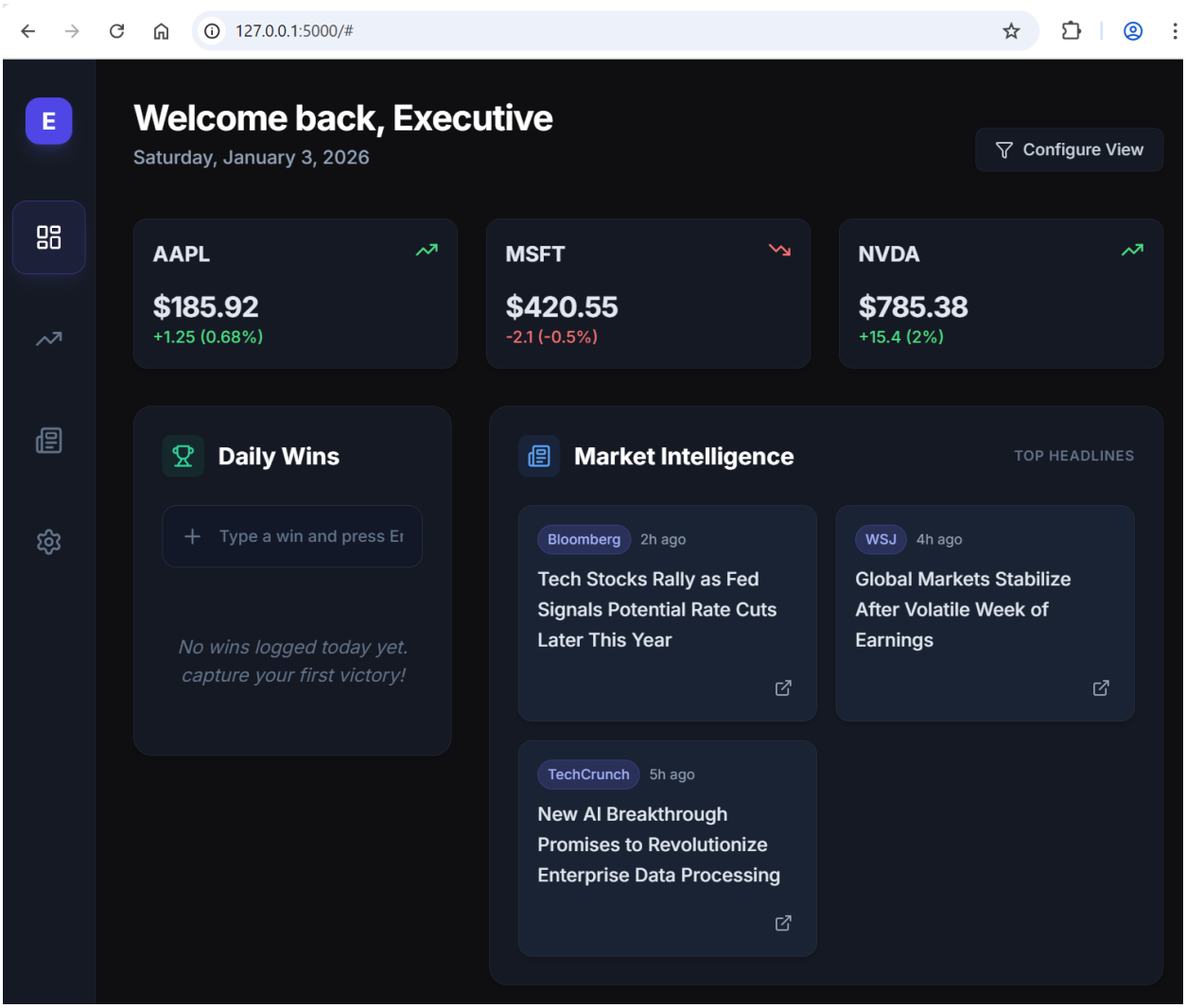

Figure 5. Completed Vibe Coding Tutorial Activity
(Source: Authors, created with Google Antigravity).

ISSN: 2473-4901
v12 n6798

APPENDIX C

Suggested External Data Sources

Financial & Market Data	
Yahoo Finance (yfinance)	Provides stocks, ETFs, currencies, and cryptocurrency data without requiring an API key, making it accessible for beginner projects.
Alpha Vantage	Supports specialized financial data such as commodities, stock prices, and economic indicators, though it requires a free API key.
FRED	Federal Reserve Economic Data source useful for macroeconomic dashboards, including interest rates, unemployment, inflation, and housing data.
News & Industry Intelligence	
RSS Feeds	Open feeds from major news outlets that can be used to display current business headlines or industry updates.
Google News RSS	Customizable by search term, allowing students to create topic-specific feeds such as supply chain disruption or retail technology.
Reddit API	Can support market sentiment or trend-monitoring features by pulling recent discussion topics from business or investing communities.
Real-World Environment & Logistics	
OpenWeatherMap	Provides current weather, forecasts, and air quality data for logistics, retail, travel, and location-based applications.
REST Countries API	Offers country-level data such as currencies, population, languages, and time zones without requiring an API key.
Exchange Rate API	Provides real-time currency conversion rates for international business, travel, procurement, and finance applications.
Productivity & Utilities	
Public Holiday API (Nager.Date)	Enables global business calendar features by identifying public or bank holidays in different countries for scheduling and planning.

ISSN: 2473-4901
v12 n6798

Appendix D.

PML 672 Reflection Survey Instrument

This appendix presents the reflection survey administered to participants following the PML 672 technology workshop. Items marked with an asterisk required responses.

Section	Survey item	Response format
Assignments: perceived value	Which assignments were most valuable to your learning? * Prompting Activity In-class Vibe Coding In-class Data Analysis In-class Vibe Coding Project Cases	Likert scale: Strongly Disagree, Disagree, Somewhat Agree, Strongly Agree. Respondents marked one option per row.
Vibe coding project	Three adjectives to describe the vibe coding project. *	Open-ended response. No prompting given.
Assignments: learning outcomes	The prompting activity helped me learn the importance of constructing structured prompts. The data analysis activity helped me better understand how AI can be used to analyze real data. The vibe coding assignment helped me turn my idea into a potential product/service. The topic of vibe coding was new to me.	Likert scale: Strongly Disagree, Disagree, Somewhat Agree, Strongly Agree. Respondents marked one option per row.
AI learning and confidence	I now have a better understanding of when and why to use AI tools, rather than just how. Vibe coding was new to me. The vibe coding assignment helped me turn my idea into a potential product/service. I feel more confident in using AI tools. I feel better prepared to use or evaluate AI tools for academic or professional contexts.	Likert scale: Strongly Disagree, Disagree, Somewhat Agree, Strongly Agree. Respondents marked one option per row.
Time on task	How much time did you spend on the vibe coding project in total? *	Multiple choice: 0 to 4 hours; 4 to 6 hours; 6 to 8 hours; 8 to 10 hours; 10 to 12 hours; More than 12 hours.
Workload	For most of this workshop, the workload was *	Multiple choice: Very manageable; Manageable; Heavy but

ISSN: 2473-4901
v12 n6798

Section	Survey item	Response format
		manageable; Way too much and overwhelming.
Project reflection	What is the most important thing you learned by completing the vibe coding project? *	Open-ended response.
Course improvement	If this course is offered again, what should we keep, modify, or remove? *	Open-ended response.
Additional content	Anything we should add? *	Open-ended response.
Advice to future students	What advice would you give to someone taking this workshop in the future? *	Open-ended response.
Recommendation	Based on my tech lab experience this term, I would recommend this course to students enrolling in Spring 2027. *	Five-point scale from 1 to 5, anchored by Strongly Disagree and Strongly Agree.