Journal of Contemporary Educational Research, 2026, Volume 10, Issue 10
https://bbwpublisher.com/journal/JCER
Online ISSN: 2208-8474
Print ISSN: 2208-8466

Exploration of Teaching Reform for the Web Language Foundation Course in the Context of AI Assistance

Yiyong Ye[1]*, Liqing Huang[2]

[1]School of Economics and Management, Wuyi University, Jiangmen 529020, Guangdong, China
[2]Jiangmen Polytechnic, Jiangmen 529020, Guangdong, China

Corresponding author: Yiyong Ye, yyyong2022@163.com

Copyright: © 2026 Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0), permitting distribution and reproduction in any medium, provided the original work is cited.

Abstract: Generative artificial intelligence (AI) is profoundly reshaping the teaching paradigm of programming courses in universities. Taking the "Web Language Foundation" course offered by the School of Economics and Management at Wuyi University as the research object, this paper systematically examines the impact of AI-assisted programming tools on course instruction. The study finds that while AI technology brings opportunities such as personalized learning, efficiency gains, and instructional mode innovation to the course, it also poses practical challenges, including the hollowing-out of fundamental skills, the misalignment of evaluation systems, the transformation of teachers' roles, and academic integrity concerns. In response to these dilemmas, and in light of the course's specific context, this paper proposes a systematic teaching reform framework oriented toward "human-machine collaboration and competency development." The framework covers six dimensions: reconstructing course objectives, optimizing teaching content, innovating instructional models, reforming the evaluation system, strengthening teacher capacity, and improving institutional guarantees. It provides a reference for the digital and intelligent transformation of similar applied undergraduate programming courses.

Keywords: Artificial intelligence; Web Language Foundation; Teaching reform

Online publication: November 11, 2026

1. Introduction

Since 2022, generative artificial intelligence (AI) tools such as ChatGPT, GitHub Copilot, and DeepSeek have become pervasive in software development and programming education, with capabilities including automatic code generation, semantic explanation, and intelligent error localization. In web development, AI tools can generate both front-end (HTML, CSS, JavaScript) and back-end (C#, ASP.NET, ADO.NET) code with quality comparable to that of junior developers / at a quality comparable to that of a junior developer [1,2].

Web Language Foundation is a compulsory foundational course for e-commerce majors at Wuyi University, covering C# programming, object-oriented development, ASP.NET controls, and ADO.NET database access through project-based teaching. The 2024 syllabus introduced AI-enabled instructional design, but current fragmented AI usage has not yet formed a coherent framework; systematic reconstruction of objectives, content, methods, and evaluation is needed.

Against this backdrop, systematically clarifying the opportunities and challenges of AI-empowered instruction and reconstructing the curriculum system hold significant practical value for advancing the digital transformation of programming courses for economics and management majors.

2. Development status of AI-assisted programming tools and their impact on course teaching

2.1. Types of AI-assisted programming tools in mainstream educational scenarios

AI-assisted programming tools fall into three categories: general-purpose LLMs (ChatGPT, Claude, DeepSeek) for conversational assistance [3]; specialized IDE-integrated assistants (GitHub Copilot, Cursor) for real-time completion [4]; and AI-enabled teaching platforms with automated grading [5]. The first two directly support C#/ASP.NET instruction.

2.2. Defining the capability boundaries of AI tools in web development

Mainstream AI tools offer three key advantages in C#/ASP.NET development: code generation for syntax, controls, and database operations at junior-developer quality; code explanation and debugging with line-by-line logic clarification; and code optimization for readability and database performance [6].

Nevertheless, AI tools have notable limitations: generated code may contain logical flaws and security risks such as SQL injection requiring developer scrutiny; AI cannot handle the setup and configuration of the development environment; and knowledge is constrained by training data, lagging behind the latest .NET advances. These limitations underscore the irreplaceable value of human instruction [7].

2.3. The systemic impact of AI technology on the course teaching system

AI tools exert a systemic impact across four dimensions: objectives must evolve from syntax mastery to principle understanding and code evaluation; content must include AI tool usage and prompt engineering; methods must shift toward human-machine collaboration and personalized learning; and evaluation, currently 30/10/60 percent weighted, must be reconstructed to distinguish AI-assisted work [8,9].

3. Opportunities for course teaching in the context of AI assistance

3.1. Enabling personalized learning and differentiated instruction

Traditional classrooms face a “one-size-fits-all” dilemma that is particularly acute among students of economics and management whose programming foundations vary considerably. AI supports adaptive learning paths by analyzing learning behavior and code submissions to identify weaknesses and recommend tiered exercises; provides round-the-clock personalized tutoring, functioning as a “24/7 teaching assistant,” echoing the “automatic question answering” scenario in the syllabus and preventing frustration and disengagement [10]; and enables differentiated assignments of varying difficulty, translating “teaching students

in accordance with their aptitude" into practice.

3.2. Substantial efficiency gains across the teaching process

AI can support the entire teaching process: in lesson preparation, generating case code, exercises, and courseware (e.g., a complete login-module example from a single requirement); in classroom teaching, giving live demonstrations of how AI generates C# code and guiding students to observe, analyze, and evaluate its quality, transforming the classroom into "teacher-guided analysis with student critical thinking"; in grading, automatically identifying syntax errors, style issues, and vulnerabilities such as SQL injection, freeing teachers to assess programming thinking; and in learning analytics, analyzing submission frequency and error-type distributions to support instructional decision-making [5].

3.3. Dynamic updating and frontier-oriented upgrading of teaching content

The .NET technology stack evolves rapidly, while textbook updates lag. AI bridges this gap with cutting-edge information and industry-relevant case code. More importantly, AI itself should become core curriculum content; the current syllabus focuses on AI-assisted learning but has not treated AI-enabled development as teaching content. Students should learn prompt writing, AI code review, and end-to-end code comprehension systematically [6].

3.4. Innovation in teaching models and transformation of classroom formats

AI tools enable three new teaching models: the human-machine dual-teacher model, where teachers guide thinking while AI handles demonstration and Q and A; the reverse generative method, where students analyze AI code to trace underlying principles; and the project-driven AI-assisted model, where AI integrates across all development stages to build collaboration skills.

3.5. Expanding the dimensions of student competency development

AI expands competency dimensions beyond coding to include the following: problem definition, translating business requirements into AI instructions; code review, judging correctness and security; human-machine collaboration; critical thinking, recognizing AI limitations; and innovative design, improving AI-generated code. This elevates the course from skills training to the cultivation of higher-order thinking, aligning with the technology-plus-business positioning.

4. Practical challenges for course teaching in the context of AI assistance

4.1. The risk of the hollowing-out of fundamental skills caused by students' over-reliance on AI

The most urgent challenge is the risk of fundamental-skill hollowing-out. When AI readily generates C# class definitions, control declarations, and database code, some students may simply copy and paste the AI-generated output without thinking independently. Basic syntax knowledge fails to be internalized, debugging ability is systematically weakened, and higher-order programming thinking cannot develop. Empirical research confirms that excessive AI reliance hinders the development of independent problem-solving and innovation skills [7].

4.2. The misalignment of the traditional evaluation system

The current assessment, 40% process plus 60% final project, is maladapted to the AI era. Work authenticity is hard to verify; teachers cannot distinguish independently written code from AI-generated code. Evaluation criteria need reconstruction, shifting from functional completeness to comprehension, quality, and creative improvement. Process-based evaluation remains difficult to implement due to privacy and technical barriers.

4.3. Challenges of teachers' role transformation and capacity adaptation

AI integration compels teachers to transform from knowledge transmitters into learning guides, from code demonstrators into facilitators of thinking, and from manual graders into learning analysts. These transformations require mastery of AI tools and curriculum reconstruction, posing a considerable challenge for traditionally trained teachers [9].

4.4. Ethical dilemmas and academic integrity governance

AI introduces several ethical dilemmas. The boundary of academic misconduct has become blurred, with no consensus on permissible AI use. Originality may diminish as over-reliance reduces independent thinking. The digital divide widens as AI proficiency gaps create new inequities. Data security risks arise when students inadvertently share project code or database information with AI systems [3]. Cybersecurity awareness and professional ethics also require new interpretation.

4.5. Lagging teaching resources and infrastructure support

Teaching resources and infrastructure support lag behind actual needs: equal access to AI tools is difficult due to subscription costs; existing teaching platforms are not adapted for AI integration; and textbooks predate AI popularization and lag behind .NET evolution.

5. Implementation paths for teaching reform

5.1. Reconstructing course objectives

Building on the five existing syllabus objectives, the course should upgrade its training objectives from "mastering C# syntax and ASP.NET controls and developing simple dynamic websites" to "deeply understanding web development principles, collaborating efficiently with AI, and critically evaluating and optimizing AI-generated code," with new "AI literacy" objectives and their supporting links to graduation-requirement indicators added when the syllabus is next revised. The objectives form three progressive levels: the foundation level requires completing basic coding independently without AI, since students who have never coded "from scratch" will lack the foundation on which later human-machine collaboration depends; the intermediate level requires proficient use of mainstream AI tools and prompt-writing principles (specific task descriptions, complete context); the advanced level requires independently verifying correctness, performing security checks (e.g., SQL injection inspection), evaluating standardization, identifying logical flaws, and making creative improvements.

5.2. Optimizing teaching content

Content optimization requires both "addition" and "subtraction." In addition, the syllabus's "AI-enabled instructional design" entry should be expanded from "AI-assisted learning" to "AI-assisted development,"

covering the principles and operation of mainstream AI tools, prompt engineering theory and practice, methodologies for reviewing and optimizing AI-generated code (emphasizing common e-commerce security risks such as SQL injection), human-machine collaborative workflows and industry best practices, and AI ethics and academic integrity norms. For subtraction, rote-memorization content should be streamlined—complete property parameter lists of controls and all ADO.NET method signatures—shifting focus from "what needs to be memorized" to "what needs to be understood and judged," and reallocating freed hours to principle explanation and project practice.

5.3. Innovating the teaching model

By combining the project-based teaching thread with the AI participation permitted in labs under the syllabus, a three-stage progressive paradigm can be constructed. Stage one (C# fundamentals and object-oriented labs): AI use is restricted, and fully manual coding is required, with AI-detection tools used to verify the originality of submissions. Stage two (built-in objects and controls labs): AI assistance is formally introduced by expanding the existing "using ERNIE Bot/ChatGPT to complete code writing" activity; students submit AI-generated code together with notes on their own modifications and understanding. Stage three (ADO.NET and comprehensive project practice): free AI use is permitted, but the final project must include an "AI usage declaration" plus segment-by-segment explanations of the principles and quality assessments of AI-generated code. A dedicated "human-vs-machine comparison" activity can also be designed: students code a login module independently, then repeat it with AI, and compare efficiency, quality, and security to grasp AI's capability boundaries intuitively.

5.4. Reconstructing the evaluation system

The overall framework (40% process assessment and 60% final project) can be retained, with intelligent adaptations embedded. First, implement "explanatory assignments" in process assessment: submitted code must be accompanied by a document describing functionality and design rationale, so that a qualified submission is impossible without understanding the code. Second, reform final-project evaluation by adding a documentation component requiring a written explanation of key modules' implementation logic (e.g., the ADO.NET data access layer); students who submit AI-generated code without reasonable explanations should have their scores reduced. Third, establish a standardized "AI usage declaration" system annotating AI-assisted parts, tools, usage patterns, and the student's own modifications. Fourth, expand competency dimensions with four indicators alongside "function implementation"—code comprehension, code review, human-machine collaboration, and problem definition—with well-defined scoring rubrics [5].

5.5. Strengthening teacher capacity building

The course team is the core reform agent. Three avenues are proposed: systematic specialized training covering AI tool usage, AI-assisted instructional design, curriculum reconstruction, and AI ethics, with hands-on practice; a learning community within the course team to collectively refine AI-integrated designs and cases, avoiding transformation pressure on individual teachers; and policy support for teaching and research that encourages action research with this course as the test case, tracking staged AI application data, accumulating localized experience, and feeding it back into syllabus revision.

5.6. Improving the institutional guarantee system

Course-level AI usage guidelines should be issued, defining permitted (designated labs), restricted (comprehensive assignments), and prohibited (on-site assessment of the final project) scenarios in each unit with unified declaration requirements. Resources should be coordinated to build a platform recording AI usage behavior and supporting intelligent learning analytics, centrally procure authorized access to compliant AI tools, and develop AI-integrated lecture notes, case libraries, and exercise sets. The teaching management system should formally recognize AI-assisted teaching, revise course-objective attainment evaluation, incorporate AI literacy objectives into the evaluation loop, and form a virtuous cycle of "objectives-implementation-evaluation-improvement."

6. Conclusion and outlook

AI-assisted programming tools are profoundly reshaping the teaching ecosystem of the "Web Language Foundation" course. AI opens up new possibilities for personalized learning, efficiency, and instructional innovation, but also brings challenges of the hollowing-out of fundamental skills, the inadequacy of evaluation systems, and role transformation. The pioneering exploration embedded in the 2024 syllabus provides a sound starting point, but fragmented applications must be upgraded into systematic, end-to-end reform. When AI can independently complete most basic coding tasks, the core value of programming education lies precisely in cultivating critical thinking, creative design, and ethical judgment—abilities that AI cannot replicate. This is both the core challenge and the educational mission of the course.

Funding

This work was supported by the 2021 University-level Undergraduate High-quality Curriculum Construction Reform Project of Wuyi University (Web Design and Website Construction, Project No. 5071700304C8).

Disclosure statement

The authors declare no conflict of interest.

References

[1] Liu X, Gu T, 2023, The Impact of ChatGPT on Computer Education and Countermeasures. Computer Education, (11): 38–44.

[2] Xu H, Ju X, Wang H, 2023, Challenges and Responses in Programming Teaching in the Era of Large Models. Computer Education, (11): 60–64.

[3] Zhou H, Tian C, Wu X, 2025, Educational Application and Ethical Challenges of AI-Assisted Programming Tools. Computer Technology and Development, 35(12): 123–130.

[4] Ge H, 2025, Research on Innovative Paths of AI Tools Empowering Web Front-End Development Teaching. Computer Knowledge and Technology, 21(34): 37–40.

[5] Xu Z, Huang X, Zhao S, et al., 2026, Exploring the Teaching Mode of the Java Programming Course Integrating AIGC and Process Evaluation. Computer Education, (5): 223–228.

[6] Wu Q, Cheng X, Zhao G, et al., 2025, Opportunities and Challenges of AIGC-Assisted Computer Programming Teaching. Computer Education, (7): 69–73.

[7] Qu H, Liu P, Ma H, et al., 2024, Challenges and Responses of Intelligent Language Model Tools to Programming Course Teaching. Computer Education, (5): 107–110, 117.

[8] Liu L, Wu Y, Chen W, et al., 2025, Challenges and Opportunities of Large Language Models for Programming Courses in Universities. Computer Education, (2): 118–122.

[9] Chen H, 2024, Artificial Intelligence Empowers the Teaching of Programming Courses - Taking JavaScript Programming Course as an Example. Computer Knowledge and Technology, 20(28): 137–139.

[10] Li X, Lu J, Niu J, 2025, Research on Human-Machine Dual-Teacher Collaborative Teaching Empowered by GenAI: A Case Study Based on the Computer Fundamentals Courses at Tsinghua University. Modern Educational Technology, 35(3): 34–43.

Publisher's note

Bio-Byword Scientific Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.