AI Web Agents Can Effectively Guide Lesson Design and Predict Student Outcomes

Sierra Wang, John Mitchell, and Chris Piech

Stanford University
{sierraw,john.mitchell,piech}@stanford.edu

Abstract. A central task in online learning is to design and test learning experiences that blend good pedagogy with elegant user interaction. In this work, we show that LLM-based web agents provide an efficient and helpful way to get immediate feedback on a learning design. We create a web agent that engages with an online learning experience much like a student would – autonomously navigating the interface and analyzing the website content – and generates a comprehensive description of the student experience. We validate the agent's ability to understand student behavior by showing that it is useful in predicting student outcomes in a massive, open-access, online CS1 course. Specifically, agent-generated descriptions significantly improve our ability to predict student dropout in a lesson, outperforming all other measured approaches. We also demonstrate how designers could use the agent to compare potential lesson designs through case studies. Our qualitative analysis confirms that the agent can be used to identify confusing lesson content and provide actionable feedback for the designer. Such AI web agents have the potential to accelerate meaningful advancements in online learning by giving fast, low-cost, and helpful insights for new learning experiences.

Keywords: Web agent · Learning experience design · Student outcomes

1 Introduction

One of the main bottlenecks in developing effective online education is the time and human resources needed to test the learning activities. When a new learning experience is created, human testers can find errors or provide suggestions for improvement, but in many cases, the first critical feedback comes from students *after* the course is launched. To address this bottleneck, we investigate how to use generative AI to identify issues in a new learning experience *before* students engage with the material.

A digital learning activity generally combines a user interface with pedagogical content; if either is poorly designed or if they are not well integrated, the student experience suffers. Traditional heuristic evaluation methods typically assess these components in isolation, focusing on usability or content quality, yet this approach leads to an incomplete understanding of the student experience. We therefore pose the *Zero-Shot Learning Experience Evaluation Challenge*: to

evaluate digital learning experiences holistically – predicting student outcomes and extracting actionable insights – without testing on real students. We define evaluation criteria for meeting this challenge, propose solutions, and evaluate these solutions using the specified criteria.

After experimenting extensively with using multiple AI agents to simulate a distribution of students, we found a more effective and efficient method: a two-phase single-agent approach. In phase one, a web agent autonomously steps through the activity and produces a detailed description of the student experience. In phase two, the agent-generated description is used to infer student outcomes and provide feedback to the learning experience designer. For example, suppose an instructor created a new lesson with a reading, a video, and a guided exercise. The agent would navigate through each step and describe what the student is expected to do, what prior knowledge they should have, and where they might struggle. By working through the lesson in the same way a student would, the agent is able to capture a holistic view of the learning experience (Figure 1). The agent's output can then be leveraged to predict student outcomes and generate design recommendations for the instructor.

We evaluate several methods for predicting student outcomes in a massive, open-access, online CS1 course with 6,515 students from 149 countries and find that agent-generated descriptions significantly improve dropout prediction accuracy, outperforming all other approaches. We also conduct multiple case studies to test whether the agent can identify differences in lesson designs, explain their potential impact on students, and offer relevant suggestions. Our qualitative analysis confirms that the agent provides insightful recommendations to the course designers. While we focus on predicting student dropout and comparing lesson designs in CS1, we believe our method could be generalized beyond these use cases, as AI web agents are surprisingly effective for this purpose.

Main Contributions

1. We pose the **Zero-Shot Learning Experience Evaluation Challenge** and define the evaluation criteria. This challenge establishes a framework for automated design evaluation in educational settings.
2. We develop a **multi-modal LLM-based web agent** that autonomously works through online learning experiences much like a student would – continually processing the website content and interacting with the interface. By simulating the student experience, the agent is able to generate detailed insights that can be used to predict student outcomes and guide design.
3. We develop a **method for predicting student dropout** in a new learning experience. By prompting a large language model with agent-generated insights alongside transfer student data, we significantly improve dropout prediction accuracy, outperforming all other measured approaches.
4. We observe that using web agents to **simulate a distribution of students** is substantially more expensive and less effective at providing insights.
5. We use our method to generate **actionable insights** for real lessons in a massive, open-access, online CS1 course.

Our code is available at `github.com/sierrawang/lesson-design-agent.git`.

1.1 The Zero-Shot Learning Experience Evaluation Challenge

A novel opportunity in education is to use generative AI to help improve online learning – *without* testing the learning experience on real humans. In order to track progress on this important problem, we evaluate solutions based on their ability to (1) predict student outcomes and (2) provide actionable insights.

A method's ability to **predict student outcomes** is a falsifiable and quantifiable validation that the method is able to understand the learners' behavior. Among the many valid measures of outcomes, two informative metrics are the completion rate and dropout distribution of the new learning experience. The completion rate is the percentage of students who successfully complete the learning experience. The dropout distribution is the probability distribution over the steps of the learning experience that indicates where students—who start but do not finish—are most likely to drop out. If a model can predict where a student will drop out of a learning experience with high accuracy, we can both trust the model more and directly use that information to improve designs.

While student outcome prediction is a strong validation, we ultimately care about a method's ability to **provide actionable insights**. Can a method observe different versions of a learning experience and generate useful recommendations to the designer? This can be tested through case studies in which learning experiences are deliberately altered, with qualitative analyses to determine whether the method provides relevant feedback.

1.2 Related Work

Learning experience design is a "human-centric, theoretically-grounded, and socio-culturally sensitive approach to learning design, intended to propel learners towards identified learning goals, and informed by UXD" [28]. Developing effective learning experiences necessitates holistic evaluation methods that prioritize student outcomes [7, 8, 12, 21]. There has been extensive work on how to effectively assess learning experiences [34], including interactive design processes [23], heuristic and user testing [30], and AI based methods [20]. In parallel, researchers have developed models of student behavior [19, 32] using techniques such as knowledge tracing [26] and dropout prediction [9, 14]. More recently, large language models (LLMs) have been employed to simulate human behavior [1, 2, 24, 25], test educational tools [11], and even train teachers [13, 18]. LLMs are also being used to accelerate educational content development, particularly through content generation [5, 27, 33]. To the best of our knowledge this paper is the first work to evaluate the utility of an LLM web *agent* for learning experience design.

Multi-modal LLMs have further expanded possibilities for AI-assisted digital design processes. For example, UIClip leverages multi-modal capabilities to evaluate UI quality [35], botroast.io provides automated landing page feedback [3], and others use LLMs for rapid UI prototyping [6, 16, 37]. This technology has also accelerated the development of LLM-based web agents, which can autonomously navigate websites and execute tasks [4, 15, 36, 38]. These agents rely on LLMs to

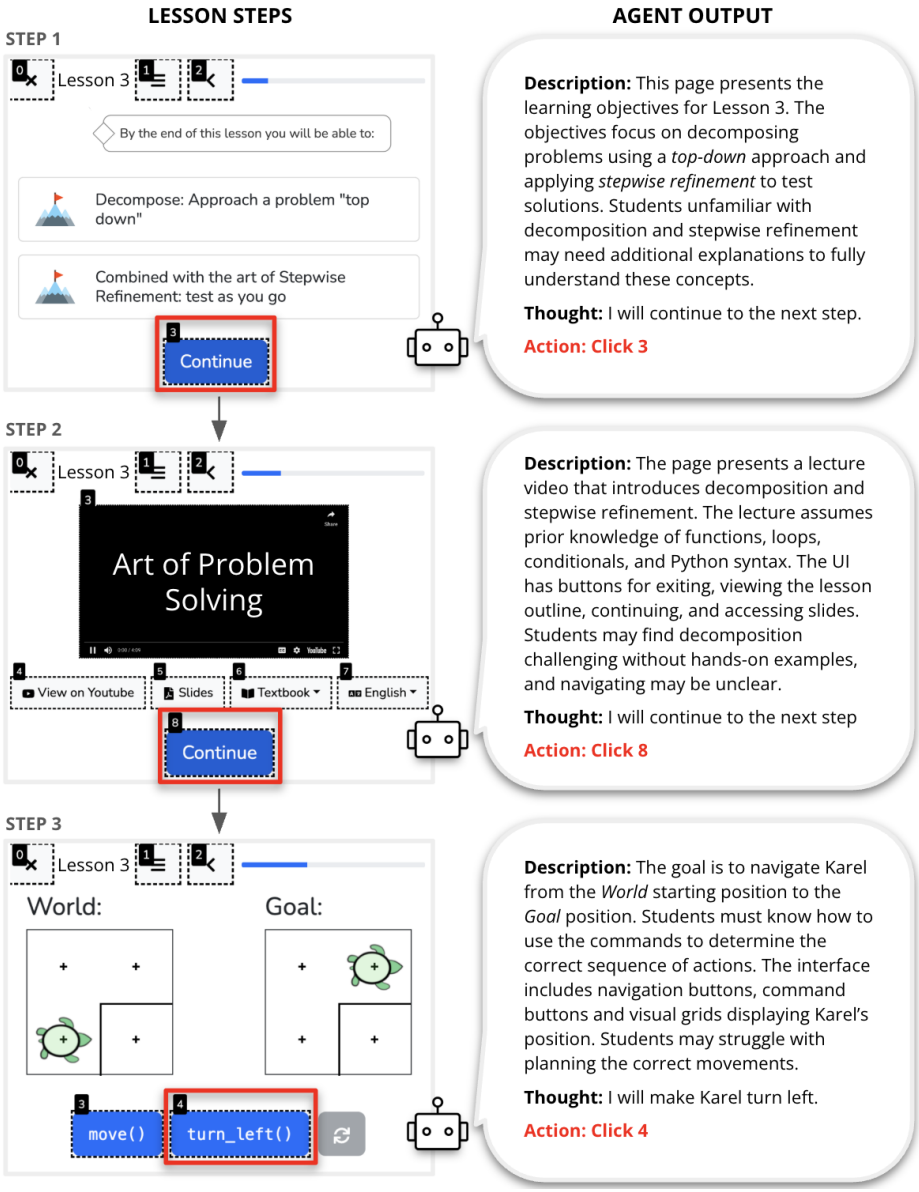


Fig. 1: Illustration of the web agent progressing through three steps in a lesson. At each step, the agent generates a description of the current webpage, a thought process for its next action, and an action to navigate to the next lesson step.

process web page information and choose actions (e.g., clicking the Next button), and web automation tools to interact directly with the browser (e.g., Selenium [29]). Our work builds on WebVoyager [10], a web agent designed for reliable task execution on real-world websites. WebVoyager enhances the LLM processing abilities by drawing bounding boxes around interactive elements on each web page, enabling the model to better interpret visual information and make effective decisions. Our agent extends this infrastructure to effectively engage with online learning experiences.

2 Web Agents for Learning Experience Design

We developed a **multi-modal LLM-based web agent** to evaluate new learning experiences. The agent autonomously navigates a learning experience like a student would, continuously processing the website content and interacting with the interface. By directly engaging with the experience in this way, the agent is able to generate detailed descriptions of the student experience that can be used to predict student outcomes and guide design. For example, suppose an instructor designs a new lesson consisting of three steps: (1) an outline, (2) a lecture video, and (3) a guided exercise. The agent navigates to each step in order and answers the following questions:

1. What information is presented? What are the concepts being discussed?
2. What prior knowledge is required to be able to complete this step?
3. How is the content delivered? What is the student expected to do?
4. What does the web page look like? What are the major UI elements?
5. Where might students struggle?

Figure 1 illustrates the agent progressing through three steps of a lesson. The agent proceeds through the lesson according the logic in Algorithm 1. Our agent is an adaptation of the WebVoyager web agent [10] and uses Selenium [29] to interact with the browser.

Algorithm 1 Web Agent Logic

while lesson is not complete:
  step 1: Capture the current web page content including:
    · A screenshot of the web page
    · A text description of the interactive elements (e.g., "Next" button)
    · The audio transcript (if applicable)
  step 2: Send the web page content to GPT-4o [22] to generate:
    · A detailed description of the current web page
    · A thought process (e.g., "I want to click the Next button because...")
    · The next agent action (e.g., "click Next")
  step 3: Execute the specified action on the web page.
end while

3 Experimental Setup

We evaluate the web agent by assessing its effectiveness in predicting student outcomes and providing meaningful feedback to learning experience designers. In this section, we describe our experiment setup and methodology.

3.1 Evaluation Setting: A global CS1 course

We focus on predicting the student outcomes of Code in Place, an open-access online CS1 course that enrolled 6,515 students from 149 countries [31]. In this course, students learn about new programming concepts through multi-step lessons that consist of lecture videos and guided exercises. We predict two key metrics for each lesson:

1. **Completion rate**: the percent of the students who start the lesson that complete it.
2. **Dropout distribution**: the probability distribution over the lesson steps that indicates where students are most likely to drop out. A student's *dropout step* is defined as the farthest step they reached in the lesson. To start a lesson, a student must reach at least the second step.

3.2 Methods of Predicting Student Outcomes

We evaluate three methods for predicting student outcomes on a lesson: two baseline approaches and one LLM-based method. For the LLM-based method, we conduct an ablation study, which involves systematically removing components of the prompt to assess the impact of the agent-generated lesson descriptions.

We evaluate each method by comparing its predicted outcomes to the actual student outcomes. In accordance with the "Zero-Shot Learning Experience Evaluation Challenge" (Section 1.1), a method can leverage student data from previous lessons in the course (transfer data), as well as lesson details (e.g. title, steps) to make its prediction. The three methods are as follows:

Baseline Method: Sample-Based Prediction
This method uses the real student outcomes from the preceding (sample) lesson as the prediction for the target lesson.

To predict the **completion rate** for the target lesson, we directly use the observed completion rate of the sample lesson.

To predict the **dropout distribution** for the target lesson, we:

1. Scale the step indices of the sample and target lessons to a common range, ensuring that the lessons' steps are aligned based on their relative positions.
2. Apply linear interpolation to the sample distribution and use this to estimate the dropout probability at each step of the target lesson.
3. Normalize all probabilities to sum to 1.

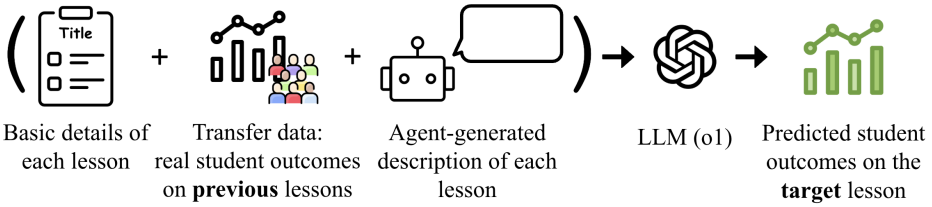


Fig. 2: The agent-based method for predicting student outcomes. This approach leverages the basic details of each lesson, real student data from the previous lessons, and agent-generated descriptions of each lesson to prompt the LLM (o1) to predict student outcomes on the target lesson.

Baseline Method: Regression-Based Prediction
This method uses regression modeling of the real student outcomes on previous lessons to predict student outcomes on the target lesson.

To predict the **completion rate**, we fit a line to the indices and completion rates of all previous lessons. We then predict the completion rate of the target lesson as the next possible index.

To predict the **dropout distribution**, we:

1. Scale the step indices of all lessons to a common range, ensuring that the lessons' steps are aligned based on their relative positions.
2. Record the dropout probability at each step index of each previous lesson and use this as training data.
3. Fit a polynomial regression model to the training data, with a degree equal to the number of steps in the target lesson.
4. Use the regression model to predict the dropout probability for each step index of the target lesson (ensuring all values remain non-negative).
5. Normalize all probabilities to sum to 1.

LLM-Based Method
This method leverages OpenAI's LLM, o1, to predict student outcomes [22]. We experiment with three pieces of information as context in the LLM prompt: basic lesson details (title and steps), student outcomes on previous lessons in the course (transfer data), and agent-generated descriptions of each lesson. The LLM is prompted to identify patterns in student behavior in previous, similar scenarios and use these insights in predicting outcomes for the target lesson. This method is illustrated in Figure 2.

To evaluate the contribution of each component of the prompt, we conduct an ablation study with four variations:

1. **Target Lesson Details**: The prompt only includes the title and steps of the target lesson and must make predictions without additional context.
2. **Previous Lesson Details**: In addition to (1), the prompt includes the title and steps of previous lessons, providing more course context.

3. **Transfer Data**: In addition to (2), the prompt includes the real completion rates and dropout distributions of the previous lessons, allowing the model to identify patterns in student behavior in previous, similar scenarios.
4. **Agent-Generated Descriptions**: In addition to (3), the prompt includes agent-generated lesson descriptions for the previous and target lessons. This final setup evaluates whether the agent's output improves the model's ability to predict student outcomes.

3.3 Evaluating Predictions

We evaluate each method's ability to predict the **dropout distribution** across 11 lessons in the course. These encompass all lessons in the course except those in which the prediction would be trivial, either because they are required materials or have too few steps. For each lesson, we compare the predicted dropout distribution to the true dropout distribution using Jensen–Shannon Divergence (JSD) [17]. To summarize each method's performance, we calculate the average JSD across all 11 lessons. For LLM-based methods, we repeat each prediction – including using the agent to generate descriptions – 30 times, and we report the mean and standard error of the mean across these runs.

We evaluate each method's ability to predict the **completion rate** across 14 lessons in the course. These encompass all lessons in the course except the application materials since they were required to enter the course. To summarize each method's performance, we compute the Root Mean Square Error (RMSE) across all 14 lessons. For LLM-based methods, each prediction – including using the agent to generate descriptions – was repeated 30 times, and we report the mean and standard error of the mean across these runs.

4 Results: Predicting Student Outcomes

4.1 Predicting Dropout

We find that the LLM-based prediction improves as more context is added to the prompt, with **the best results achieved when agent-generated descriptions are included**. Specifically, the agent-based approach reduces the error to 0.060 ± 0.003, a significant improvement over all other methods ($p = 0.000$). This suggests that the agent descriptions provide meaningful insights about the lesson that are not reflected in the basic lesson details and transfer data alone.

When the model is only given the target lesson details, its performance is similar to the Sample baseline (0.116 ± 0.003). Adding previous lesson details has minimal impact (0.115 ± 0.004), indicating that lesson structure alone does not significantly enhance predictions. Incorporating transfer data leads to a notable improvement (0.100 ± 0.002), demonstrating the importance of providing examples of student behavior to the LLM. Among the baseline methods, the Sample approach (0.114) beats the Regression model (0.176), suggesting that distribution sampling is more effective than regression modeling for this task.

Dropout Prediction Method	Error (↓)
Baseline Methods	
Sample	0.114
Regression	0.176
LLM-based Methods	
Target Lesson	0.116 ± 0.003
+ Previous Lesson	0.115 ± 0.004
+ Transfer Data	0.100 ± 0.002
+ **Agent**	**0.060 ± 0.003**

Table 1: Dropout prediction error for each method, measured as the average JSD [17] between the true and predicted dropout distributions. The agent-based method achieves the lowest error, significantly outperforming all other methods ($p = 0.000$).

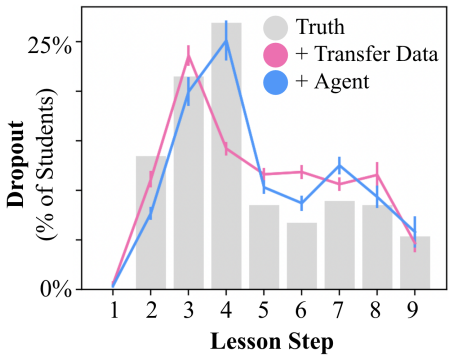


Fig. 3: Example dropout distribution for a lesson. The gray bars show the true dropout distribution. The blue and pink curves are predicted dropout distributions. The agent-based prediction (blue) more closely aligns with the true distribution.

We also conducted an ablation study to test the importance of each question in the agent prompt (e.g. "What information is presented?", "What prior knowledge is required?", etc.). We found that removing any detail nearly doubled the error, underscoring the value of each detail.

Table 1 summarizes the results across all dropout prediction methods. Figure 3 shows the true dropout distribution and two predictions for Lesson 5, illustrating how the agent descriptions improve prediction accuracy.

4.2 Predicting Completion Rates

Unlike dropout prediction, where agent descriptions led to the best accuracy, their impact on predicting completion rates is limited. Among the LLM-based methods, incorporating agent-generated descriptions improves performance, reducing the error from 0.055 ± 0.001 to 0.051 ± 0.001; however, we find that the regression model achieves the lowest error, 0.048, of all methods. This suggests that the agents are not capturing key factors that influence lesson completion. The Sample baseline has the highest error of 0.062.

4.3 Multi-Web Agent Student Simulation

We also experimented with using multiple web agents to directly simulate students engaging with a new learning experience, and using their results to gain meaningful insights into the experience design. The key difference between simulating students and our proposed method lies in the task of the LLM. Rather than simply *describe* a student's experience at each step (proposed method), the LLM has to *choose actions* that authentically reflect student behavior.

We explored this approach extensively, attempting to simulate meaningful distributions of student behavior. Each simulated student (agent) was given a real student persona that included how the student performed on previous tasks and instructed to act like this student while they do a new target task. We sampled these personas according to the predicted completion rate of the target task; if we predicted a 50% completion rate, we sampled personas so that half were students who had completed the prior task, and the other half had not.

Despite these efforts, we found that our simulated students exhibited a far more limited range of behavior than real students. For example, when we ran 100 agents on five test lessons, on average, they simulated only 4% of the paths that real students took. As a result, they failed to provide substantial insights into the difficulty of a new learning experience. This approach was also significantly more computationally intensive than our proposed method, as it required simulating an entire distribution of student behaviors rather than using a single agent to describe the experience.

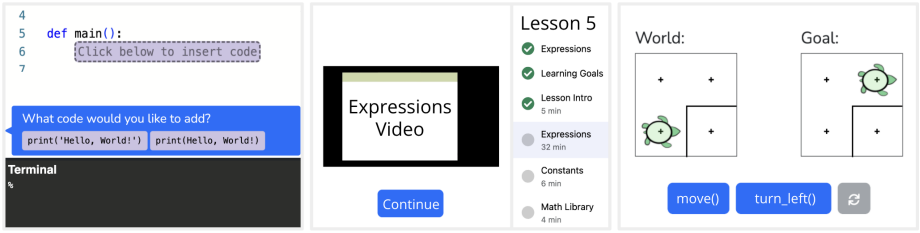
5 Results: Agent Insights for Design

Through four case studies, we tested how lesson designers could use the web agent to evaluate potential lesson designs. In each case, we used the agent to generate descriptions of different versions of a lesson, then used a language model (o1) to compare the descriptions and make a recommendation [22]. We qualitatively analyzed the model's output to assess whether it identifies the design differences, explains their impact on students, and offers relevant suggestions; we found the method surprisingly effective. Figure 4 illustrates some of the lesson designs tested in case studies 2–4.

Case Study 1: Should we include a video on for-loops?
Lesson 2: Control Flow introduces for-loops, while-loops, and if-statements. We created two versions of the lesson: Version 1 includes a lecture video explaining for-loops, while Version 2 omits the video and starts directly with a code example. The model recommended keeping the video, stating:

> The primary difference lies in the inclusion of a dedicated video lecture on for-loops in Version 1 that is omitted in Version 2... This lecture in Version 1 provides theoretical background, explanations, and practical examples that are crucial for understanding for-loops... The omission of the for-loops lecture reduces the lesson's duration but at the potential cost of student comprehension... Students in Version 2 might struggle with the code examples involving for-loops due to the lack of a formal introduction to the concept. For novices, jumping straight into code without understanding the underlying principles can lead to confusion and hinder learning.

Case Study 2: How should we introduce the terminal?
In a prior version of *Lesson 4: Introduction to Python*, students learned about the terminal, printing, and input through three lecture videos and one code example.



Case Study 2 **Case Study 3** **Case Study 4**

Fig. 4: Lesson designs from case studies 2–4. Case Study 2 shows an interactive code example. Case Study 3 displays the lesson interface with the outline visible. Case Study 4 presents a Karel exercise where Karel is shown as a turtle.

This version took about 71 minutes to complete and had a high dropout rate. To improve the experience, the instructor redesigned the lesson to consist of four shorter videos and two interactive code examples. In these new examples, students fill in parts of the code before running it. The model preferred the new version, explaining:

> The inclusion of interactive exercises where students actively participate in coding enhances comprehension and retention of programming concepts. Interactive learning allows students to apply knowledge immediately, receive instant feedback, and correct misunderstandings in real-time. Therefore, incorporating the interactive elements and transitional content from Version 2 would be more beneficial to students, fostering a deeper understanding and a more immersive learning experience.

Case Study 3: Should the lesson outline always be visible?
In the course platform, the lesson outline is hidden by default and students must click a button to view it. The lesson designer wondered whether displaying the outline might help students stay motivated. We created two versions of *Lesson 5: Expressions* – Version 1 keeps the outline hidden and Version 2 displays the outline in a sidebar throughout the lesson – and used the web agent for an initial comparison. The model preferred the outline to be visible, noting:

> The instructional content remains consistent between the two versions... the primary differences lie in the user interface... Adopt the sidebar from Version 2 that displays the lesson outline with progress indicators and estimated times. This feature helps students visualize their learning journey, track their progress, and manage their study time more efficiently.

Case Study 4: Which Karel graphic is the most clear?
In this case study, we evaluate whether the agent can analyze the pedagogical impact of visual variations in *Karel*, a small graphical figure that moves around a grid by following simple commands. In the course, Karel is used to introduce

key programming concepts. We used the agent to compare three different depictions of Karel – a robot, a turtle, and an arrow. We created three versions of *Lesson 3: Art of Problem Solving* which includes multiple Karel exercises, one with each depiction of Karel. Notably, the web agent only successfully completed the lesson when Karel was depicted as an arrow. This suggests that the arrow representation provides the most intuitive user experience, likely because it directly conveys directionality, making navigation tasks more clear.

6 Discussion

Limitations. The most significant limitation of our methodology is the potential bias in both the LLM's training data and the transfer data. If this data is skewed toward specific student behaviors, our method may inadvertently favor certain student groups while neglecting others. This challenge is not unique to our approach; it is a fundamental concern in experience testing, highlighting the need for efforts to mitigate these biases.

The current technical limitations of our method include the model's context window and the need for an intuitive UI for the web agent to navigate effectively. We do not see these as long-term barriers as ongoing advancements in LLMs and web agent technology are likely to address them.

Future Work. In this work, we show how an LLM-based web agent can be used to effectively predict student outcomes and provide design feedback for lessons in a massive, global CS1 course. Although we specifically focused on a few controlled case studies of lesson design modifications, we believe that our approach can extend well beyond these initial scenarios. Future research should apply this method to a wider range of learning experiences and work closely with instructional designers to enhance its practical impact. Moreover, developing a benchmark suite of diverse, real-world cases would be invaluable for developing robust web agents that can reliably offer meaningful insights into any online learning experience.

In our research, we were surprised to find that a single web agent was more effective for learning design insights than a collection of web agents that simulate a distribution of students. Simulating realistic student behavior remains a compelling direction for future work. To support future research, our code is available at `github.com/sierrawang/lesson-design-agent.git`.

7 Conclusion

We introduce a novel approach for evaluating learning experience design with LLM-based web agents. Through rigorous testing and qualitative analysis, we demonstrate that agent-generated insights significantly enhance student dropout prediction and provide valuable feedback for improving lesson designs. Our findings establish web agents as an effective tool for evaluating learning experiences and we believe this method has the potential to accelerate meaningful advancements in online learning.

References

1. Aher, G.V., Arriaga, R.I., Kalai, A.T.: Using large language models to simulate multiple humans and replicate human subject studies. In: International Conference on Machine Learning. pp. 337–371. PMLR (2023)
2. Argyle, L.P., Busby, E.C., Fulda, N., Gubler, J.R., Rytting, C., Wingate, D.: Out of one, many: Using language models to simulate human samples. Political Analysis **31**(3), 337–351 (2023)
3. BotRoast: BotRoast - Design feedback in 1 click. (2025), `https://botroast.io/`, accessed: 2025-02-17
4. Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., Su, Y.: Mind2web: Towards a generalist agent for the web. Advances in Neural Information Processing Systems **36** (2024)
5. Diwan, C., Srinivasa, S., Suri, G., Agarwal, S., Ram, P.: Ai-based learning content generation and learning pathway augmentation to increase learner engagement. Computers and Education: Artificial Intelligence **4**, 100110 (2023)
6. Duan, P., Chen, C.Y., Hartmann, B., Li, Y.: Visual prompting with iterative refinement for design critique generation. arXiv preprint arXiv:2412.16829 (2024)
7. Floor, N.: This is Learning Experience Design: What it is, how it works, and why it matters. New Riders (2023)
8. Fournier, H., Kop, R.: Mooc learning experience design: Issues and challenges. International journal on E-Learning **14**(3), 289–304 (2015)
9. Halawa, S., Greene, D., Mitchell, J.: Dropout prediction in moocs using learner activity features. elearning papers, 37, 1-10 (2014)
10. He, H., Yao, W., Ma, K., Yu, W., Dai, Y., Zhang, H., Lan, Z., Yu, D.: Webvoyager: Building an end-to-end web agent with large multimodal models (2024), `https://arxiv.org/abs/2401.13919`
11. He-Yueya, J., Goodman, N.D., Brunskill, E.: Evaluating and optimizing educational content with large language model judgments. arXiv preprint arXiv:2403.02795 (2024)
12. Jahnke, I., S.M.E.Y., Tawfik, A.: Theoretical considerations of learning experience design. Theories to influence the future of learning design and technology **1** (2022)
13. Jin, H., Lee, S., Shin, H., Kim, J.: Teach ai how to code: Using large language models as teachable agents for programming education. In: Proceedings of the CHI Conference on Human Factors in Computing Systems. pp. 1–28 (2024)
14. Kim, J., Guo, P.J., Seaton, D.T., Mitros, P., Gajos, K.Z., Miller, R.C.: Understanding in-video dropouts and interaction peaks inonline lecture videos. In: Proceedings of the first ACM conference on Learning@ scale conference. pp. 31–40 (2014)
15. Koh, J.Y., Lo, R., Jang, L., Duvvur, V., Lim, M.C., Huang, P.Y., Neubig, G., Zhou, S., Salakhutdinov, R., Fried, D.: Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. arXiv preprint arXiv:2401.13649 (2024)
16. Kolthoff, K., Kretzer, F., Fiebig, L., Bartelt, C., Maedche, A., Ponzetto, S.P.: Zero-shot prompting approaches for llm-based graphical user interface generation. arXiv preprint arXiv:2412.11328 (2024)
17. Lin, J.: Divergence measures based on the shannon entropy. IEEE Transactions on Information theory **37**(1), 145–151 (1991)
18. Markel, J.M., Opferman, S.G., Landay, J.A., Piech, C.: Gpteach: Interactive ta training with gpt-based students. In: Proceedings of the tenth acm conference on learning@ scale. pp. 226–236 (2023)

19. Matsuda, N., Cohen, W.W., Sewall, J., Lacerda, G., Koedinger, K.R.: Evaluating a simulated student using real students data for training and testing. In: International Conference on User Modeling. pp. 107–116. Springer (2007)
20. Moore, S., Costello, E., Nguyen, H.A., Stamper, J.: An automatic question usability evaluation toolkit. In: International Conference on Artificial Intelligence in Education. pp. 31–46. Springer (2024)
21. Nakakoji, K., Yamada, K., Yamamoto, Y., Morita, M.: A conceptual framework for learning experience design. In: First Conference on Creating, Connecting and Collaborating Through Computing. pp. 76–83. IEEE (2003)
22. OpenAI: Openai model documentation (2024), `https://platform.openai.com/docs/models`, accessed: 2024-02-04
23. Park, J.Y.: iled: interactive learning experience design. Journal of Online Learning and Teaching **4**(3), 357–370 (2008)
24. Park, J.S., O'Brien, J., Cai, C.J., Morris, M.R., Liang, P., Bernstein, M.S.: Generative agents: Interactive simulacra of human behavior. In: Proceedings of the 36th annual acm symposium on user interface software and technology. pp. 1–22 (2023)
25. Persdre: Awesome llm human simulation. `https://github.com/Persdre/awesome-llm-human-simulation` (2025), accessed: 2025-02-18
26. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. Advances in neural information processing systems **28** (2015)
27. Sarsa, S., Denny, P., Hellas, A., Leinonen, J.: Automatic generation of programming exercises and code explanations using large language models. In: ACM Conference on International Computing Education Research. pp. 27–43 (2022)
28. Schmidt, M., Huang, R.: Defining learning experience design: Voices from the field of learning design & technology. TechTrends **66**(2), 141–158 (2022)
29. Selenium: Selenium: Web browser automation (2025), `https://www.selenium.dev/`, accessed: 2025-02-07
30. Tan, W.s., Liu, D., Bishu, R.: Web evaluation: Heuristic evaluation vs. user testing. International Journal of Industrial Ergonomics **39**(4), 621–627 (2009)
31. University, S.: Code in place 2024 (2025), `https://codeinplace.stanford.edu/cip4/studenthome`, accessed: 2025-04-28
32. VanLehn, K., Ohlsson, S., Nason, R.: Applications of simulated students: An exploration. Journal of artificial intelligence in education **5**, 135–135 (1994)
33. Wang, Z., Valdez, J., Basu Mallick, D., Baraniuk, R.G.: Towards human-like educational question generation with large language models. In: International conference on artificial intelligence in education. pp. 153–166. Springer (2022)
34. Williams, D.D., South, J.B., Yanchar, S.C., Wilson, B.G., Allen, S.: How do instructional designers evaluate? a qualitative study of evaluation in practice. Educational Technology Research and Development **59**, 885–907 (2011)
35. Wu, J., Peng, Y., Li, X., Swearngin, A., Bigham, J., Nichols, J.: Uiclip: a data-driven model for assessing user interface design. In: Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology. pp. 1–16 (2024)
36. Yao, S., Chen, H., Yang, J., Narasimhan, K.: Webshop: Towards scalable real-world web interaction with grounded language agents. Advances in Neural Information Processing Systems **35**, 20744–20757 (2022)
37. Yuan, M., Chen, J., Quigley, A.: Maxprototyper: A multi-agent generation system for interactive user interface prototyping. arXiv preprint arXiv:2405.07131 (2024)
38. Zheng, B., Gou, B., Kil, J., Sun, H., Su, Y.: Gpt-4v (ision) is a generalist web agent, if grounded. arXiv preprint arXiv:2401.01614 (2024)