

Can GPT Help? Supporting Teachers to Brainstorm Customized Instructional Scratch Projects

Minh Tran
ngminhtran@uchicago.edu
University of Chicago
Chicago, IL, USA

Elaine Zhou
ezhou@uchicago.edu
University of Chicago
Chicago, IL, USA

David Gonzalez-Maldonado
dagm@uchicago.edu
University of Chicago
Chicago, IL, USA

Diana Franklin
dmfranklin@uchicago.edu
University of Chicago
Chicago, IL, USA

Abstract

While many recent studies have explored how large language models can transform computer science instruction from the instructor perspective, they are primarily at the college level. Thus, little is known about using large language models towards curriculum development and teacher supports outside of the college setting. Given the emphasis placed on culturally responsive teaching at the K-8 level and well-documented evidence of insensitive and inaccurate language model outputs from a cultural perspective, it is imperative to perform systematic and principled research before considering their use in this setting.

This paper explores the potential of teachers using large language models to brainstorm instructional Scratch projects. Specifically, we use GPT-3 to mimic structured projects from an existing computer science curriculum but situate the generated projects in different contexts/themes. We qualitatively analyze 300 project ideas generated by GPT and find 81% of the generated ideas satisfy our metrics for technical alignment and theme quality. We identify two major weaknesses: code complexity of generated projects and presence of potential insensitive elements that would require human filtering. We conclude that, while not ready as a student-facing solution, teachers could use GPT to effectively brainstorm customized instructional materials.

CCS Concepts

• **Social and professional topics** → **K-12 education; Computer science education.**

Keywords

K-8, Scratch programming, curriculum customization, teacher supports, large language models

ACM Reference Format:

Minh Tran, David Gonzalez-Maldonado, Elaine Zhou, and Diana Franklin. 2025. Can GPT Help? Supporting Teachers to Brainstorm Customized Instructional Scratch Projects. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025)*, February 26-March 1, 2025, Pittsburgh, PA, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3641554.3701858>

1 Introduction

In K-8 (5-13 y.o.), culturally responsive teaching [13, 25] and instructional materials that advance equity have been increasingly developed and adopted [4, 11, 16, 22]. As experts on their own students, teachers are best positioned to tailor culturally responsive materials to their classrooms. However, this process is time-consuming and challenging because teachers need to create materials that both address the intended learning goals and are situated in contexts that resonate with the lived experiences of their students [32].

The emergence of large language models (LLMs) has the potential to reshape many aspects of computer science (CS) instruction, *from the instructor perspective*. Prior work has explored using LLMs in the creation of high-quality, college-level learning materials at scale (e.g., programming exercises and code explanations [29], worked examples [20]) or on demand by students (e.g., personalized Parsons puzzles [17], contextually personalized programming exercises [8, 26]). At the K-8 level, where it is critical to provide culturally responsive and engaging learning experiences [7, 9, 30], little is known about using LLMs towards curriculum development and teacher supports.

This study studies the potential of using GPT-3 to *assist a teacher* in brainstorming customized Scratch projects that their students would use to explore the code related to a specific CS concept. Specifically, we prompt GPT-3 with a simple description of a Scratch project from an intermediate CS curriculum for upper-elementary and middle school classrooms and ask it to suggest new project ideas that cover the same technical content but are placed in a different context. This paper presents an analysis of 300 GPT-generated project ideas, guided by the following questions:

RQ1: What are the strengths and weaknesses of GPT in generating customized, structured Scratch project ideas?

RQ2: To what extent can teachers use GPT to efficiently brainstorm ideas for customized instructional Scratch projects?



This work is licensed under a Creative Commons Attribution International 4.0 License.

SIGCSE TS 2025, February 26-March 1, 2025, Pittsburgh, PA, USA
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0531-1/25/02
<https://doi.org/10.1145/3641554.3701858>

2 Background and Related Work

2.1 Theoretical Framing

This work is inspired by culturally responsive teaching pedagogy (CRT), the spheres of influence framework (Sol), and the zone of proximal development theory (ZPD).

CRT emphasizes the incorporation of students’ cultural knowledge and experiences into their learning [13, 25, 30]. In CS, there are two common approaches to CRT: open-ended projects [12, 21, 27] and curriculum designed for a particular population [11, 18, 19, 23, 28]. Our work aims to support teachers in creating CRT materials tailored to their classrooms. Sol describes the connection between students’ career aspiration and four different spheres: home/family, school, hobbies/leisure, and TV [1]. Through participatory design sessions with middle school students, Coenraad et al. expanded the original Sol to include home/family, hobbies/leisure, school/work, media, interest, peers, and identity [6]. The expanded Sol informs our selection of student-relevant topics used in our GPT inquiries. ZPD implies there is a beneficial range of difficulty that requires academic growths [33], but is still within the learner’s reach [5]. Our goal is to keep students in their ZPD by ensuring the customized materials are of the right difficulty. Consequently, we evaluate the generated projects for code complexity in relation to the technical attributes of the base curriculum’s projects.

2.2 Customizing Instructional Materials

Developing teaching materials can be laborious and difficult, even for experienced teachers. For this reason, many fully-featured CS materials have been developed by prominent organizations and research groups [4, 11, 16, 22]. However, teachers may want to create customized materials for many reasons, including the constraints of their teaching schedules and the needs of their students [2]. Customizing available resources is common practice for teachers in many subjects [3], including CS [10]. Unfortunately, customization requires teachers to spend a significant amount of time and have a solid understanding of the existing curriculum’s structure [31].

In a previous study, we explored challenges CS teachers face when attempting to customize modules from the Scratch Encore curriculum [11], and found that teachers tend to add complexity to the materials and/or spend an excessive amount of time polishing the visual elements of the materials [32].

2.3 LLMs for Class Preparation

With the emergence of LLMs, CS education researchers have studied *how LLMs can support college-level instructors in class preparation*. One way instructors can leverage LLMs is in the creation of high-quality instructional materials at scale. Sarsa et al. examined programming assignments automatically generated by Codex. They found the generated exercises, sample solutions, test cases, and code explanations sensible and novel, with many being ready to use with minimal modifications [29]. Other researchers explored using LLMs to reduce the workload and time required by instructors to create a variety of worked examples in introductory programming courses [20]. They developed an LLM-powered tool to generate worked examples, and proved the quality and effectiveness of the generated materials from both the expert and student perspectives.

Another application of LLMs from the instructor perspective is to create learning materials on demand by students. Hou et al. introduced a system that uses LLMs to create real-time, personalized Parson puzzles for struggling students [17]. Their system could reliably produce correct, personalized, and concise puzzles, and the generated puzzles was well-perceived by students. A study by Gutierrez et al. demonstrated that LLMs are capable of creating programming exercises with a variety of contextualized problem descriptions [8]. More recent work by Logacheva et al. performed a systematic evaluation of LLM-generated, contextually personalized programming exercises, from both the expert and student perspectives [26]. While students generally found the generated exercises engaging and useful, expert evaluation showed that they are often easier than requested and shallow in terms of theme.

Our work *builds upon prior research on instructors’ use of LLMs toward creating and enhancing CS instructional materials*, and *expand it to K-8*, where efficiently creating high-quality, customized materials is an ongoing challenge for many teachers. One major difference between the K-8 and the college-level settings is the use of block-based rather than text-based projects. Existing work shows that LLMs have great difficulty generating block-based code [15], so we focus on generating project ideas that a teacher can easily evaluate and implement in a block-based programming language.

3 Methods

3.1 Base Project Selection

We selected base projects from Scratch Encore [11], an intermediate CS curriculum for upper elementary and middle school classrooms. It has 14 modules, each covering a specific CS concept. For each module, Scratch Encore provides three equivalent sample projects that only differ by theme; students observe and explore a sample project’s code before making small modifications. For example, the sample projects in the animation module are implemented using almost identical scripts, but with two different themes: Animal Race (animate two animal sprites across the screen) or Dragon Boat Race (animate two boat sprites across the screen).

A key challenge when teachers create a customized Scratch Encore project is brainstorming an idea relevant to students, and that, when implemented, retains the technical attributes of the base project in the current module (e.g., number of sprites and blocks, how sprites animate) [32]. Consider a teacher with students who celebrate Lunar New Year, remixing a *base project* including (1) a monkey standing in one place and waving a flag, (2) a bee flying and (3) a snake wiggling across the screen. A possible *customized project* could include (1) a child standing in one place and waving a sparkler, (2) a dragon and (3) a lion dancing across the screen. If the technical attributes are not followed, students may not have the knowledge necessary to move to the next module (if missing newly introduced code blocks) or they could be overwhelmed when exploring the code (if using advanced blocks or too many blocks). The technical/code mismatch could also make the customized projects incompatible with the auto-graders provided by Scratch Encore.

3.2 Project Idea Generation

We approached project idea generation from the perspective of a person with limited AI/LLM knowledge (e.g., a layperson or teacher).

With this in mind, we started from a very simple prompt consisting of three parts: (1) overall project requirements for a module, (2) description of an existing project, and (3) a question asking GPT to suggest three different projects that fit the project requirements and are related to a specific topic. The first author interacted with the model, iteratively refining the prompt (varying length, format, level of detail, and vocabulary), and discussed improvements with other authors. After a number of iterations, we performed a larger-scale, systematic evaluation of the generated outputs. This process is discussed in Section 3.3. Due to space limitations, we do not include prompt design detail in this paper. Rather than focusing on prompt engineering to obtain the best outputs, our goal is to investigate GPT performance on a simple prompt because it is more likely to align with how teachers/non-expert AI users interact with LLMs.

We chose base projects from five modules of Scratch Encore: events, animation, conditional loops (CondLoop), decomposition by sequence (SeqDecomp), and one-way synchronization (1WaySync). In selecting a model, we used the free version of GPT-3 available on OpenAI Playground, as it was the most accessible LLM at the time of our study. Figure 1 shows the prompt template we used to generate customized project ideas for the animation module.

In all GPT inquiries, the project requirements and the base project description were fixed for each module; only the topic was different. We used 20 similar topics to generate project ideas for each module. In total, we obtained 300 project ideas for evaluation. Topics were drawn from the expanded SoI [6] and through the CRT lens [13, 25], representing a variety of strands: *youth culture* (hobbies/leisure, media, and interests spheres), *local community culture* (home/family sphere, focusing on local events), *family culture* (home/family sphere, focusing on food and holidays/traditions), and *historical events for URM*s (CRT lens, focusing on social justice [30]). Topics we used for idea generation are:

- (1) **Youth culture (Yth Cult.):** Baseball, Mario Kart, Harry Potter, Environmental Awareness
- (2) **Local community culture (Local Cult.):** Chicago Air and Water Show, Albuquerque International Balloon Fiesta, San Francisco Pride, Aloha Festivals
- (3) **Family culture - Food:** Street food, Food Festival, Chinese food, Mexican food
- (4) **Family culture - Holidays/traditions (Hod./Trad.):** Juneteenth, Indigenous Peoples' Day, Day of the Dead, Lunar New Year
- (5) **Historical events for URM**s (Hist. Ev.): Women's March, Harlem Renaissance, Battle of the Little Bighorn, #MeToo

3.3 Project Idea Evaluation

We qualitatively evaluate each project idea, using seven metrics in Table 1 with a 1 (“yes”) or 0 (“no”) response. The metrics are designed to address both the quality of the theme (T1-T3) and alignment to the base project’s code (C1-C4). For theme quality, we analyze whether an idea is age appropriate (T1 – no sex, drugs, alcohol, or violence), whether it matches the desired theme (T2), and whether it is coherent (T3 – e.g., a soccer ball being kicked through a donut on a boat would be incoherent). Code alignment involves several factors. We analyze whether the sprites have the

Figure 1: Prompt template to generate animation projects

same functionality as in the base project (C1). An example of non-alignment could be a base project with two cats running across the screen from left to right but the generated project has two cats chasing each other around a box. We then evaluate whether an idea is implementable in Scratch (C2), and whether it could be implemented with a nearly identical script to the base project’s code (C3). If the script is different, we analyze the relative code complexity (C4) – whether it requires more sprites/scripts/blocks.

Finally, we compare the three generated projects in each GPT query for variability (V – how different the three projects are from each other) and consistency (C – how many projects out of three are good). A project idea triplet is considered *good* if it has no complexity issue (“no” response to all C4 questions), and all other metrics score 1 (“yes” response). A score of 1 for variability means “three projects are clearly different”, while 0 means “two or more projects are very similar”. Consistency is tagged 1 if “two or more projects are good”, and 0 if “none or only one is good”.

Metric	Description
T1: Age approp. (AA)	Appropriate for K-8?
T2: Topic alignment (TA)	Aligns with the given topic?
T3: Reasonability (R)	Idea coherent?
C1: Code intent (CI)	Sprites function similarly as in base project?
C2: Code feasibility (CF)	Implementable in Scratch?
C3: Code similarity (CS)	Scripts match with base project?
C4: Code complexity (CC)	Has more sprites, scripts, blocks?

Table 1: Rubric to evaluate GPT-generated project ideas

Process. Every week, the first three listed authors coded the same subsets of project ideas, and met to resolve differences and clarified the description of evaluation metrics as needed. This process was iterated until they reached 0.85–1.00 Fleiss’ Kappa [14] on all metrics. They then independently coded a separate subset of remaining project ideas, and met weekly to discuss any lingering uncertainties.

Topic	Project Idea				
Food Festival	Backdrop: A food festival celebration with a diverse array of festival-goers Sprites: A chef, two waiters – The chef slices vegetables with a single knife – Two waiters carry trays of dishes across the screen				
T1:AA	T2:TA	T3:R	C1:CI	C2:CF	C3:CS
Yes	Yes	Yes	Yes	Yes	Yes

Topic	Project Idea (Generated)	Project Idea (Modified)			
Aloha Festival	Backdrop: A tropical beach with coconut trees and a grass hut Sprites: Hula hoop, tiki totem, flower headband – The hula hoop moves around in a circle – The tiki totem spins as it moves around – The flower headband spins around a person’s head as they move across the screen, heading to the festival	Backdrop: Same Sprites: Hula hoop, tiki totem, <i>person</i> – Same – The tiki totem spins <i>across screen</i> – As the <i>person</i> moves across the screen, heading to the festival, a <i>flower headband</i> spins around their head.			
T1:AA	T2:TA	T3:R	C1:CI	C2:CF	C3:CS
Yes	Yes	No	No/Yes	Yes	No/Yes

Topic	Project Idea				
Chinese Food	Backdrop: A Chinese city with busy streets and traditional architecture Sprites: A cyclist, two cats – The cyclist pedals and delivers dumplings across the screen – Two cats chasing each other around a delivery box				
T1:AA	T2:TA	T3:R	C1:CI	C2:CF	C3:CS
Yes	No	No	No	Yes	No

Table 2: Three examples with codes. The first example can be used as expressed. The last example has deficiencies in both theme and code. The middle example has coding issues as expressed, but a very minor change in description solves the coding issues.

4 Results

In this section, we first perform a deep dive into the coding of three projects of greatly differing quality. We then present the results from analyzing the entire corpus of GPT-generated project ideas.

4.1 Example Deep Dive

Table 2 provides detailed analysis of three project ideas for the animation module. A base project has three sprites: a bee and a snake that move across the screen (loop with movement, costume changes, and wait timing), and a monkey that waves a flag in place (loop with costume changes and wait timing). All three example generated project ideas are creative, appropriate for K-8, and implementable in Scratch. However, they vary in terms of code alignment.

Food Fest project (well-aligned). This project matches the project requirements very well, from both the user and the programmer perspectives. It involves three sprites – the chef animating in place and two waiters moving across the screen. A teacher can efficiently implement this idea by just swapping the sprites and backdrop in the base project; no code changes are needed.

Aloha Festival project (semi-aligned) This project has problems with reasonability and code. It is not clear of the purpose or cause of the hula hoop and tiki totem spinning. Does someone off-screen spin the tiki totem? How does the hula hoop twirl by itself? Thus, the idea is not coherent. When we consider the code, we see that GPT identifies three sprites, as expected (hula hoop,

tiki totem, flower headband). However, the flower headband “*spins around a person’s head as they move across the screen*”, adds a hidden sprite – a person. Further, “*spins around a person’s head as they move across the screen*” implies a complex position and time dependency between the two sprites. However, it turns out that a minor adjustment in how this project idea is expressed solves the problem entirely. If the third sprite is the person and the headband is integrated as part of the person’s costume, it can be implemented just as easily as the base project. However, a teacher would need to think of this implementation, which is not apparent from GPT’s identification of the headband, not the person, as a sprite. Therefore, we consider this idea semi-aligned.

Chinese Food project (poorly-aligned). This project has several problems with both theme quality and code alignment. The suggested project merely involves dumplings, and a Chinese city backdrop, presenting a very surface-level understanding of Chinese culture. It is also incoherent because two cat sprites are not relevant to either the specific topic (dumpling delivery) or the general topic (Chinese food). They seem to be included to meet the number of required sprites as in the base project. Looking at the code, the stationary sprite (cyclist) is not stationary (“*pedals and delivers dumplings across the screen*”). Delivering dumplings also implies many other hidden sprites that get placed at various intervals (e.g., other sprites the cyclist would interact with). Further, “*two cats chasing each other around the delivery box*” creates complex movement

Module	Theme			Code			Multi-Idea	
	T1	T2	T3	C1	C2	C3	V	C
Events	1.00	.75	.92	.93	1.00	.90	.35	.60
Animation	1.00	.85	.62	.62	1.00	.63	.70	.35
CondLoop	1.00	.83	.83	.95	1.00	1.00	.80	.60
SeqDecomp	.92	.88	.78	.78	1.00	.75	.50	.45
1WaySync	.97	.75	.88	.88	1.00	.87	.45	.45
Average	.98	.81	.81	.83	1.00	.83	.56	.49

Table 3: Quality of generated project ideas across modules

(in a circle instead of across the screen from left to right) that requires code blocks involving rotation (e.g., `turn left __ degrees`) and position dependencies between the two sprites. The closest modification would be the cyclist as the moving sprite and the two moving cats integrated into the image of the delivery box sprite. In this case, we would have one stationary sprite with in-place movement and one moving sprite, which is still unaligned with the base project. Therefore, this idea is poorly aligned.

4.2 Overall Quality

Our overall result for theme quality, code alignment, and multi-idea is shown in Table 3 (broken down by module) and Table 4 (broken down by strand). Each cell of the table presents the percentage of “yes” answers to the evaluation questions. Comparing Tables 3 and 4 reveals that performance is similar with either split, so our detailed results below refer to the module breakdown. Finally, the result for complexity issues is shown in Table 5.

Finding 1: On average, GPT-generated project ideas satisfy all individual criteria. Most of the project ideas (98%) are appropriate for K-8 students (T1), 81% align with the topic specified in the prompt (T2), 81% make sense (T3), 83% include sprites that function similarly as the sprites in the base project (C1), 100% are implementable in Scratch (C2), and 83% do not require code change in the base project to implement (C3).

Finding 2: Almost all GPT-generated project ideas are appropriate for K-8 and are implementable in Scratch. Our analysis reveals that all events, animation, and conditional loops project ideas, 92% of sequential decomposition project ideas, and 97% of one-way synchronization project ideas are age-appropriate, and 100% are implementable in Scratch. One inappropriate example is a project involving a hunter throwing a spear toward an animal (violence). Another example is a Juneteenth project with slaves ringing a freedom bell; we believe that a delicate topic such as this would be better taught in a different setting or medium.

Finding 3: GPT produces a supermajority of ideas that are aligned with the topic. Generated project ideas are aligned with the topic 75% to 88% of the time depending on module. Given that there are three ideas per topic, this implies that there are typically at least two ideas aligned to the topic for each query. Of 100 queries (20 queries per module, each prompts GPT to generate three project ideas), only 11 queries (about 9%) give one idea that aligns with the prompted topic, and 6 queries (5%) give an output with no aligned idea. One example not aligned with the given topic is an animation project about Mexican food that has “a Mexican village background”, “a chili pepper waving a maraca”, and “a pig and a chicken racing

Topic	Theme			Code			Multi-Idea	
	T1	T2	T3	C1	C2	C3	V	C
Yth. Cult.	1.00	.97	.87	.90	1.00	.87	.50	.60
Local Cult.	1.00	.67	.83	.83	1.00	.83	.40	.45
Food	.98	.85	.73	.88	1.00	.88	.75	.50
Hol./Trad.	.97	.68	.83	.83	1.00	.85	.70	.45
Hist. Ev.	.93	.90	.77	.72	1.00	.72	.45	.45
Average	.98	.81	.81	.83	1.00	.83	.56	.49

Table 4: Quality of generated project ideas across strands

across the screen”. Here, the Mexican village background can be replaced by many other images, highlighting that the idea is not tailored to Mexican cuisine. Further, the chili pepper, chicken, and pig are not specifically presented as food (e.g., one could perceive them as people wearing mascot costumes). Thus, it appears that GPT arbitrarily picks a Mexican-related backdrop and some food items, proposing the idea as representing Mexican food.

Finding 4: Reasonability, code intent, and code similarity significantly varies by module. While the average percentage of “yes” answers to reasonability, code intent, and code similarity evaluation questions are at least 81%, GPT performs worse for animation and sequential decomposition modules. Only 62% of the animation project ideas make sense, 62% lead to similar functionality (code intent), and 63% can be implemented with no code change (code similarity). For the sequential decomposition module, the number of ideas that qualify for reasonability, code intent, and code similarity are 78%, 78%, and 75% respectively. For other modules, this number ranges from 83% to 100% for these criteria.

Module	Complexity Issues		
	More sprites	More scripts	More blocks
Events	0	0	9
Animation	8	4	5
CondLoop	6	0	0
SeqDecomp	2	4	8
1WaySync	3	8	6
Total	19	16	28

Table 5: Number of complexity issues in generated ideas

Finding 5: GPT-generated project ideas have poor variability and consistency. Half of the generated responses suggest two or more projects that are very similar (only 56% of the responses provide three distinct ideas). Further, variability differs widely between modules, ranging from 35% in events to 80% in conditional loops, with the other modules spread in-between. One particularly poor variability example is when GPT is asked to suggest three different animation project ideas related to the Albuquerque International Balloon Fiesta. All three suggested ideas include “one balloon animates in place” and “two balloons floating across the sky”. They only differ by the backdrop, which are “a backdrop of the Albuquerque skyline”, “a backdrop of the Albuquerque sunset”, and “An open field where the hot air balloon festival happens”. For consistency, only 49% of the responses suggest two or more good projects. Consistency also varies by modules (ranging from 35% to 60%).

Finding 6: There are complexity issues in about 20% of the GPT-generated project ideas. In total, we identify 63 potential issues: 28 issues are extra unique blocks needed to implement (blocks not included in the base project), 19 issues are extra sprites needed, and 16 are extra scripts needed (events to start a sprite’s movement). For example, a common complexity issue in the animation module is the *sprite supposed to only animate in place is instead moving* in many generated project ideas (e.g., cyclist sprite in the Chinese Food project), resulting in extra scripts and extra blocks. Another example is that *“a headband rotating around a moving person”* could be interpreted as two separate sprites, resulting in an extra sprite.

5 Discussion

Our results suggest that using GPT to brainstorm Scratch projects based on an existing structured project and a specific theme shows promise. At least 81% of the project ideas suggested by GPT satisfy our metrics for both theme quality and code alignment. While the quality for reasonability, code intent, and code similarity considerably differs by module, GPT performed consistently well in terms of age-appropriateness and code feasibility, and fairly well in terms of topic alignment. We propose that **teachers can use GPT with our simple prompt template to brainstorm customized projects that incorporate themes relevant to their students and retain the structure of provided base projects.** A good project idea, with an appropriate theme and well-aligned code to the base project, can help greatly reduce the amount of time needed to spend on project implementation, which has been shown to be a key challenge for teachers [32]. To implement such ideas, teachers only need to replace the sprites and backdrop in the base project, avoiding tedious and significant code changes.

We also found that our generated project ideas fall short in terms of technical complexity, which is critical for selecting a good project. There are 63 potential complexity issues in 300 generated ideas. Most issues involve extra unique blocks needed due to complicated movements and hidden sprites. Using a simple prompt like ours, GPT cannot understand critical implementation details such as whether two objects are separate sprites or multiple costumes of a single sprite (e.g., the flag is part of the monkey’s costume, not a stand-alone sprite). The Aloha Festival project idea has *“a headband rotating around a moving person”* that would be overly complicated to implement without any adjustments. If the person and the headband are separate sprites, implementing them would require complex position and time dependency. However, if merging them into the one sprite, it is quite simple. While a complex prompt might solve this issue, that might introduce extra complexity for teachers, who may have limited AI experience. Thus, **teachers may need additional scaffolds, such as warnings about this issue, how to spot extra complexity, and tips on how to simplify some types of ideas** (e.g., changing an animation to a glide block and combining sprites that move together).

Finally, the literature is clear on the importance of providing culturally responsive educational experiences in many subjects [13, 24], including CS [7, 9, 30]. While empowering teachers to tailor existing instructional materials to the needs of their students is an effective approach to CRT at the classroom-level, creating high-quality customized materials is challenging for many CS teachers, even with

static scaffolding such as individual brainstorming exercises and group brainstorming sessions shown to accelerate this process [32]. **This work presents another step towards using generative AI to support teachers in creating localized, classroom-level instructional materials.** However, before bringing AI-generated materials into the classroom, teachers should consider any cultural norms or biases that do not match their knowledge, and perform a member check to filter out or adjust inappropriate ideas.

6 Limitations and Future Work

This study has several limitations. First, the poor variability and consistency found in the generated projects could be due to our prompt specificity, not necessarily model performance. We acknowledge that further prompt-tuning may improve the response quality, however, it is not the central goal of this study. We instead investigated GPT performance on a simple yet reasonable prompt to understand the potential of using LLMs from a teacher/non-AI expert perspective [34]. Second, the topics we used to generate project ideas may not be fully representative of topics teachers used in their classrooms. A follow-up study involving teachers and students can address this limitation. Finally, we examined only one causal language model, and chose GPT-3 as it was the most accessible free model at the time of our study. It is possible that other models or later versions of GPT will outperform our results. Future work may explore using different models to obtain different-quality outputs.

7 Conclusion

This work explores the possibility of teachers using GPT to brainstorm customized Scratch project ideas serving as demo and code exploration exercises for students. The goal is to obtain projects that *retain the structure of the base curriculum and relates to a specific topic relevant to students* through a simple prompt. Our study has three major contributions. First, it examines the strengths and weaknesses of LLMs in supporting CS teachers to efficiently customize instructional materials. From an individual project idea standpoint, GPT created a plethora of creative ideas as prompted, but had certain issues in (1) matching the exact technical requirements and (2) providing superficial ideas that are not yet ready for teachers to use as is. We identified specific patterns in mismatches between GPT-generated project ideas and the technical attributes of the projects we asked it to mimic. Second, this study provides practical guidance in how teachers can effectively use GPT and other publicly-available language models, even in the face of their weaknesses. While LLMs are very useful for an early ideation phase, teachers should not directly implement the ideas without performing complexity check and content bias check. Finally, we propose a novel application of generative AI in K-8 CS education: supporting teachers in the creation of localized instructional materials.

Acknowledgements

The authors would like to thank Andrew Chu for their help with the study and feedback on the paper. This work is supported by the National Science Foundation under Grant No. 2201313.

References

- [1] Louise Archer, Jennifer DeWitt, and Billy Wong. 2014. Spheres of influence: What shapes young people's aspirations at age 12/13 and what are the implications for education policy? *Journal of Education Policy* 29, 1 (2014), 58–85.
- [2] Deborah Loewenberg Ball and David K Cohen. 1996. Reform by the book: What is—or might be—the role of curriculum materials in teacher learning and instructional reform? *Educational researcher* 25, 9 (1996), 6–14.
- [3] Sharon E Booth and Shaun B Kellogg. 2015. Value creation in online communities for educators. *British Journal of Educational Technology* 46, 4 (2015), 684–698.
- [4] Karen Brennan, Christan Balch, and Michelle Chung. 2014. *Creative computing: A design-based introduction to computational thinking*. <https://creativecomputing.gse.harvard.edu/guide/>
- [5] Seth Chaiklin et al. 2003. The zone of proximal development in Vygotsky's analysis of learning and instruction. *Vygotsky's educational theory in cultural context* 1, 2 (2003), 39–64.
- [6] Merijke Coenraad, David Weintrop, Donna Eateringer, Jen Palmer, and Diana Franklin. 2021. Identifying youths' spheres of influence through participatory design. *Designs for learning* 13, 1 (2021).
- [7] Kalisha Davis, Shana V White, Becton-Consuegra Dinah, and Allison Scott. 2021. *Culturally Responsive-Sustaining Computer Science Education: A Framework*. Technical Report. Technical Report. Kapor Center. <https://www.kaporcenter.org/equitables>.
- [8] Andre Del Carpio Gutierrez, Paul Denny, and Andrew Luxton-Reilly. 2024. Evaluating Automatically Generated Contextualised Programming Exercises. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. 289–295.
- [9] Ron Eglash, Juan E Gilbert, and Ellen Foster. 2013. Toward culturally responsive computing education. *Commun. ACM* 56, 7 (2013), 33–36.
- [10] Sally Fincher, Michael Kölling, Ian Utting, Neil Brown, and Phil Stevens. 2010. Repositories of teaching material and communities of use: nifty assignments and the greenroom. In *Proceedings of the Sixth international workshop on Computing education research*. 107–114.
- [11] Diana Franklin, David Weintrop, Jennifer Palmer, Merijke Coenraad, Melissa Cobian, Kristan Beck, Andrew Rasmussen, Sue Krause, Max White, Marco Anaya, et al. 2020. Scratch Encore: The design and pilot of a culturally-relevant intermediate Scratch curriculum. In *Proceedings of the 51st ACM technical symposium on computer science education*. 794–800.
- [12] Jason Freeman, Brian Magerko, Tom McKlin, Mike Reilly, Justin Permar, Cameron Summers, and Eric Fruchter. 2014. Engaging underrepresented groups in high school introductory computing through computational remixing with EarSketch. In *Proceedings of the 45th ACM technical symposium on Computer science education*. 85–90.
- [13] Geneva Gay. 2018. *Culturally responsive teaching: Theory, research, and practice*. teachers college press.
- [14] Natasa Gisev, J Simon Bell, and Timothy F Chen. 2013. Interrater agreement and interrater reliability: key concepts, approaches, and applications. *Research in Social and Administrative Pharmacy* 9, 3 (2013), 330–338.
- [15] David Gonzalez-Maldonado, Jonathan Liu, and Diana Franklin. 2025. Evaluating GPT for use in K-12 Block Based CS Instruction Using a Transpiler and Prompt Engineering. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*.
- [16] Google. 2023. *CS First: A computer science curriculum that makes coding easy to teach and fun to learn*. <https://csfirst.withgoogle.com/s/en/home>
- [17] Xinying Hou, Zihan Wu, Xu Wang, and Barbara J Ericson. 2024. CodeTailor: Personalized Parsons Puzzles are Preferred Over AI-Generated Solutions to Support Learning. *arXiv preprint arXiv:2401.12125* (2024).
- [18] Garima Jain, Fred Martin, Bernardo Feliciano, Hsien-Yuan Hsu, Barbara Fauvel-Campbell, Gillian Bausch, Lijun Ni, and Elizabeth Thomas-Cappello. 2022. CS pathways: A culturally responsive computer science curriculum for middle school. In *2022 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.
- [19] Mengying Jiang and Kristin A Searle. 2024. Developing Culturally Responsive-Sustaining Computer Science Curricula: A Design-Based Research Study. In *Proceedings of the 18th International Conference of the Learning Sciences-ICLS 2024*, pp. 1630–1633. International Society of the Learning Sciences.
- [20] Breanna Jury, Angela Lorusso, Juho Leinonen, Paul Denny, and Andrew Luxton-Reilly. 2024. Evaluating Llm-generated worked examples in an introductory programming course. In *Proceedings of the 26th Australasian Computing Education Conference*. 77–86.
- [21] Yasmin B Kafai, Eunkyong Lee, Kristin Searle, Deborah Fields, Eliot Kaplan, and Debora Lui. 2014. A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles. *ACM Transactions on Computing Education (TOCE)* 14, 1 (2014), 1–20.
- [22] Filiz Kaleioğlu. 2015. A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior* 52 (2015), 200–210.
- [23] Michael Lachney, Aman Yadav, Matt Drazin, and Briana Green. 2023. Community embedded computing education: Shaping young people's perceptions of self-confidence and personal expression with computer science in a youth boxing gym. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 1103–1109.
- [24] Gloria Ladson-Billings. 1995. Toward a theory of culturally relevant pedagogy. *American educational research journal* 32, 3 (1995), 465–491.
- [25] Gloria Ladson-Billings. 2014. Culturally relevant pedagogy 2.0: aka the remix. *Harvard educational review* 84, 1 (2014), 74–84.
- [26] Evanfiya Logacheva, Arto Hellas, James Prather, Sami Sarsa, and Juho Leinonen. 2024. Evaluating Contextually Personalized Programming Exercises Created with Generative AI. In *Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1*. 95–113.
- [27] Brian Magerko, Jason Freeman, Tom McKlin, Scott McCoid, Tom Jenkins, and Elise Livingston. 2013. Tackling engagement in computing with computational music remixing. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 657–662.
- [28] Kathryn M. Rich, Jill Bowdon, and Marissa Spang. 2023. Implementing Culturally Responsive Coding Projects with Indigenous Communities. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2* (Toronto ON, Canada) (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 1355. <https://doi.org/10.1145/3545947.3576299>
- [29] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. 27–43.
- [30] Kimberly A Scott, Kimberly M Sheridan, and Kevin Clark. 2015. Culturally responsive computing: A theory revisited. *Learning, media and technology* 40, 4 (2015), 412–436.
- [31] Mike Sosteric and Susan Hesemeier. 2002. When is a Learning Object not an Object: A first step towards a theory of learning objects. *International Review of Research in Open and Distributed Learning* 3, 2 (2002), 1–16.
- [32] Minh Tran, Heather Killen, Jen Palmer, David Weintrop, and Diana Franklin. 2024. Harmonizing Scratch Encore: Scaffolding K-8 Teachers in Customizing Culturally Responsive Computing Materials. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. 1335–1341.
- [33] Lev Vygotsky et al. 2011. *Interaction between learning and development*. Linköpings universitet.
- [34] JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–21.