

# Impersonating Chatbots in a Code Review Exercise to Teach Software Engineering Best Practices

Juan Carlos Farah<sup>\*</sup>, Basile Spaenlehauer<sup>\*</sup>, Vandit Sharma<sup>†</sup>,  
María Jesús Rodríguez-Triana<sup>‡</sup>, Sandy Ingram<sup>§</sup>, and Denis Gillet<sup>\*</sup>

<sup>\*</sup>School of Engineering, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland  
Email: {juancarlos.farah,basile.spaenlehauer,denis.gillet}@epfl.ch

<sup>†</sup>Department of Computer Science, Eidgenössische Technische Hochschule Zürich (ETHZ), Zurich, Switzerland  
Email: sharmav@ethz.ch

<sup>‡</sup>School of Digital Technologies, Tallinn University, Tallinn, Estonia  
Email: mjrt@tlu.ee

<sup>§</sup>School of Engineering and Architecture of Fribourg, University of Applied Sciences (HES-SO), Fribourg, Switzerland  
Email: sandy.ingram@hefr.ch

**Abstract**—Over the past decade, the use of chatbots for educational purposes has gained considerable traction. A similar trend has been observed in social coding platforms, where automated agents support software developers with tasks such as performing code reviews. While incorporating code reviews and social coding platforms into software engineering education has been found to be beneficial, challenges such as steep learning curves and privacy considerations are barriers to their adoption. Furthermore, no study has addressed the role chatbots play in supporting code reviews as a pedagogical tool. To help address this gap, we developed an online learning application that simulates the code review features available on social coding platforms and allows instructors to interact with students using chatbot identities. We then embedded this application within a lesson on software engineering best practices and conducted a controlled in-class experiment. This experiment examined the effect that explaining content via chatbot identities had on three aspects: (i) students’ perceived usability of the lesson, (ii) their engagement with the code review process, and (iii) their learning gains. While our findings show that it is feasible to simulate the code review process within an online learning platform and achieve good usability, our quantitative analysis did not yield significant differences across treatment conditions for any of the aspects considered. Nevertheless, our qualitative results suggest that students expect explicit feedback when performing this type of exercise and could thus benefit from automated replies provided by an interactive chatbot. We propose to build on our current findings to further explore this line of research in future work.

**Index Terms**—chatbots, code review, code linting, software engineering education, online learning, digital education

## I. INTRODUCTION

The use of chatbots in educational settings has been rising steadily over the past decade. Recent surveys have shown active research into the development and use of chatbots for education [1], as well as their wide availability on instant messaging platforms [2]. Software engineering education is particularly poised for the adoption of chatbots, given the widespread use of chatbots on social coding platforms [3]. Well-established in industry, social coding platforms (e.g., GitHub, GitLab, Bitbucket) are collaborative workspaces focused on software development [4]. These platforms are often

based on version control software and provide bespoke interfaces to support reviewing code, leaving comments, visualizing changes, and carrying out other related tasks. Chatbots can help with some of these tasks and have thus become a common feature of the social software development process [5]. Nevertheless, while several studies have assessed integrating social coding features into educational contexts [6], the impact that chatbots could have on supporting these integrations has not been addressed in the literature. In particular, while the code review process has been found to be suitable for educational purposes [7], [8], no study has assessed the role chatbots could play in supporting code review exercises.

Outside of education, a recent exploratory study showed that adopting software agents to perform code reviews had a positive impact on contributions to open-source software projects and also resulted in decreased communication between maintainers and contributors [9]. Though preliminary, these results may imply that chatbots could play a positive part in software engineering education. Drawing a parallel from a social coding platform to an educational context—with contributors as students and maintainers as instructors—we posit that chatbots might be able to increase student engagement during code review exercises while also reducing instructors’ workloads with respect to the guidance required during these exercises.

Our work aims to explore this line of research. Specifically, we investigate the effects of using a chatbot to present content related to code quality standards within a simulated code review exercise. To approach this goal, we posed the following research questions:

- 1) Can we successfully simulate the code review functionalities offered by social coding platforms within an online learning environment? (RQ1)
- 2) Will presenting concepts related to code quality using a chatbot have an impact on students’ perceived usability of the lesson, their engagement with the code review process, and their learning gains? (RQ2)

In this study, we present the design, implementation, and evaluation of an application that allows users—both instructors and students—to review and comment code in a way that simulates social coding platforms. This application supports different use cases that could be applicable to software engineering education. Here, we focus on the following two use cases. First, instructors can seed the code review application with a code snippet alongside comments explaining the issues present in the snippet. This can provide students with an example of what developers are looking for when a code review is performed. Second, instructors can seed the code review application with a code snippet containing issues that students are then asked to find and annotate. This can serve as a way to evaluate students’ understanding of the code review process and the quality of the code at hand. Our design also allows instructors to integrate chatbots into the code review process. Instructors can define chatbot identities and then use them to provide comments in the examples selected to illustrate the code review process. In our current implementation, these chatbots only participate in the code review process when impersonated by the instructor or by teaching assistants. That is, our chatbots cannot interact automatically with students.

To address our research questions, we configured the code review application following the two aforementioned use cases and incorporated it into an online lesson on enforcing code quality standards in JavaScript. We then evaluated this implementation in a controlled in-class experiment comprising 30 undergraduate software engineering students.

Our objective was twofold. First, to validate the usability of a learning scenario that incorporated the code review application. Second, to test whether there was an effect on perceived usability, student engagement, and learning gains if explanations regarding code quality were provided differently. To do this, we evaluated two treatments: (i) providing explanations as the course instructor within the code review application (instructor condition) and (ii) providing explanations within the code review application through a chatbot that the course instructor impersonated (chatbot condition). Finally, to provide a baseline, we also included a condition in which the code review application was only used to display the code snippet, while explanations of the issues therein were provided in the text preceding the code review application (control condition).

Our paper is structured as follows. We begin by reviewing related work and motivating our study. In Section III, we present the design of our code review application, chatbot integration, chatbot identity, and learning scenario. We outline our methodology in Section IV and report on the results in Section V. These results are discussed in Section VI alongside limitations and future work. Finally, we summarize our conclusions in Section VII.

## II. BACKGROUND AND RELATED WORK

The use of social coding platforms in education has been widely studied [6], [10], [11], [12], [13], [14], with a recent survey of 7530 students and 300 educators concluding that the use of GitHub “predicted better learning outcomes and

classroom experiences” [12]. However, several researchers have highlighted a number of challenges associated with the use of social coding platforms in education, including steep learning curves and privacy considerations due to the often public visibility of the interactions on such platforms [6], [10], [11]. These challenges motivate the development of our code review application as a way to address RQ1. Namely, to lower the barrier to entry for incorporating and exploiting the code review process within online learning environments.

Our focus on using code review as a pedagogical tool builds on prior work highlighting its applicability to software engineering education. A study conducted by Li et al. on effectively teaching coding standards in programming revealed that most students believe coding standards are important [7]. However, students still fail to comply with them, thus highlighting possible flaws in current teaching strategies. To that end, the researchers suggested code review as a possibly effective pedagogical tool, given that it provides an element of learning by example and practice, as well as an opportunity for obtaining feedback from teachers and other learners. Another study conducted by Bacchelli et al. on Microsoft employees highlighted additional benefits of code review, such as knowledge transfer, increased team awareness, and the creation of alternative solutions to problems [15].

Given the benefits of code review, studies have explored designs for incorporating the code review process as a part of programming education [16], [17], [18]. Wang et al. proposed an online assessment system based on code review to assess how students learn programming languages [8]. Empirical results from their study showed that the code review-based assessment significantly improved student learning outcomes in several areas such as programming skills, collaborative learning, and compliance with coding standards. Positive results were also obtained when testing code review activities in high school and university settings [19], [20].

While peer code review provides various opportunities for learning, it often entails a significant amount of human effort from the various entities involved [21]. Automated agents such as chatbots can help reduce this effort by automatically checking code standards and providing necessary feedback where required. Balachandran proposed *Review Bot*, an automated agent designed to integrate automatic static analysis within the code review process [22]. Indeed, using bots for automating code maintenance tasks has been gaining popularity in recent years. A study conducted by Wessel et al. in 2018 revealed that nearly 26% of popular open-source projects on GitHub already use bots for numerous tasks [3]. Among the 48 bots identified in the study, 14 bots (29.2%) were being used for reviewing code and pull requests, highlighting the suitability of bots for code review. Our work builds on these results to motivate the integration of chatbot identities into our code review application, which was designed for educational purposes.

In education, the rise of instant messaging platforms has motivated both educators and learners to adopt chatbots, with recent surveys identifying 89 educational conversational agents on the Facebook Messenger platform alone [2]. However—as

recently as 2021—researchers have raised the issue that there are “few empirical studies investigating the use of effective learning designs or learning strategies with chatbots” [23]. Although the findings of some of these empirical studies are actually promising [24], there is ample need for more research to ground the use of chatbots in education on solid results. In fact, our understanding of the factors affecting chatbot user experiences is limited even outside educational settings [25].

Our paper contributes to addressing this gap by tackling RQ2. Specifically, we aim to shed light on whether the simple act of presenting students with information via a static chatbot has an effect on students’ perception of the usability of the interface in which the lesson is presented, their engagement with the tools provided by the interface, and the short-term learning gains acquired during the lesson. Moreover—to the best of our knowledge—no study has evaluated the impact of incorporating chatbots in a code review exercise within an online learning scenario. Given the widespread use of chatbots in social platforms and the trend to incorporate these platforms into educational contexts, we believe our study is both timely and relevant.

### III. DESIGN

In this section, we introduce the design of our code review application. We then show how we integrated a chatbot interface into this application and present *Lint Bot*, the chatbot identity used in this study. Finally, we illustrate how the code review application was embedded within a lesson on enforcing code quality standards in JavaScript.

#### A. Code Review Application

To provide the context in which to implement our chatbot integration, we developed an online learning application aimed at allowing students to review snippets of code. Code reviews are a key part of any software development project [26]. As teams of engineers collaborate to develop an application, website, or other software, they often follow a process whereby one team member submits code that a colleague will review before the code gets merged into a central repository. This process often happens on social coding platforms, such as GitHub, GitLab, or Bitbucket. Nevertheless, full-fledged social coding platforms can be daunting for entry-level programming students [6]. In order to provide an introduction to code reviews for beginner programmers, our code review application provides the basic code review functionalities typically supported by a social coding platform.

First, the code presented is static, as its purpose is not to be modified but rather to be commented on by users to start a discussion about possible improvements. Users can add comments to a specific code line by pressing the “+” button at the beginning of the line. An editor supporting Markdown [27] provides users with the functionality to input, preview, create, edit, and delete comments. Users also have the possibility of replying to existing comments to start a thread-like exchange, as shown in Figure 1. While replies to comments are displayed as nested boxes inside their parent comment, a “hide” button

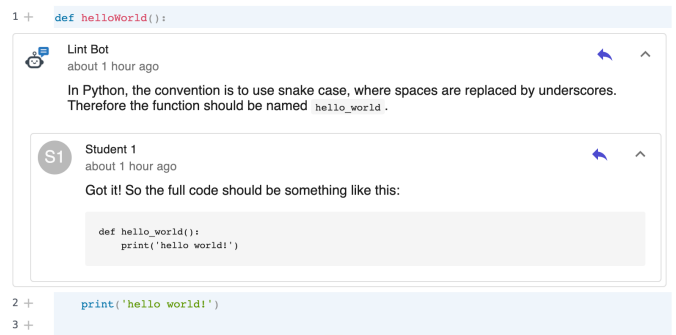


Fig. 1. The code review application allows students to reply to comments seeded by the instructor, leading to thread-like exchanges. As depicted here, the instructor can impersonate a chatbot when annotating the code snippet.

toggles the visibility of a comment’s content, freeing up space to let the user focus on the underlying code or other comments.

The application also provides an interface for instructors. Within this interface, instructors can view interactions on a per-student basis and seed the interface with comments that will be shown to all students. Finally, comments are only visible to the student, the instructor, and optionally to the student’s peers, if configured as such by the instructor.

#### B. Chatbot Integration

Our objective in the design of our chatbot integration was to provide instructors with a way of easily configuring and impersonating chatbots in order to interact with students within an online learning application. To achieve this, we designed an interface whereby an instructor can easily create chatbots that can be then used to interact with students through the learning application in which the interface is embedded.

To guide instructors, we incorporated three best practices for the design and implementation of chatbots that are relevant to scenarios where chatbots are being impersonated by human users [25], [28], [29]. Namely, we wanted to ensure that instructors (i) provide the chatbot with an identity, (ii) explicitly state the chatbot’s purpose, and (iii) can interact through messages including content that students might find pleasant, evocative, or playful. First, with respect to (i) and (ii), the interface provided a way to define a chatbot’s identity, which was given by its name, an optional avatar, and an optional description, with both the name and description serving as a way to explicitly state the chatbot’s purpose. Second, with respect to (iii), the messages sent by the instructor under the chatbot’s guise can be formatted using Markdown, which supports embedding rich text and multimedia, making it possible to easily create engaging content.

We implemented this chatbot integration within the code review application. Figure 2 shows an instance of the code review application that has been seeded with three chatbot identities. Once a chatbot identity has been created, instructors can impersonate it to provide feedback to students, as shown in Figure 3. Students can then reply back, leading to interactions similar to the one shown previously in Figure 1.













Bot Users			
Avatar	Name	Description	Actions
	Lint Bot	Lint Bot checks the quality of your code.	  
	Test Bot	Test Bot helps you test your code.	  
	Teaching Bot	Teaching Bot helps teachers explain content to their students.	  

Fig. 2. Instructors can create different bot identities that they can then impersonate when interacting with students.

### C. Lint Bot

As mentioned in Section I, many social coding platforms support using chatbots to perform a variety of tasks [3]. One of these tasks is to verify that code submitted to a repository is formatted according to the repository’s respective style and guidelines. The term “linting” refers to the use of static analysis tools (“linters”) to detect bugs and other issues (“lint”) in software programs [30]. Therefore, for our evaluation, we created a bot identity using the name *Lint Bot*. This identity would be used to explain code quality issues to students. As noted in Section I, it is important to underline that Lint Bot was simply an identity and did not have any agency or script to reply automatically to student comments. That is, for Lint Bot to interact with students, instructors had to manually post comments and replies using Lint Bot’s identity. In our case, for example, the instructor that participated in our evaluation impersonated Lint Bot to seed the comments in one of the treatment conditions (chatbot condition). Students assigned to this condition were unaware that the chatbot was impersonated by the instructor, following the Wizard of Oz technique [31].

### D. Learning Scenario

Our goal was to evaluate the code review application and the effect of equipping it with a chatbot interaction in a real software engineering lesson. To do this, we created a lesson introducing the concept of code linting. This lesson was hosted on the Graasp online learning platform [32] and was structured following five phases. In the first phase, students were presented with a code review application containing a snippet of code. Students were explained how to use the code review application and asked to add comments to the parts of the code that had potential issues. This exercise served as a pre-test to (i) gauge their level of engagement at the start of the lesson and (ii) see if they could find six issues that had been explicitly seeded in the code. In the second phase, students were introduced to the concept of code linting, while in the third phase, they were introduced to ESLint [33], a popular tool specifically designed to find linting errors in JavaScript [34]. In the fourth phase, four specific code quality issues were presented and exemplified in detail using the code

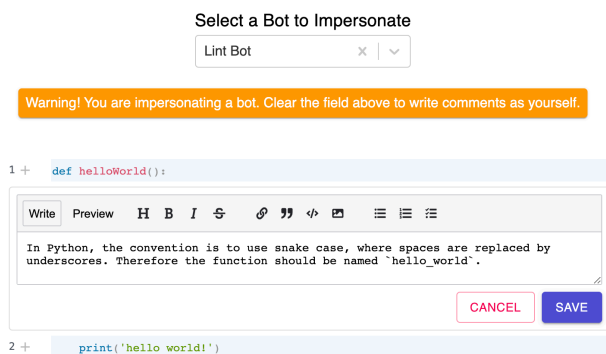


Fig. 3. Instructors can impersonate chatbots to seed a code snippet with comments or to provide feedback to students.

review application. This fourth phase was the only part of the scenario that varied across conditions (see Section IV-B for more details). Finally, in the fifth phase, students were again presented with an exercise using the code review application. This exercise served as a post-test to (i) gauge their level of engagement at the end of the lesson and (ii) see if they could find seven issues present in the code snippet. Figure 4 depicts the learning scenario, highlighting the lesson’s second phase (*Introduction*).

## IV. METHODOLOGY

The approach to addressing our research questions consisted of a two-step process. The first step was to tackle RQ1 by implementing the code review application, embedding it within an online learning scenario, and then using this scenario to teach a lesson on software engineering best practices. This practical implementation then served as an indication that it is possible to successfully simulate code review functionalities within an online learning platform. In a second step, we

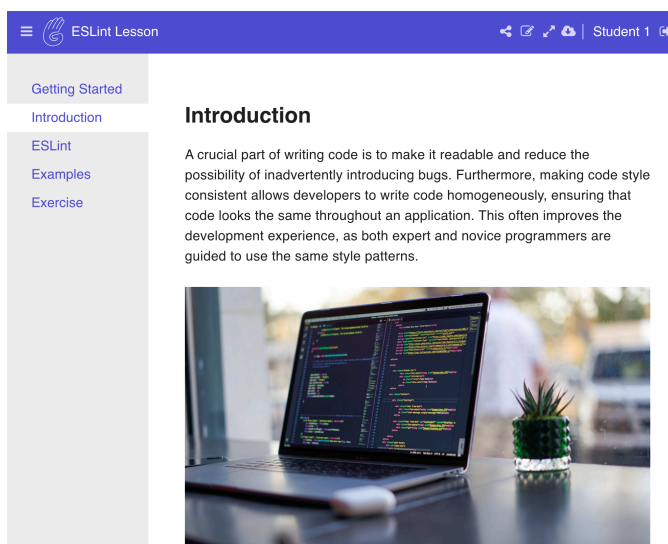


Fig. 4. We prepared an online lesson introducing students to the concept of code linting in JavaScript.

☰ ESLint Lesson
🔗 📄 🔄 🗑️ | Student 1 🏠

Getting Started

Introduction

ESLint

Examples

Exercise

## Confusing Code

Using the same name for variables across scopes and providing unintuitive conditions in clauses can result in code that is confusing for other developers. Avoiding these situations or adding comments that explain why they are used will allow others to better understand your code.

```

1 + let result = 2;
2 + if (true) {
3 +   let result = 3;
4 +   console.log("inner: " + result);
5 + }
6 + console.log("outer: " + result);

```

Lint Bot  
17 days ago

**Warning: Unexpected constant condition.**

A constant expression (for example, a literal) as a test condition might be a typo or development trigger for a specific behavior.

Lint Bot  
17 days ago

**Warning: Variable 'result' is already declared in the upper scope.**

Shadowing is the process by which a local variable shares the same name as a variable in its containing scope. This can cause confusion while reading the code and make it impossible to access the variables in the upper scope.

Fig. 5. The *Examples* phase of the learning scenario varied across experimental conditions. This figure depicts the interface for the chatbot condition, showcasing an example code snippet with linting errors that have been highlighted by comments from an instructor impersonating *Lint Bot*.

addressed RQ2 by evaluating three variations of our learning scenario. More specifically, we tested the effect of impersonating chatbots when presenting concepts in software engineering education, conducting a between-subjects controlled experiment comprising one control and two treatments. In all conditions, subjects were presented with the learning scenario described in Section III-D. The conditions differed only in the way we explained the linting errors illustrated by the example code snippets. In this section, we explain the methodology of our evaluation in detail.

### A. Participants

We recruited 31 undergraduate software engineering students following a course on front-end web development at the School of Engineering and Architecture of Fribourg (HEIA-FR). Students consented to participate in this study. Responses from one participant were omitted after controlling for data collection errors, resulting in a total of 30 valid participants (2 female, 28 male), with a distribution of 9 participants in the control condition, 10 participants in the instructor condition, and 11 participants in the chatbot condition.

### B. Procedure

The experimental part of our study was conducted in November 2021. As part of an in-class exercise, students were given 30 minutes to complete the learning scenario described in Section III-D. The content of the lesson was identical for all students except for the fourth phase (*Examples*), which

differed in the way the code quality issues were explained. In the control condition, the explanation was presented in text alongside—but not embedded within—the code review application (control condition). In the two treatments, we either provided the explanation within comments in the code review application (i) via a user representing the course instructor (instructor condition) or (ii) under the alias of Lint Bot (chatbot condition). All explanations were in English, based on the feedback provided by the ESLint command line interface [33]. Figure 5 depicts the *Examples* phase for the chatbot condition. Interactions with the lesson were recorded using Graasp’s learning analytics (LA) pipeline [35]. Upon completion of the lesson, students completed a post-questionnaire containing the instruments described in the following section.

### C. Instruments

To assess the impact of the treatment conditions of our controlled experiment, we operationalized three concepts—(i) usability, (ii) engagement, and (iii) learning gains—using the following instruments. To collect data regarding usability, we used the System Usability Scale (SUS) [36]. The SUS is a ten-item questionnaire using a five-point Likert scale to measure usability. Results are presented as scores ranging from 0 to 100, with higher scores representing better usability. We refer to this metric as the *SUS score*.

For engagement and learning gains, we built bespoke instruments using the results of the pre- and post-tests. As explained in Section III-D, the pre-test assessed if students could find six

issues seeded in a code snippet, while the post-test assessed if they could find seven issues seeded in a different code snippet. Engagement was operationalized using the length of the comments added by students. The total length  $h_{l,t}$  of the set of comments  $C_{l,t}$  added by a given student  $l$  in test  $t$  can be calculated as the sum of the length of each comment in the set.

$$h_{l,t} = \sum_{c \in C_{l,t}} \text{len}(c)$$

The total length of the comments added by a given student in the pre-test ( $h_{l,\text{pre}}$ ) served as an indicator of how engaged the student was at the beginning of the activity, before being exposed to their respective condition. The total length of comments added in the post-test ( $h_{l,\text{post}}$ ) then served as an indicator of how engaged the student was at the end of the lesson, after being exposed to their respective condition. The engagement resulting from their exposure was then measured as the difference between the total length of comments left by the student in the post-test and the pre-test. We refer to this metric as the *engagement score* ( $e$ ).

$$e_l = h_{l,\text{post}} - h_{l,\text{pre}}$$

Learning gains were operationalized using the scores of the post-test and the pre-test. That is, for a given student  $l$ , their learning gain  $g_l$  was calculated as the difference between their score in the post-test  $w_{l,\text{post}}$  and their score in the pre-test  $w_{l,\text{pre}}$ . We refer to this metric as the *learning gain* ( $g$ ).

$$g_l = w_{l,\text{post}} - w_{l,\text{pre}}$$

Finally, we also captured qualitative feedback through an open-ended question asking students to provide comments, suggestions, and general observations in short answer form.

#### D. Data Analysis

Data collected through the aforementioned instruments were analyzed using mixed methods. Quantitative data were analyzed using descriptive and inferential statistics. In terms of descriptive statistics, we report the sample mean ( $\bar{x}$ ), median ( $\tilde{x}$ ), and standard deviation ( $s$ ). In terms of inferential statistics, data on usability, engagement, and learning gains were all normally distributed and homoscedastic within each condition, and were therefore compared using analysis of variance (ANOVA). Qualitative data—in the form of open-ended comments—were analyzed by articulating emergent themes using line-by-line data coding [37].

### V. RESULTS

In this section, we present the main results of our evaluation. These results are also summarized in Table I and illustrated in Figure 6.

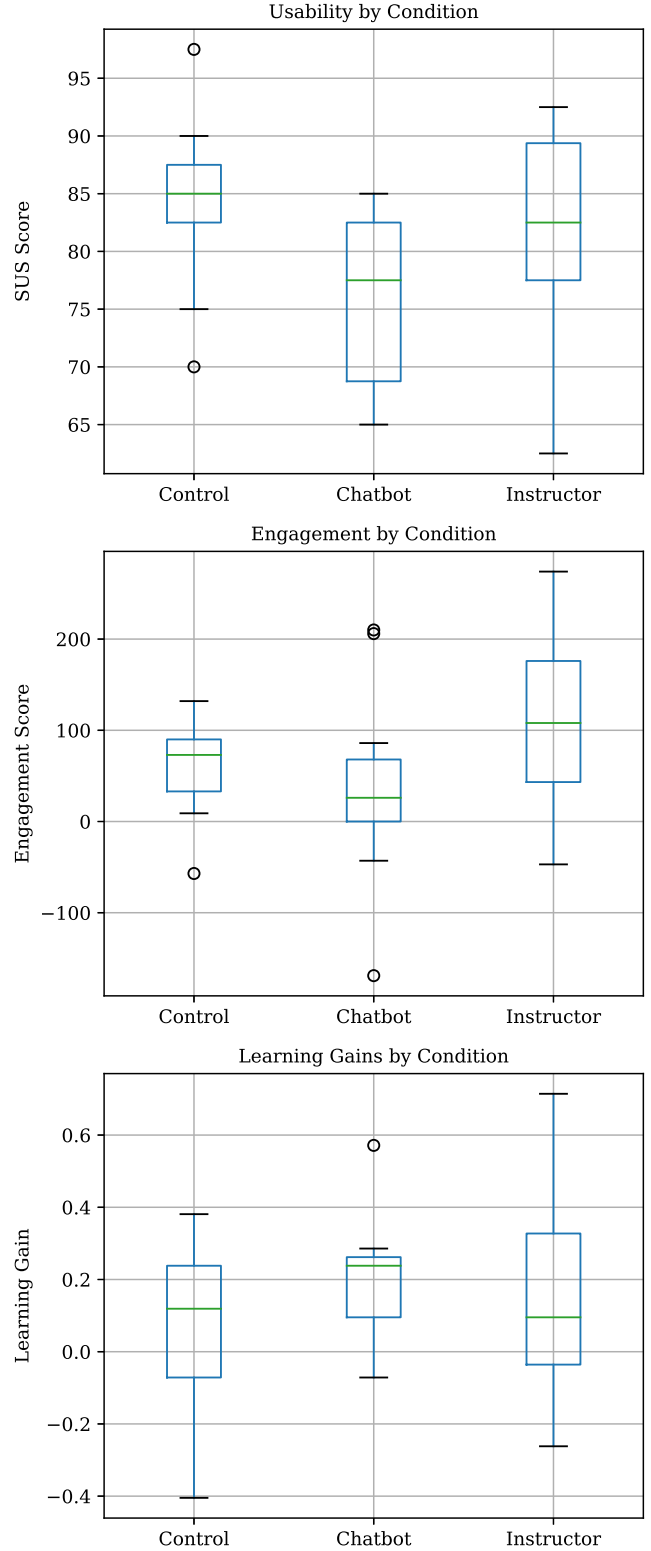


Fig. 6. Box plots summarizing the descriptive statistics across conditions for the three aspects considered by our quantitative analysis: usability (top), engagement (middle), and learning gains (bottom).

TABLE I  
DESCRIPTIVE STATISTICS FOR EACH CONDITION AND ANOVA RESULTS ACROSS CONDITIONS

Aspect	Condition			ANOVA F (p-value)
	Control Mean (SD)	Chatbot Mean (SD)	Instructor Mean (SD)	
Usability	84.167 (8.004)	75.682 (7.508)	82.250 (9.238)	2.978 (0.068)
Engagement	57.556 (56.593)	38.727 (106.593)	108.500 (97.803)	1.606 (0.219)
Learning Gains	0.071 (0.231)	0.184 (0.179)	0.145 (0.287)	0.577 (0.568)

#### A. Usability

The mean SUS scores were  $\bar{x} = 82.250$  ( $\hat{x} = 82.500$ ,  $s = 9.238$ ) in the instructor condition,  $\bar{x} = 75.682$  ( $\hat{x} = 77.500$ ,  $s = 7.508$ ) in the chatbot condition, and  $\bar{x} = 84.167$  ( $\hat{x} = 85.000$ ,  $s = 8.004$ ) in the control condition. These scores all correspond to usability that can be described as *good* [38]. Furthermore, one-way ANOVA showed that there were no significant differences between the conditions ( $F(2, 27) = 2.978$ ,  $p = 0.068$ ).

#### B. Engagement

The mean engagement scores were  $\bar{x} = 108.500$  ( $\hat{x} = 108.000$ ,  $s = 97.803$ ) in the instructor condition,  $\bar{x} = 38.727$  ( $\hat{x} = 26.000$ ,  $s = 106.593$ ) in the chatbot condition, and  $\bar{x} = 57.556$  ( $\hat{x} = 73.000$ ,  $s = 56.593$ ) in the control condition. These descriptive results suggest that engagement was higher in the instructor condition. However, one-way ANOVA showed that there were no significant differences between the conditions ( $F(2, 27) = 1.606$ ,  $p = 0.219$ ).

#### C. Learning Gains

The mean learning gains were  $\bar{x} = 0.145$  ( $\hat{x} = 0.095$ ,  $s = 0.287$ ) in the instructor condition,  $\bar{x} = 0.184$  ( $\hat{x} = 0.238$ ,  $s = 0.179$ ) in the chatbot condition, and  $\bar{x} = 0.071$  ( $\hat{x} = 0.119$ ,  $s = 0.231$ ) in the control condition. These results show that all conditions led to—on average—positive learning gains for students. Nevertheless, when comparing between the conditions, one-way ANOVA showed that there were no significant differences ( $F(2, 27) = 0.577$ ,  $p = 0.568$ ).

#### D. Qualitative Feedback

A total of 10 students provided open-ended feedback, four from the instructor condition, three from the chatbot condition, and three from the control condition. Two themes were presented by more than one student. First, two students (one in the instructor condition and one in the control condition) described the examples in the lesson as “repetitive”. Second, four students (one in the instructor condition, one in the chatbot condition, and two in the control condition) expressed the need for feedback after having completed the post-test. Specifically, one student expressed the following with respect to the need for feedback: “You explain the concept, yes, but the fact that we do an exercise without being able to see the answer

key makes it all frustrating and not at all useful in learning.”<sup>1</sup> This quote illustrates how the second theme—which was the only one seen across all three conditions—was the one that students who provided open-ended comments were most vocal about.

## VI. DISCUSSION

The design process and results of our evaluation provide key insights with respect to our research questions. In this section, we discuss these results as well as certain limitations of our study and future work that we aim to carry out.

Concerning RQ1, the design of our code review application—as well as its implementation within a learning scenario and its use to teach software engineering best practices to undergraduate students—is a proof-of-concept validation that code review functionalities offered by social coding platforms can be simulated within online learning environments. Students that participated in the experiment were able to annotate the different code snippets as if they were engaging in an actual code review process, reporting SUS scores that resulted in positive mean usability ratings across all conditions. Furthermore, the instructor interface served to evaluate students’ responses and gather part of the data used for our study.

This prototype is a strong first step towards lowering the barrier to entry for novice students to exploit functionalities offered by social coding platforms. Students are not required to pre-install software or create accounts, and comment visibility is limited to the concerned parties. The code review application also serves instructors and researchers in education looking to better understand how students behave and learn in scenarios involving social coding interactions. By integrating the code review application within an online learning platform, instructors and researchers can gain insight into students’ learning experiences using the built-in LA features often supported by learning platforms.

Nevertheless, our current evaluation was limited to software engineering students. To test whether our code review application can indeed lower the barrier to entry, we aim to perform a similar study with non-technical students following

<sup>1</sup>“On nous explique le truc oui, mais le fait qu’on aie [*sic*] un exercice sans pouvoir avoir un corrigé rend le tout frustrant et pas du tout utile dans l’apprentissage.” (Translated from French by the authors.)

an introductory programming course, as well as with a more gender-balanced cohort of participants.

With respect to RQ2, our evaluation provided inconclusive yet promising results regarding the role chatbots could play in learning scenarios incorporating code review exercises. First, our quantitative analysis showed that overall, usability was good in all conditions. Similarly, all conditions led to positive learning gains. These findings imply that the learning scenario in and of itself was both usable and effective. Nevertheless, there were no significant differences between the conditions for either usability or learning gains. Second, while descriptive results suggest that engagement was higher in the instructor condition, no significant differences were observed between conditions. This lack of significant differences across conditions for all of the aspects considered could be in part due to our limited sample size. As illustrated in Figure 6, the presence of outliers within various condition/aspect combinations could have affected our ability to draw conclusions from our results. Future work aims to further explore our research questions with a larger group of students.

Moreover, our findings could suggest that the differences across conditions, which were limited to the three code snippets used in the *Examples* phase of the lesson, were not distinct enough, and therefore resulted in a very similar learning experience. Indeed, the qualitative analysis we conducted sheds some light on this matter. First, it was reported that the exercises were repetitive. This perception of repetitiveness could have resulted in an adaptation effect, diminishing the impact of the treatments. Second, students expressed disappointment at the absence of feedback provided during the lesson. These remarks point to a possible explanation for the non-significant results across conditions. Namely, that the mere use of identities (human or chatbot) to present static information, adds very little to the learning experience, at least within the scope of our learning scenario. Instead, students expect this type of scenario to include feedback and interactivity. While instructors might not be able to cope with giving real-time feedback to large numbers of students, chatbots following rule-based scripts could serve as a first layer of interaction. A promising next step for our research would be to build on these results and assess the role that interactive chatbots could play within exercises simulating code reviews.

## VII. CONCLUSIONS

In this paper, we addressed the feasibility of simulating the code review process—widely used on social coding platforms—within an online learning environment. We also explored the role chatbots could play in supporting a lesson on software engineering. Our approach was to first design a simple, configurable code review application that could serve as a way to introduce students to best practices in software engineering and software development processes often encountered in industry. We then equipped our application with an interface to enable instructors to impersonate chatbots when presenting content or interacting with their students. This code review application was integrated into an online lesson on code

linting and used to evaluate three ways of explaining linting errors presented via example code snippets. This evaluation was conducted through a controlled in-class experiment with 30 undergraduate software engineering students.

The code review application developed for this study provides a proof-of-concept validation that functionalities from social coding platforms can be simulated within online learning environments. This application could indeed serve as an entry point for instructors looking to incorporate an environment simulating social coding platforms into their practice. Nevertheless, findings from our evaluation suggest that there are no significant differences between the three conditions we explored—explaining concepts (i) via a user representing the course instructor, (ii) via a chatbot alias, or (iii) in the accompanying text—with respect to perceived usability, engagement, and learning gains. However, suggestions obtained through open-ended responses provide key insights into the possible need for interactivity and feedback in order to better support students in this type of learning scenario. We aim to build on these findings to equip our chatbot identities with interactivity and evaluate them in future work.

## ACKNOWLEDGMENT

Images used in this study include icons made by Vector Stall, Nadiinko, and Flat Icons (available on Flaticon<sup>2</sup>), as well as a photograph by James Harrison (available on Unsplash<sup>3</sup>).

## REFERENCES

- [1] J. Quiroga Pérez, T. Daradoumis, and J. M. Marquès Puig, “Rediscovering the Use of Chatbots in Education: A Systematic Literature Review,” *Computer Applications in Engineering Education*, vol. 28, no. 6, pp. 1549–1565, 2020.
- [2] P. Smutny and P. Schreiberova, “Chatbots for Learning: A Review of Educational Chatbots for the Facebook Messenger,” *Computers & Education*, vol. 151, 2020.
- [3] M. Wessel, B. M. de Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, “The Power of Bots: Characterizing and Understanding Bots in OSS Projects,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, 2018.
- [4] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, “Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository,” in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12)*. ACM, 2012, pp. 1277–1286.
- [5] C. Lebeuf, M.-A. Storey, and A. Zagalsky, “Software Bots,” *IEEE Software*, vol. 35, no. 1, pp. 18–23, 2018.
- [6] A. Zagalsky, J. Feliciano, M.-A. Storey, Y. Zhao, and W. Wang, “The Emergence of GitHub as a Collaborative Platform for Education,” in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 2015, pp. 1906–1917.
- [7] X. Li and C. Prasad, “Effectively Teaching Coding Standards in Programming,” in *Proceedings of the 6th Conference on Information Technology Education (SIGITE '05)*. ACM Press, 2005, p. 239.
- [8] Y. Wang, H. Li, Y. Feng, Y. Jiang, and Y. Liu, “Assessment of Programming Language Learning Based on Peer Code Review Model: Implementation and Experience Report,” *Computers & Education*, vol. 59, no. 2, pp. 412–422, 2012.
- [9] M. Wessel, A. Serebrenik, I. Wiese, I. Steinmacher, and M. A. Gerosa, “Effects of Adopting Code Review Bots on Pull Requests to OSS Projects,” in *Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020.

<sup>2</sup>Flaticon: flaticon.com

<sup>3</sup>Unsplash: unsplash.com



- [10] J. Fiksel, L. R. Jager, J. S. Hardin, and M. A. Taub, "Using GitHub Classroom To Teach Statistics," *Journal of Statistics Education*, vol. 27, no. 2, pp. 110–119, 2019.
- [11] J. Feliciano, M.-A. Storey, and A. Zagalsky, "Student Experiences Using GitHub in Software Engineering Courses: A Case Study," in *Proceedings of the 38th International Conference on Software Engineering Companion*. ACM, 2016, pp. 422–431.
- [12] C. Hsing and V. Gennarelli, "Using GitHub in the Classroom Predicts Student Learning Outcomes and Classroom Experiences: Findings from a Survey of Students and Teachers," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 2019, pp. 672–678.
- [13] L. Haaranen and T. Lehtinen, "Teaching Git on the Side: Version Control System as a Course Platform," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2015, pp. 87–92.
- [14] O. G. Glazunova, O. V. Parhomenko, V. I. Korolchuk, and T. V. Voloshyna, "The Effectiveness of GitHub Cloud Services for Implementing a Programming Training Project: Students' Point of View," *Journal of Physics: Conference Series*, vol. 1840, no. 012030, 2021.
- [15] A. Bacchelli and C. Bird, "Expectations, Outcomes, and Challenges of Modern Code Review," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 712–721.
- [16] D. A. Trytten, "A Design for Team Peer Code Review," *ACM SIGCSE Bulletin*, vol. 37, no. 1, pp. 455–459, 2005.
- [17] X. Li, "Incorporating a Code Review Process into the Assessment," in *Proceedings of the 20th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2007)*, N. Bridgeman and S. Mann, Eds., 2007, pp. 125–131.
- [18] M. Tang, "Caesar: A Social Code Review Tool for Programming Education," Master's thesis, Massachusetts Institute of Technology, 2011.
- [19] Z. Kubincová and M. Homola, "Code Review in Computer Science Courses: Take One," in *Advances in Web-Based Learning – ICWL 2017*, ser. Lecture Notes in Computer Science, H. Xie, E. Popescu, G. Hancke, and B. Fernández Manjón, Eds., vol. 10473. Springer, 2017, pp. 125–135.
- [20] Z. Kubincová and I. Csicsolová, "Code Review in High School Programming," in *Proceedings of the 2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*. IEEE, 2018.
- [21] T. D. Indriasari, A. Luxton-Reilly, and P. Denny, "A Review of Peer Code Review in Higher Education," *ACM Transactions on Computing Education*, vol. 20, no. 3, 2020.
- [22] V. Balachandran, "Reducing Human Effort and Improving Quality in Peer Code Reviews Using Automatic Static Analysis and Reviewer Recommendation," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 931–940.
- [23] G.-J. Hwang and C.-Y. Chang, "A Review of Opportunities and Challenges of Chatbots in Education," *Interactive Learning Environments*, 2021.
- [24] S. Tegos, S. Demetriadis, P. M. Papadopoulos, and A. Weinberger, "Conversational Agents for Academically Productive Talk: A Comparison of Directed and Undirected Agent Interventions," *International Journal of Computer-Supported Collaborative Learning*, vol. 11, no. 4, pp. 417–440, 2016.
- [25] A. Følstad and P. B. Brandtzaeg, "Users' Experiences with Chatbots: Findings from a Questionnaire Study," *Quality and User Experience*, vol. 5, no. 1, 2020.
- [26] K. E. Wiegers, *Peer Reviews in Software: A Practical Guide*. Addison-Wesley, 2002.
- [27] J. Gruber and A. Swartz, "Markdown," 2004. [Online]. Available: [daringfireball.net/projects/markdown/](http://daringfireball.net/projects/markdown/)
- [28] A. P. Chaves and M. A. Gerosa, "How Should My Chatbot Interact? A Survey on Human-Chatbot Interaction Design," *International Journal of Human-Computer Interaction*, 2020.
- [29] M. A. Ferman Guerra, "Towards Best Practices for Chatbots," Master's thesis, University of Victoria, 2018.
- [30] S. C. Johnson, "Lint, A C Program Checker," Bell Laboratories, Tech. Rep., 1978.
- [31] N. Dahlbäck, A. Jönsson, and L. Ahrenberg, "Wizard of Oz Studies—Why and How," *Knowledge-Based Systems*, vol. 6, no. 4, pp. 258–266, 1993.
- [32] D. Gillet, "Personal Learning Environments as Enablers for Connectivist MOOCs," in *2013 12th International Conference on Information Technology Based Higher Education and Training (ITHET)*. IEEE, 2013.
- [33] N. C. Zakas, "ESLint," 2013. [Online]. Available: [eslint.org](http://eslint.org)
- [34] K. F. Tómasdóttir, M. Aniche, and A. van Deursen, "The Adoption of JavaScript Linters in Practice: A Case Study on ESLint," *IEEE Transactions on Software Engineering*, vol. 46, no. 8, pp. 863–891, 2020.
- [35] J. C. Farah, J. Soares Machado, P. Torres da Cunha, S. Ingram, and D. Gillet, "An End-to-End Data Pipeline for Managing Learning Analytics," in *Proceedings of the 19th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2021.
- [36] J. Brooke, "SUS: A 'Quick and Dirty' Usability Scale," in *Usability Evaluation In Industry*, 1st ed. CRC Press, 1996.
- [37] K. Charmaz, *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. Sage, 2006.
- [38] A. Bangor, P. T. Kortum, and J. T. Miller, "An Empirical Evaluation of the System Usability Scale," *International Journal of Human-Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.