ISSN: 2473-4901
v8 n5752

A Chatbot for Teaching Secure Programming

James Walden
waldenj1@nku.edu

Nicholas Caporusso
caporusson1@nku.edu

Ludiana Atnafu
atnaful1@nku.edu

Department of Computer Science
College of Informatics
Northern Kentucky University
Highland Heights, KY (USA)

Abstract

Even experienced developers find it difficult to always write secure code. However, students and people who are learning to program in a language or environment for the first time need additional guidance to help them understand and learn how to use secure code. To this end, we created a chatbot with an authoritative knowledge base on secure programming to help teach student developers. We designed, implemented, and evaluated a novel chatbot with a knowledge base covering secure programming in PHP using the Rasa framework. In this paper, we present an experiment in which we evaluated user experience with the chatbot and compared it to other information sources, such as question and answer sites. Participants solved secure web programming problems in a custom web application developed for the experiment with the aid of either the chatbot or their choice of Internet resources.

We found that students interacted with the chatbot throughout the experiment more than with other information sources to learn about security topics and solved web programming challenges. Although the perceived performance of the chatbot was lower than other systems, such as search engines, its low effort was ranked as a higher factor for adoption. Furthermore, although search engines and developer communities provide materials that were perceived as more accurate, users reported that screening resources requires additional effort in addition to the uncertainty of the quality. Also, responses about the overall user experience suggest that the chatbot can be utilized as a convenient support tool.

Keywords: Secure programming, software security, chatbots, user experience.

1. INTRODUCTION

Novice programmers and students learning web development are faced with the problem of acquiring multiple skills simultaneously, including writing source code in a new programming language, using a new development environment, and ensuring that their code is secure. Unfortunately, security often is considered among the least important skills by students because the impact of less secure code is not immediately evident to them: while it is easy to test an expected feature in a web application (e.g., adding an item to its database), students need additional knowledge and skills to verify that the application does so securely.

Even if security is considered equal in importance to functional requirements, it can be difficult to learn secure programming due to the problem of identifying accurate sources of information about security, the difficulty of using Application Programming Interfaces (APIs) securely (Green and Smith, 2016; Oliveira et al., 2018), the complexity of testing security flaws in software (Tahaei and Vaniea, 2019), and the evolution of new types of vulnerabilities in web applications (Hiesgen et al., 2022). While insecure information is often associated with online sources like Stack Overflow (Fischer et al., 2017), even college textbooks contain insecure code examples ("College Software Texts Found To Teach Insecure Coding", 2008).

To help students learn secure web development, we created a chatbot to answer their secure programming questions. The chatbot has a curated knowledge base with accurate information and secure code snippets for web programming in PHP and for connecting to and querying a MySQL database. Designing a chatbot specific to our web programming course enabled us to address both problems associated with learning secure programming. The knowledge base was focused on exactly the security APIs that students encountered in their class, and it was created with accurate information about security issues.

The chatbot was initially introduced in a web development course in the Spring semester of 2021, when we designed an initial experiment and collected data about the overall design of the tool and its integration within a custom website that was used as a learning and development environment. Data from our first study were utilized to evaluate the appropriateness of the system, improve the knowledge base (e.g., add code snippets), and improve the learning environment and its integration into the course.

In this paper, we present the results of a study that was realized in Fall 2021, when the revised version of the chatbot and learning environment was tested with a group of students enrolled in a web programming course. Previous studies (Abd-Alrazaq et al., 2020) found that user experience is one of the key adoption factors of chatbots. Therefore, in our experiment, we focused on evaluating the overall user experience of the system based on the dimensions defined in the Unified Theory of Acceptance and Use of Technology (UTAUT) (Venkatesh et al., 2003).
The contributions of our work are twofold:
The development of a chatbot to support learning secure programming practices in PHP.
The evaluation of how effective the chatbot is in helping students write secure code.

2. RELATED WORK

Information Sources for Learning Secure Coding
Acar et al. studied the effect of developers' use of different information sources on the functionality and security of the code they produced (Acar et al., 2016). The authors divided developers in their experiment into four groups. The first three groups had access to single sources of information: books only, official Android documentation only, and Stack Overflow only, while the fourth group had free choice of information sources. Developers restricted to only use Stack Overflow produced significantly less secure code than developers using the official documentation or books. However, developers using only official documentation produced significantly less functional code than those using only Stack Overflow.

Stack Overflow is the most popular question and answer site for software developers, including students. Multiple studies have found insecure answers and code snippets in answers for questions on a variety of programming languages and environments on the site (Chen et al., 2019; Fischer et al., 2017; Meng et al., 2018; Verdi et al., 2020). One study found that insecure answers received more up votes, comments, favorites, and views than secure answers (Chen et al., 2019).

Automated Tools for Learning Secure Coding
Automated tools can also make it easier for instructors to incorporate cybersecurity into their classes and can provide knowledge and feedback at the precise point in time when students need that. While there are a variety of tools used to assist developers in finding security vulnerabilities through static or dynamic analysis, there are few automated tools designed to help teach students about secure programming. CrypTool has been widely used to assist in the teaching of cryptography (Adamovic et al., 2018), but the focus of the tool is teaching how cryptography works rather than teaching how to write code to securely use cryptographic APIs. CryptoExplorer is a web search application that can provide insecure and secure examples of cryptographic API use, but it is aimed at professional developers (Hazhirpasand et al., 2020).

Plugins for Integrated Development Environments (IDEs) can provide secure programming assistance to students in the same environment in which they are writing their code. Whitney et al. incorporated secure Java web programming instruction into an Eclipse plugin called Educational Security in the Integrated Development Environment (ESIDE) (Whitney et al., 2018). ESIDE adds warning icons in Eclipse when problematic code patterns are detected. When students click on the warning, ESIDE provides multiple information options with short explanations and a link to a page that provides a detailed explanation of the potential security issue. ESIDE was based on an earlier plugin, ASIDE, created for professional developers (Xie et al., 2011). Nguyen et al. created a plugin to help professional developers write secure mobile code in Android Studio called FixDroid (Nguyen et al., 2017). While FixDroid was not designed for educational purposes, it would be used for that purpose.

Chatbots for Teaching and Learning
The use of chatbots in education for a wide variety of purposes from providing deadlines to delivering course content is rapidly expanding Okonkwo and Ade-Ibijola, 2021. A recent survey of chatbots in education found that chatbots serve in four pedagogical roles: learning, assisting, and mentoring (Wollny et al., 2021). The focus of this study is on the learning role. Educational chatbots have been used to help students learn a variety of skills, including computer programming. Both Python-bot Okonkwo and Ade-Ibijola, 2020 and APIHelper (Zhao et al., 2020) were designed to help students learn how to program.

Evaluation measures for tool adoption
Security tools generally see poor adoption by professional developers, who usually prefer to look up information on the Internet, by visiting developer communities (e.g., Stack Overflow) (Tahaei and Vaniea, 2019), tutorials, and, more recently, videos on YouTube (MacLeod et al.), 2015. Xiao et al. interviewed professional developers, exploring how security tool adoption was affected by social environments and communication channels (Xiao et al., 2014). They used the diffusion of innovation theory to evaluate the role of social influence and found that dynamics such as acceptance within the developer community are among the leading factors that promote the adoption of tools that address security.

Previous studies focusing on the adoption of chatbots in healthcare (Abd-Alrazaq et al., 2020) and finance (Sugumar and Chandra, 2021) found that user experience is a key factor and outlined a variety of technical measures that could be used to assess users' willingness to employ conversational agents. Almahri et al. (Almahri et al., 2020) utilized a revised version of the UTAUT (i.e., UTAUT2), to analyze the specific user dynamics that affect the acceptance, adoption, and use of chatbots in universities in the United Kingdom. They found the performance of the chatbot to be the main predictor of the behavioural intention to use this type of technology. Furthermore, severa authors (Sugumar and Chandra, 2021 and Ling et al., 2021) highlighted that when users know that they are entertaining a conversation with a chatbot, their interaction tends to be more opportunistic and utilitarian with respect to their goal and less influenced by aspects that are more typical of a conversation with a human agent (e.g., empathy).

3. DESIGN AND IMPLEMENTATION OF THE CHATBOT

We developed a chatbot to assist students to write secure code in their course on web application development. The purpose of the chatbot was to provide an authoritative source of correct information on secure programming that was also easy to use. The server-side programming language used in the course was PHP, so we wrote all code examples in PHP. The chatbot was deployed as a web widget in a custom web application that presented web application security problems for students to solve.

The chatbot was based on the Rasa chatbot framework, an open-source project written in Python. Rasa includes both natural language understanding and dialog management capabilities. The two major tasks in creating a chatbot are designing conversation flow, creating a knowledge base, and training the bot to associate questions with the correct answers in the knowledge base. The conversation flow was simple for the secure programming chatbot, tying single questions to single answers.

During our development and testing process, the Rasa framework changed rapidly, including both API and data format changes. After starting development using version 1 of Rasa, we found it impossible to deploy the chatbot to new machines, because it became impossible to install the required dependencies from our saved conda environment. This combination of dependency problems and lack of security updates for Rasa 1.x led us to update the bot's code and data to use version 2 of the Rasa framework before performing the experiment.

We designed the secure coding knowledge base to include aspects of secure coding that were directly relevant to the topics students were learning in the web development course, including authentication, input validation, cross-site scripting, and SQL injection. In addition to answering conceptual questions, the bot could also provide code examples when asked. For example, one answer included example code showing students how to perform SQL queries with prepared statements. Figuring out how to include code snippets in Rasa's data files required some trial and error, as the documentation did not support this use case and the data file format changed from JSON to YAML between versions 1 and 2 of the framework.

The final step to creating the secure programming bot was training it to answer secure programming questions. The authors interacted with the bot repeatedly, asking the same questions in a variety of ways to build the initial version of the bot. Once the bot was working, we focused training on teaching the bot to distinguish between similar questions. For example, password security could refer to HTML form input fields, transmitting passwords over HTTPS, or securely storing passwords in a database. While training data for most question and answer pairs consisted of a couple dozen example questions, training data for related topics required approximately twice as many examples to ensure the bot could reliably provide the desired answer.

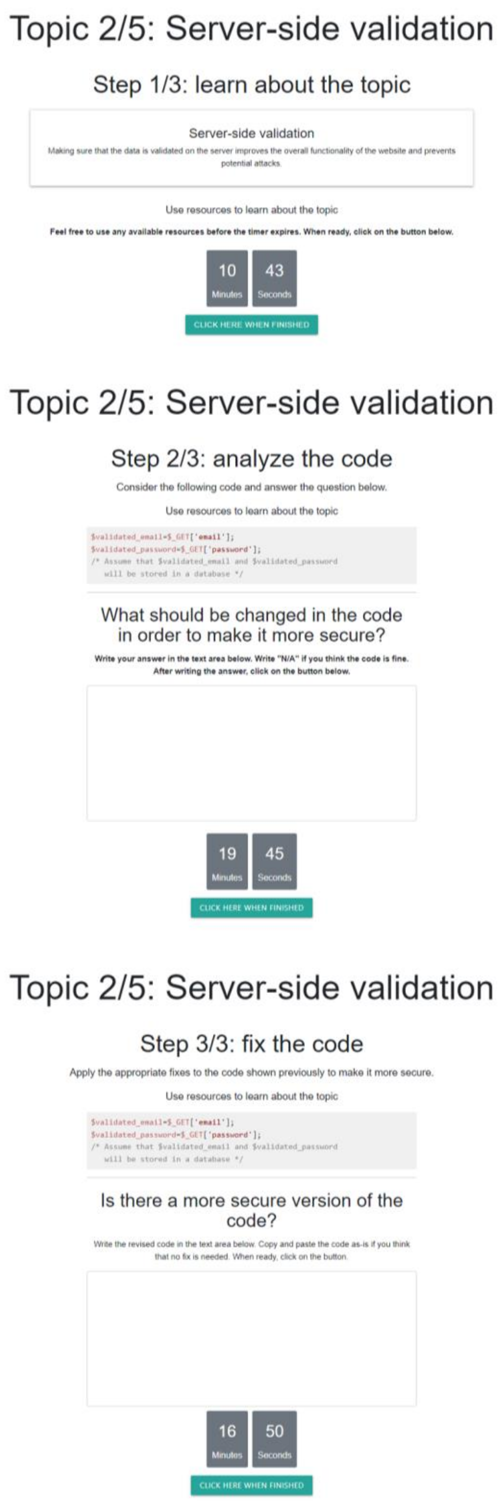
4. EXPERIMENTAL STUDY

Materials and methods

We realized an experimental study that evaluated perceived user experience and effectiveness of the chatbot in supporting students and beginner programmers in learning key cybersecurity concepts in client- and server-side web development, including well known security issues in web applications, such as cross-site scripting and SQL injection.

To this end, we designed a custom web application in which users could interact with the chatbot while practicing with code challenges consisting in analyzing and fixing existing code snippets containing cybersecurity flaws. Screenshots of the web application can be seen in Figure 1.

The web application contains five challenges, each addressing a key cybersecurity problem in a web authentication workflow. Participants were required to complete all five challenges.



Topic 2/5: Server-side validation

Step 1/3: learn about the topic

Server-side validation

Making sure that the data is validated on the server improves the overall functionality of the website and prevents potential attacks.

Use resources to learn about the topic

Feel free to use any available resources before the timer expires. When ready, click on the button below.

Topic 2/5: Server-side validation

Step 2/3: analyze the code

Consider the following code and answer the question below.

Use resources to learn about the topic

```
$validated_email=$_GET['email'];
$validated_password=$_GET['password'];
/* Assume that $validated_email and $validated_password
   will be stored in a database */
```

What should be changed in the code in order to make it more secure?

Write your answer in the text area below. Write "N/A" if you think the code is fine. After writing the answer, click on the button below.

Topic 2/5: Server-side validation

Step 3/3: fix the code

Apply the appropriate fixes to the code shown previously to make it more secure.

Use resources to learn about the topic

```
$validated_email=$_GET['email'];
$validated_password=$_GET['password'];
/* Assume that $validated_email and $validated_password
   will be stored in a database */
```

Is there a more secure version of the code?

Write the revised code in the text area below. Copy and paste the code as-is if you think that no fix is needed. When ready, click on the button.

Figure 1 The experiment website

- Front-end and web forms: use of correct input fields to prevent over- the-shoulder attacks when typing a password; HTTP requests and client- server communication (e.g., use of correct and secure HTTP methods and protocols to prevent man-in-the-middle attacks or information leaks).

- Server-side data processing: proper handling of data submitted via HTML forms to prevent missing input and code injection attacks.

- Password security: secure password validation and storage using hashing algorithms.

- SQL injections: use of prepared statements and other mechanisms for preventing potential database attacks.

- Cross-site scripting: use of systems for preventing phishing attacks and injection of scripts and snippets.

For each challenge, the website provided participants with a description of the topic that the challenge focused on and a small piece of source code that contained security flaws.

Subjects were required to complete an experimental task organized into three parts as follows:

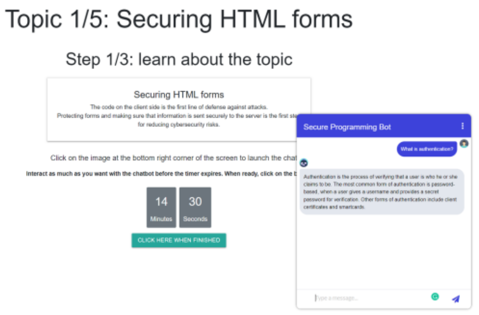
- Topic and code review: participants were invited to learn more about the topic and analyze the content of the code snippet.

- Code analysis: subjects were asked to identify and submit a short report in which they described the security flaws.

- Bug fixing: the experimental software showed the original snippet and provided subjects with an editor in which they could write a revised version of the source code.

Before starting the experiment, subjects were asked to complete a pre-survey to collect information about the participants, including their experience with cybersecurity and web development. Participants were given a maximum of 15 minutes to complete each of the three sections (i.e., 45 minutes total for one challenge). During this time, they could work on each of the three parts of the section with the help of either the chatbot or other information resources.

Participants were split into experiment and control groups as follows. Group A was provided with the chatbot for challenges one, three, and five, whereas Group B was provided with the chatbot for challenges two and four (see Figure 2). By diving the participants into two groups, we provided subjects with the opportunity of using the chatbot as well as other resources in the experimental sections. Conversely, subjects did not get access to the chatbot in control sections. By doing this, they could compare their learning and programming experience and evaluate the value of the chatbot as a learning tool.


Topic 1/5: Securing HTML forms

Step 1/3: learn about the topic

Securing HTML forms

The code on the client side is the first line of defense against attacks.
Protecting forms and making sure that information is sent securely to the server is the first ste
for reducing cybersecurity risks.

Click on the image at the bottom right corner of the screen to launch the chat

Interact as much as you want with the chatbot before the timer expires. When ready, click on the b

Secure Programming Bot

What is authentication?

Authentication is the process of verifying that a user is who he or she claims to be. The most common form of authentication is password-based, when a user gives a username and provides a secret password for verification. Other forms of authentication include client certificates and smartcards.

Figure 2 Chatbot widget

After participants completed each challenge, the website provided them with a short questionnaire. At the end of the entire experiment, after completing all challenges, participants were asked to evaluate their overall experience with the chatbot and to compare it with other resources they used during the experiment. Specifically, in our study, we analyzed intrinsic and extrinsic aspects that characterize user experience and the willingness to adopt and use technology. To this end, we utilized the UTAUT model, a widely adopted user experience framework that utilizes the five dimensions indicated below as predictors of the intention to adopt and use technology.

Performance expectancy. This aspect refers to the belief that the use of a particular technology will enhance the performance of an individual or will produce some advantage in realizing a task.

Effort expectancy. This is a two-fold measure: on the one hand, it refers to the perceived skillset required to be able to utilize a system and the expected learning curve (human-machine co-evolution). Simultaneously, it relates to the extent of convenience perceived in using the system.

Social influence. This component refers to user's perception of beliefs or impressions that the product will generate in others (their milieu, their social group, or the external community). This includes the ability of a device to improve the social status of an individual or to create a desired social image. Moreover, this measure involves social acceptance of technology in each context of reference.

Facilitating conditions. Extrinsic factors, such as, battery life, device compatibility, and availability of product accessories and features that render the product more versatile might be a driver for adoption. Also, presence of technical support and a user's guide might increase the likelihood of acquiring products. Switching costs and longevity are additional aspects that contribute to this dimension.

Hedonic motivation. Intrinsic factors that are not related to product experience are associated with individuals' conditions or beliefs, social background, and education. As this often is a multifaceted aspect, we included open-ended questions to elicit participants' comments and feedback.

Participants
A total of 20 individuals volunteered to participate in the experiment. Participants were recruited from a server-side web development course that taught PHP and MySQL. Subjects were aged 19-32 (21 on average), 18 were males and 2 females. Six were sophomores, eight were juniors, and six were seniors. They all had from one to three years of experience with programming, though they were not familiar with PHP and MySQL prior to the course and had never utilized a chatbot as a learning resource, though they were familiar with chatbot technology in other contexts.

5. RESULTS

Bot chat analysis
We collected 25 student conversations with the chatbot. The number of conversations is greater than the number of participants, because students could exit the chatbot in one section then restart it in a different section of the experiment application. These conversations included 305 questions, 165 questions asked by students in group A and 140 asked by students in group B. We manually analyzed the questions and their answers to determine if questions were related to secure programming and whether the bot provided relevant answers to the questions.

We found that 215 out of the 305 questions students asked the bot were related to secure programming. The remaining 30% of questions included technical questions about PHP, JavaScript, or SQL that were not related to security, greetings like ”hello”, tests of the bot like ”are you secretly a human?”, and general chit chat.

The bot answered 213 (70%) questions correctly, including both secure programming and non-programming questions like requests for information about the bot. Out of the 92 questions answered incorrectly, 26 were not questions about secure programming. Incorrect answers for questions about secure programming questions fell into four categories: nonspecific questions (10), questions about topics not in the bot's knowledge base (33), questions where the bot provided a wrong answer (22), and questions consisting solely of source code (1).

Five of the nonspecific questions were requests for more information on the question that the bot had just answered. As the bot does not retain context, it is impossible for it to answer such questions. Other nonspecific questions including asking for code examples without specifying a topic and completely open questions like ”How?” Student questions included 688 words. The bot responded to these questions with 13,893 words. The top twenty most common words used by the students and the bot are listed in Table 1, while the word cloud diagram below visualizes the frequency of student word use.

User word	User count	Bot word	Bot count
password	42	code	723
secure	34	password	521
validate	31	input	393
html	26	user	275
email	20	data	218
php	20	web	216
code	18	php	211
passwords	18	validation	183
cross	16	passwords	172
scripting	15	hash	165
site	14	email	163
forms	13	application	144
sql	12	secure	144
form	11	sql	136
server	11	output	120
xss	10	post	109
get	9	filter	108
input	9	function	108
protect	9	length	107

Table 1: Top 20 words

The 26 conversations that students had with the chatbot consisted of 688 words. The bot responded to these conversations with 13,893 words. The top twenty most common words used by the students and the bot are listed in Table 1, while the word cloud diagram below (see Figure 3) visualizes the frequency of student word use. The three most common words (password, secure, and validate) are all relevant to security queries, indicating student concerns about how to use and store passwords and how to validate user input.

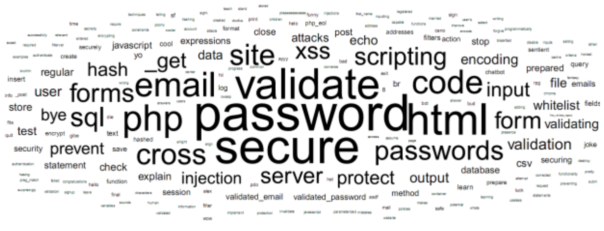


Figure 3 Word cloud of chatbot interactions

Survey Analysis

A total of 19 participants completed the experiment and responded to the surveys. One subject only finished two sections and, thus, we did not include their data in our analysis. First, we analyzed the overall perceived level of interaction with all the available resources, which is shown in Figure 4. Compound data from surveys about sections from one to five show that, when subjects were provided with it, the chatbot was the first type of resource used, as it represented 35% of the queries. Search engines were the second preferred resource, utilized in 30% of the cases. Developer communities were ranked third in terms of preference, with 17% of usage rate, followed by tutorials (9%), YouTube video (4%), other resources (3%), and books (2%). No statistically significant trends, differences, or training effects were found between individual sections, which shows that subjects did not increase or decrease the use of a specific resource throughout the experiment. Although our data show that subjects preferred to interact with the chatbot even if they were allowed to use other types of materials, students commented that they sometimes had to use additional resources to complete the challenge. This could be due to a lack of familiarity with interacting with such a chatbot and to the inherent switching cost with respect to systems that they already are familiar with and use.

Subsequently, we analyzed participants' responses with respect to the specific User Experience dimensions defined by the UTAUT model. As shown in Figure5, it is possible to identify two groups of resources based on participants' responses. Search engines, the chatbot, and developer communities received very high scores in terms of adoption metrics, with average results of 74%, 78%, and 70%, respectively. On the contrary, the other types of materials were ranked lower. Specifically, tutorials, YouTube videos, other resources, and books, had an average of 51%, 48%, 35%, and 23%, respectively.

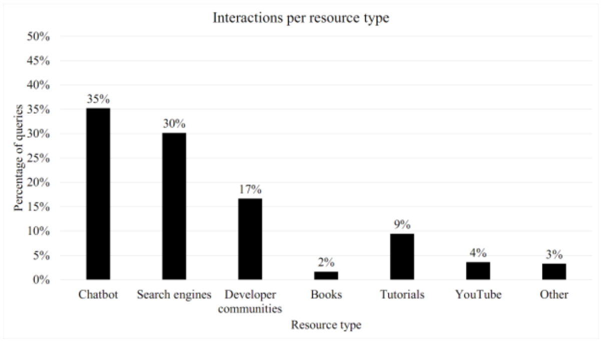


Figure 4 Interactions per resource type

Data about individual user adoption dimensions indicate that the chatbot was perceived as having a lower performance expectancy (67%) with respect to search engines (80%) and developer communities (70%). This could be due to the lack of familiarity with the system and how to query the knowledge base. Also, this could be caused by the content of the knowledge base itself, which can be improved, in the future. Tutorials, YouTube videos, other resources, and books were ranked lower, with 60%, 55%, 40%, and 14%. of preference. Based on previous studies (Almahri et al., 2020), performance expectancy is a critical aspect in the adoption and use of chatbot technology. Thus, our results suggest that further work is needed before using the chatbot more consistently in web programming courses, because its current perceived performance expectancy might be a cause of discontinuation.

As far as effort expectancy is concerned, the chatbot ranked first (87%) compared to search engines (73%), other resources (61%), developer communities (55%), YouTube videos (49%), tutorials (22%), and books (12%). Students indicated that the chatbot was the most convenient resource to gain an initial understanding of the topic. Although this could be due to the fact that the chatbot was integrated into the website, participants' comments mostly report the ease of typing questions in natural language and the responsiveness of the system, which returned either relevant results or answers that were clearly inaccurate. On the contrary, other systems require students to evaluate the response, identify the significant portion within a larger block of text or code, or filter out solutions that are imprecise or did not address the security requirements mentioned in the challenge.

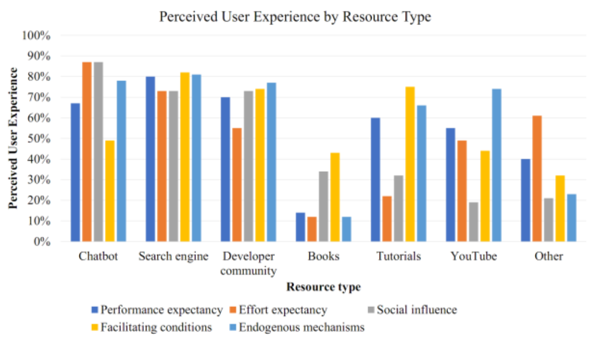


Figure 5 Perceived User Experience by Resource Type

Social influence was reported the highest in the chatbot (87%), followed by developer communities (73%), search engines (73%), books (34%), tutorials (32%), other resources (21%), and YouTube videos (19%). Comments from participants indicate that while search engines and developer communities are more socially accepted in the programming world, using other systems (e.g., YouTube videos or tutorials) could be interpreted as a lack of professionalism. Conversely, they considered the chatbot more of an innovative coding companion that could improve their professional posture as a developer.

Facilitating conditions was the lowest ranking measure for the chatbot (i.e., 49%). On the contrary, search engines scored 82%, tutorials 75%, developer communities 74%, YouTube 44%, books 43%, and other systems 32%. This may suggest and confirm the presence of a switching cost between systems that students are already familiar with and typically use for coding tasks and a novel interface. For instance, tutorials could have received a higher preference because they guide developers step by step. Thus, users' perception of performance expectancy and facilitating conditions may change if the chatbot provided students with a more sophisticated interface that offered a code snippet with a more detailed implementation example.

In terms of hedonic motivation, the chatbot (78%) was ranked as comparable to search engines (81%), developer communities (77%), and YouTube videos (74%), whereas other systems scored lower (tutorials 66%, other resources 23%, books 12%). This measure suggests that students are already motivated to incorporate chatbots into their learning and programming activities as they already do with other digital tools (i.e., search engines, developer communities, and YouTube videos). Our findings are in line with other published studies about the introduction of chatbots as a learning aid in the classroom (Mokmin and Ibrahim, 2021), which found that students' personal motivation is also a predictor of the behavioral intention to use this type of technology, though they are in contrast with other studies (Sugumar and Chandra, 2021) that found aspects such as a pleasurable conversation less important, given the opportunistic nature of the interaction with a chatbot in specific domains.

6. CONCLUSION

In this paper, we detailed the design, use, and user experience evaluation of a chatbot aimed at teaching secure programming concepts to students enrolled in web development courses. Our objective is to provide students with a user friendly learning environment that simultaneously is a reputable source of information. Previous studies in other domains (e.g., Abd-Alrazaq et al., 2020, Mokmin and Ibrahim, 2021) also evaluated whether users enjoyed the conversational aspect of chatbot. On the contrary, we primarily focused on the performance of the chatbot in providing accurate answers and on user experience metrics directly related with the goal of learning key aspects of secure programming via inductive reasoning guided by coding challenges. Quantitative data from our experiment show that the students interacted with the chatbot, which, in turn, was able to accurately address most questions. Furthermore, quantitative and qualitative data from our survey show that participants considered their user experience with the chatbot as extremely positive.

In the future, we plan to expand the bot's knowledge base to answer secure programming questions asked by students that currently have no answer. We also plan to improve the bot's training using the data provided by the students during the experiment. To help student's with requests for additional information on a question, we plan to add suggestions for additional questions that the bot can answer in answers provided by the bot.

7. REFERENCES

Abd-Alrazaq, A., Safi, Z., Alajlani, M., Warren, J., Househ, M., Denecke, K., et al. (2020). Technical metrics used to evaluate health care chatbots: Scoping review. Journal of medical Internet research, 22 (6), e18301.

Acar, Y., Backes, M., Fahl, S., Kim, D., Mazurek, M. L., & Stransky, C. (2016). You get where you're looking for: The impact of information sources on code security. 2016 IEEE Symposium on Security and Privacy (SP), 289–305.

Adamovic, S., Sarac, M., Stamenkovic, D., & Radovanovic, D. (2018). The importance of the using software tools for learning modern cryptography. International Journal of Engineering Education, 34 (1), 256–262.

Almahri, F. A. J., Bell, D., & Merhi, M. (2020). Understanding student acceptance and use of chatbots in the United Kingdom universities: A structural equation modelling approach. 2020 6th International Conference on InformationManagement (ICIM). https://doi.org/10.1109/icim49319.2020.24 4712

Chen, M., Fischer, F., Meng, N., Wang, X., & Grossklags, J. (2019). How reliable is the crowdsourced knowledge of security implementation? 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), 536–547.

College software texts found to teach insecure coding. (2008). SANS. https://web.archive.org/web/202101271720 18/ https://www.sans.org/newsletters/newsbites /x/57#313

Fischer, F., Böttinger, K., Xiao, H., Stransky, C., Acar, Y., Backes, M., & Fahl,

S. (2017). Stack Overflow considered harmful? the impact of copy&paste on Android application security. 2017 IEEE Symposium on Security and Privacy (SP), 121–136.

Green, M., & Smith, M. (2016). Developers are not the enemy!: The need for usable security APIs. IEEE Security & Privacy, 14 (5), 40–46.

Hazhirpasand, M., Ghafari, M., & Nierstrasz, O. (2020). CryptoExplorer: An interactive web platform supporting secure use of cryptography APIs. 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), 632–636.

Hiesgen, R., Nawrocki, M., Schmidt, T. C., & W¨ahlisch, M. (2022). The race to the vulnerable: Measuring the log4j shell incident. arXiv preprint arXiv:2205.02544.

Ling, E. C., Tussyadiah, I., Tuomi, A., Stienmetz, J., & Ioannou, A. (2021). Factors influencing users' adoption and use of conversational agents: A systematic review. Psychology Marketing.

MacLeod, L., Storey, M.-A., & Bergen, A. (2015). Code, camera, action: How software developers document and share program knowledge using youtube. 2015 IEEE 23rd International Conference on Program Comprehension, 104–114.

Meng, N., Nagy, S., Yao, D., Zhuang, W., & Argoty, G. A. (2018). Secure coding practices in Java: Challenges and vulnerabilities. Proceedings of the 40th International Conference on Software Engineering, 372–383.

Mokmin, N. A. M., & Ibrahim, N. A. (2021). The evaluation of chatbot as a tool for health literacy education among undergraduate students. Education and Information Technologies, 1–17.

Nguyen, D. C., Wermke, D., Acar, Y., Backes, M., Weir, C., & Fahl, S. (2017). A stitch in time: Supporting Android developers in writing secure code. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 1065–1077.

Okonkwo, C. W., & Ade-Ibijola, A. (2020). Python-Bot: A chatbot for teaching python programming. Engineering Letters, 29 (1).

Okonkwo, C. W., & Ade-Ibijola, A. (2021). Chatbots applications in education: A systematic review. Computers and Education: Artificial Intelligence, 2, 100033.

Oliveira, D. S., Lin, T., Rahman, M. S., Akefirad, R., Ellis, D., Perez, E., Bobhate, R., DeLong, L. A., Cappos, J., & Brun, Y. (2018). Api blindspots: Why experienced developers write vulnerable code. Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018), 315–328.

Sugumar, M., & Chandra, S. (2021). Do i desire chatbots to be like humans? exploring factors for adoption of chatbots for financial services. Journal of International Technology and Information Management, 30 (3), 38–77.

Tahaei, M., & Vaniea, K. (2019). A survey on developer-centred security. 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 129–138.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. MIS quarterly, 425–478.

Verdi, M., Sami, A., Akhondali, J., Khomh, F., Uddin, G., & Motlagh, A. K. (2020). An empirical study of C++ vulnerabilities in crowd-sourced code examples. IEEE Transactions on Software Engineering.

Whitney, M., Lipford, H. R., Chu, B., & Thomas, T. (2018). Embedding secure coding instruction into the IDE: Complementing early and intermediate CS courses with ESIDE. Journal of Educational Computing Research, 56 (3), 415–438.

Wollny, S., Schneider, J., Di Mitri, D., Weidlich, J., Rittberger, M., & Drachsler,

H. (2021). Are we there yet?-a systematic literature review on chatbots in education. Frontiers in artificial intelligence, 4.

Xiao, S., Witschey, J., & Murphy-Hill, E. (2014). Social influences on secure development tool adoption: Why security tools spread. Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing, 1095–1106.

Xie, J., Chu, B., Lipford, H. R., & Melton, J. T. (2011). ASIDE: IDE support for web application security. Proceedings of the 27th Annual Computer Security Applications Conference, 267–276.

Zhao, J., Song, T., & Sun, Y. (2020). Apihelper: Helping junior android programmers learn api usage. IAENG International Journal of Computer Science, 47 (1), 92–97.