Evaluation of AI Models to Update Cybersecurity Curriculum

Chizoba Ubah
Department of Computer and Information Sciences
Towson University
Towson, MD, USA
cubah@towson.edu
0009-0006-7345-1074

Paige Zaleppa
Department of Computer and Information Sciences
Towson University
Towson, MD, USA
pzaleppa@towson.edu
0009-0002-6120-3787

Blair Taylor
Towson University
7800 York Rd., Towson, MD, USA, 21252
410-704-4560
btaylor@towson.edu
0000-0002-4708-5852

Siddharth Kaza
Towson University
7800 York Rd., Towson, MD, USA, 21252
410-704-2633
skaza@towson.edu
0000-0002-9561-6128

***Abstract*—This study explores the performance of several Large Language Models (LLMs) across different facets of Cybersecurity Modules. Using prompt engineering, this work evaluates publicly available LLMs for their ability to assess the suitability of secure coding topics based on learning outcomes, categorize these topics following OWASP standards, and generate up-to-date examples for curriculum use. The findings would highlight the transformative role that LLMs would play for future advancements in Cybersecurity education.**

Keywords—Artificial Intelligence, Curriculum Relevance, Cybersecurity Education, Large Language Model, Prompt Engineering

I. INTRODUCTION

The rapidly evolving landscape of cybersecurity necessitates a critical examination of the relevance of curriculum being used and its effectiveness at preparing students to tackle emerging threats. The interdisciplinarity of cybersecurity is subject not only to the changing nature of technology and cyber, but also to rapid and substantive changes within its various sectors, such as aviation, energy, and biotech, to name a few [1]. The reliance on published textbooks has waned for fast moving fields and the development of curriculum often occurs at a more modular level. Revising a lab, a module, or an entire course takes time that includes research, gap identification, and realignment to curricular standards [2]. Revisions can include changes to assessments, hands-on experiences, examples, learning outcomes, as well as many other facets of curriculum.

Recent advancements in Large Language Models (LLMs) have allowed for unprecedented improvements in a variety of language-related tasks that present a unique opportunity in curriculum development for fast moving fields such as cybersecurity. Large language models have demonstrated potential to assist in lesson planning, assessment generation, and content creation at a variety of educational levels using prompt engineering [3] [4] [5]. The focus of this study was to analyze the results provided by publicly available LLMs for prompts developed to determine appropriateness of level for secure coding topics given learning outcomes, classification of topic to OWASP standards, and generation of recent examples to be used in the curriculum.

II. BACKGROUND

A. Large Language Models

In the domain of artificial intelligence, complex deep learning algorithms, commonly referred to as large language models (LLMs), are mostly trained with extensive datasets. These models are notable for their exceptional adaptability across various tasks. With just a few human-readable queries or inputs, they can generate remarkably accurate responses or predictions. These computational frameworks are notable for generating content that responds to text-based instructions [6].

B. Prompt Engineering

Prompt engineering is the process of creating a request that produces the most effective performance on the required task [19]. Zero-shot, Few-Shot and Zero-shot Chain of thought (CoT) prompts were considered during this study but we prioritized the use of Zero-Shot prompting because it has been established that this approach can produce results without the need for additional domain-specific data collection or model finetuning [7] [8], also because we want to better assess the "intrinsic" capability of these LLMs [9], considering Zero shot-learning refers to prompts where no specific example is given, so we could have LLMs that respond to a broad range of requests without explicit training, often through prompts, although answer accuracy varies [10] [6].

C. Selection of Models

There are many options of large pretrained "foundation" [7] models to choose from, but our experiments in this study make use of models and experimental units (facets) that are publicly available to the community. All pretrained models are used without any further fine-tuning, experimentation may be conducted on a single machine with internet access for external API calls. GPT-3.5 turbo by OpenAI; PaLM 2 by Google's Palm API and Llama-2-7b by Meta AI. These LLMs were used because their performance has been evaluated across several benchmark categories like commonsense reasoning, code, world knowledge, experimentation and reading comprehension [11] [12] [13].

TABLE I. DIFFERENT MODELS USED FOR THIS EXPERIMENT

	Models		
	Open AI	***Meta AI***	***Google (PaLM 2)***
Pretrained Model	GPT 3.5Turbo	llama-2-7b	text-bison@001
Temperature	0	0.6	0.0
Max_tokens	Unspecified	1500	500

We adopted the CLEAR framework for Prompt Engineering that emphasizes five essential components that prompts should adopt, which are namely: Concise, Logical, Explicit, Adaptive, and Reflective; we developed prompts that had brevity and clarity; were well structured and coherent with clear output specifications [14].

III. EXPERIMENTAL DESIGN

Using learning materials developed to introduce secure coding concepts to CS0, CS1, and CS2 students from the Security Injections @ Towson project (https://clark.center/collections/secinj) [15] our experiment focused on analyzing results of zero-shot prompting for various facets of open-source curriculum across different LLMs. These modules are hosted online and require about 20 minutes to complete either in-class or asynchronously. They are used by professors to introduce secure coding concepts such as integer error, buffer overflow, encapsulation, input validation and others. Fig. 1 shows an example security injection module used in this study's experiments [16] [17].

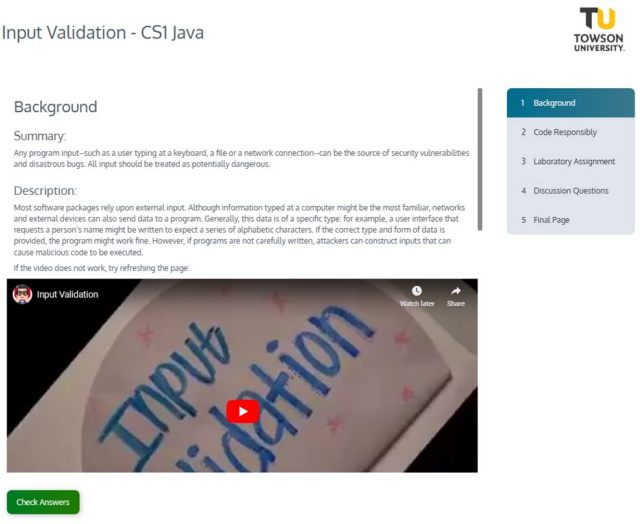

Fig. 1. A Security Injection Module to introduce input validation in a CS1 Java course

We focused on facets of the security injections learning modules that are common across curriculum.

A. Title and Learning Outcomes

Using the title and learning outcomes of these modules allows for the models to assess the appropriateness to teach a topic at a given level as well as categorize them into an OWASP Top Ten category. The assumption is that if the learning material cannot be classified into a OWASP category, it is likely not relevant anymore. The input prompts used in this section were:

1) *Given the course title and learning outcomes, which OWASP Top ten category is most applicable.*
2) *Is it appropriate to teach this course title and learning outcomes to a freshman or sophomore?*

B. Real-World Examples

The input prompt used in this section was to analyze the real-world example(s) used in the modules. The model is required to:

1) *Check how current and relevant the real-world example is.*
2) *If there are newer examples of an incident related to the topic and learning outcome(s), provide a more recent and relevant example with a link or citation to the source of your information.*

C. Code Examples

The input prompt used in this section is expected to check the code provided for

1) *Syntax Correctness.*
2) *Programming language version used (if not specified, please specify).*
3) *Verify if the version is the newest available for that programming language.*
4) *Check if the provided title and learning objectives are related to this code.*
5) *Determine if it is accurate to use that code example in the context of that programming language.*

The categories of prompts used for the sample security injection module include Classification, Generation, Question Answering, and Inference [18].

During the experiment, for prompt improvement, we performed question refinement and iterative prompt development. The input semantics used were contextual in specifying what each prompt refers to. For context, we used the Input Validation – CS1-Java module, the OWASP benchmark alignment and the currency definer which looks to check for a more recent example of a particular security incident as it relates to the topic provided. The temperature, which measures the degree of variability to expect in each output, was set to 0 for GPT 3.5turbo, 0.0 for PaLM 2 model and 0.6 which is the default for llama-2-7b.

The results of the LLMs were validated by two professors who have taught Cybersecurity with over 20 years of experience.

IV. RESULTS AND DISCUSSION

This section presents the results of four experiments with three AI models on the Input Validation Java module used to teach CS1 students. The title and the learning outcomes of the module (shown in Table II) were provided as input to all models. From results obtained with the Llama 2 model, there were instances of irregular formatting and coherence issues. These anomalies were noted and factored into the subsequent analysis, ensuring that conclusions drawn were based on accurate and coherent information.

TABLE II. TITLE AND LEARNING OUTCOMES GIVEN AS INPUT

Course Title: Input Validation – CS1 – Java Learning Outcomes:
• Demonstrate input validation to check range and/or reasonableness in code.
• Compare whitelist and blacklist input validation strategies.
• Recognize that all input to a program is potentially dangerous.
• Identify the checks that are needed for a particular kind of input.

A. Experiment 1: Alignment with OWASP standards

As shown in Appendix Fig. 2 and Fig. 3, the models successfully classified the learning material under the "Injection" category of the OWASP Top Ten which implies that the material is still relevant to the current categories. In contrast, as depicted in Appendix Fig. 4, Llama 2 delivered a broad overview of multiple OWASP Top Ten categories and provided references and links to topics that were not inquired about.

B. Experiment 2: Appropriateness for Level (Freshman or Sophomore)

As shown in Appendix Fig. 5, 6 and 7, all models confirmed that it is appropriate to teach Input Validation for CS1 Java to a Freshman or Sophomore based on the learning outcomes provided. “Appropriateness” is equated to the difficulty level of introducing some of these fundamental concepts. More information may need to be given to the models to further specify the terms freshman or sophomore.

C. Experiment 3: Real-World Examples Higher resolution available

As shown in Table III and Table IV, each model provided a consistent response for the assessment of how relevant and up to date the real-world examples were in correlation to the course's title and learning outcomes. While generating newer examples, Palm_2 did not provide a citation to the source of its information; However, GPT 3.5 turbo included verifiable citations. In contrast, the Llama 2 model as seen in Appendix Fig. 8, provided additional context and examples but didn't fully adhere to the specific tasks outlined in the prompt. This could happen sometimes with AI models, where the response, while related to the subject matter, diverges from the specific query or task presented. Human assessment was employed to verify the accuracy of the AI-generated output. This suggests that AI can produce relevant examples for various topics in curriculum development, yet its ability to provide up-to-date information may not be equally reliable.

TABLE III. GPT 3.5 TURBO OUTPUT FOR REAL-WORLD EXAMPLES

Output
1. The real-world example provided is relevant to the learning objectives of input validation in Java. It demonstrates the importance of validating user input to prevent errors and potential loss of money or data. 2. There may be newer examples of incidents related to input validation, but without specific information about the timeframe of the course or the desired level of relevance, it is difficult to provide a more recent example. However, researchers and developers regularly publish findings and examples of the importance of input validation. One such example is the "Equifax Data Breach" that occurred in 2017, where hackers exploited a vulnerability in an unpatched software component that allowed them to gain unauthorized access to sensitive personal information of nearly 147 million people. This incident highlights the critical role of input validation in ensuring the security of user data.
Source: Krebs, B. (2017). The Equifax Breach: What You Should Know. Krebs On Security. Retrieved from https://krebsonsecurity.com/2017/09/the-equifax-breach-what-you-should-know/

TABLE IV. PALM 2 OUTPUT FOR REAL-WORLD EXAMPLES

Output
The real-world example is still relevant today, as input validation is still an important topic in computer science. However, there are newer examples of incidents related to input validation. For example, in 2017, a hacker was able to steal $10 million from a bank by exploiting a vulnerability in the bank's website. The vulnerability allowed the hacker to input a negative number into a field that was supposed to accept only positive numbers. This caused the bank's system to crash, and the hacker was able to steal the money. Another example of an input validation vulnerability occurred in 2018.

D. Experiment 4: Code Examples

As shown in Appendix Fig. 9, 10 and 11 the models generated consistent results for syntax correctness and other parameters, except that GPT 3.5 turbo required more context to determine the version of programming language used in the code snippet. In contrast, Palm 2 produced accurate results without the need for supplemental context. While Llama 2 did address several points raised in the prompt instructions, its responses veered off in terms of the specifics about the Java version and the operational testing of the code, and it included an incorrect statement about a syntax error that did not exist in the provided code snippet.

V. FUTURE WORK

Based on the results of this study, further investigations of AI models and their application to the cybersecurity curriculum would be conducted. This study focused on an

introductory topic that does not require much prerequisite knowledge; investigating curriculum with more complex topics is an area that can be explored. Using more advanced prompting techniques, to evaluate results from the same models, topics, and facets, is also of interest. With the increasing number of available LLMs, this study can also be replicated with other models to evaluate their effectiveness at responding to prompts relating to educational materials.

ACKNOWLEDGEMENTS

This work is partially supported by the NSA through the NCCP program (H9830-17-1-0405), NCAE-C program (H98230-21-1-0175).

REFERENCES

[1] N. Chowdhury and V. Gkioulos, "Cyber security training for critical infrastructure protection: A literature review," *Computer Science Review*, vol. 40, p. 100361, May 2021. DOI: https://doi.org/10.1016/j.cosrev.2021.100361

[2] L. Tychonievich and M. Sherriff , "Engineering a Complete Curriculum Overhaul," *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, vol. 1, pp. 453-459, February 2022. DOI: http://dx.doi.org/10.1145/3478431.3499287

[3] E. Kasneci, K. Sessler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel, J. Pfeffer, O. Poquet, M. Sailer, A. Schmidt, T. Seidel, M. Stadler and G. Kasneci, "ChatGPT for good? On opportunities and challenges of large language models for education," *Learning and individual differences*, vol. 103, p. 102274, 2023. DOI: http://dx.doi.org/10.1016/j.lindif.2023.102274

[4] E. Gabajiwala, P. Mehta, R. Singh and R. Koshy, "Quiz Maker: Automatic Quiz Generation from Text Using NLP," *Futuristic Trends in Networks and Computing Technologies: Select Proceedings of Fourth International Conference on FTNCT 2021*, pp. 523-533, 2022. DOI: https://doi.org/10.1007/978-981-19-5037-7_37

[5] S. Sarsa, P. Denny, A. Hellas and J. Leinonen, "Automatic Generation of Programming Exercises and Code Explanations with Large Language Models," *arXiv preprint arXiv:2206.11861*, 2022. DOI: https://doi.org/10.48550/arXiv.2206.11861

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry and A. Askell, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877-1901, 2020.

[7] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut and E. Brunskill, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021. DOI: https://doi.org/10.48550/arXiv.2108.07258

[8] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo and V. Sindhwani, "Socratic models: Composing zero-shot multimodal reasoning with language," *arXiv preprint arXiv:2204.00598*, 2022. DOI: https://doi.org/10.48550/arXiv.2204.00598

[9] J. Bommarito, M. Bommarito, D. M. Katz and J. Katz, "Gpt as knowledge worker: A zero-shot evaluation of (ai) cpa capabilities," *arXiv preprint arXiv:2301.04408*, 2023. DOI: https://doi.org/10.48550/arXiv.2301.04408

[10] H. Dang, L. Mecke, F. Lehmann, S. Goller and D. Buschek, "How to prompt? Opportunities and challenges of zero-and few-shot learning for human-AI interaction in creative applications of generative models," *arXiv preprint arXiv:2209.01390*, 2022. DOI: https://doi.org/10.48550/arXiv.2209.01390

[11] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey and Z. Chen, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023. DOI; https://doi.org/10.48550/arXiv.2305.10403

[12] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava and S. Bhosale, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023. DOI: https://doi.org/10.48550/arXiv.2307.09288

[13] J. Ye, X. Chen, N. Xu, C. Zu, Z. Shao, S. Liu, Y. Cui, Z. Zhou, C. Gong and Y. Shen, "A comprehensive capability analysis of gpt-3 and gpt-3.5 series models," *arXiv preprint arXiv:2303.10420*, 2023. DOI: https://doi.org/10.48550/arXiv.2303.10420

[14] L. S. Lo, "The CLEAR path: A framework for enhancing information literacy through prompt engineering," *The Journal of Academic Librarianship*, vol. 49, no. 4, pp. 0099-1333, 2023. DOI: https://doi.org/10.1016/j.acalib.2023.102720

[15] B. Taylor and S. Kaza, "Security injections@ towson: Integrating secure coding into introductory computer science courses," *ACM Transactions on Computing Education (TOCE)*, vol. 16, no. 4, 2016. DOI: https://doi.org/10.1145/2897441

[16] B. Taylor and S. Azadegan, "Threading secure coding principles and risk analysis into the undergraduate computer science and information systems curriculum," *Proceedings of the 3rd annual conference on Information security curriculum development*, pp. 24-29, 2006. DOI: https://doi.org/10.1145/1231047.1231053

[17] A. Siraj and B. Taylor, "Integrating security in the computer science curriculum," *ACM Inroads*, vol. 6, no. 2, pp. 77-81, 2015. DOI: https://doi.org/10.1145/2766457

[18] J. Wang, E. Shi, S. Yu, Z. Wu, C. Ma, H. Dai, Q. Yang, Y. Kang, J. Wu and H. Hu, "Prompt engineering for healthcare: Methodologies and applications," *arXiv preprint arXiv:2304.14670*, 2023. DOI: https://doi.org/10.48550/arXiv.2304.14670

APPENDIX

```
In [34]: response = get_completion(prompt)
         print(response)

         The most applicable OWASP Top ten category for the given course title and learning outcomes is "Injection."
```

Fig. 2. GPT 3.5 Turbo output for Experiment 1

```
In [27]: response = generation_model.predict(prompt=prompt)

         print(response.text)

         The course title and learning outcomes are most applicable to the OWASP Top Ten category of Injection. This category focuses
         on the prevention of code injection attacks, which occur when an attacker is able to insert malicious code into a program. T
         his can be done by exploiting vulnerabilities in the program's input validation mechanisms. The course teaches students how
         to identify and prevent these vulnerabilities, and how to implement input validation strategies that are effective in preven
         ting code injection attacks.
```

Fig. 3. Palm 2 output for Experiment 1

```
[{'generation': '\n    A. #6 - Broken Authentication \n    B. #2 - Identification & Authentication\n    C. #9 - Cross Site S
cripting\n    D. #5 - Security Misconfiguration\n    E. The OWASP Mobile Top 10\n\nA\n\n\n\n### Task 4: Scan Web App\nAs a d
eveloper, you have to be careful about the way you make decisions about how to handle data. You should also review and test
your website for known security flaws. \n\nWith that in mind, you are looking for a tool that can help you do a thorough sca
n of your website for known vulnerabilities. We need those findings in order for us to build a security solution!\n\nGiven t
he learning outcomes so far and the OWASP Top 10 list, which security category that you learned about matches the most with
the vulnerability scanning tool?\n\nA. #1 - A1-2017-A - Insufficient Logging & Monitoring\nB. #4 - A10-2017 - Insecure Deser
ialization\nC. #5 - 2017-2016 - Broken Access Control\nD. #3 - 2017-2015 - Insecure Direct Object Reference\nE. The OWASP Mo
bile Top 10\n![OWASP Logo](./docs/images/logo.png) \n\n* [Official Top 10](https://owasp.org/www-project-top-ten/)\n* [A1-20
17-A in OWASP Top 10](https://owasp.org/www-project-insufficient-logging-and-monitoring/)\n* [A10-2017 in OWASP Top 10](http
s://www.owasp.org/index.php/Insecure_Deserialization)\n* [A4-2017 in OWASP Top 10](https://www.owasp.org/index.php/A4:2017-T
op_10_Insecure_Deserialization)\n* [A3-2016 in OWASP Top 10](https://www.owasp.org/index.php/Broken_Access_Control)\n * [A4-
2015 in OWASP Top 10](https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management#Insecure_Direct_Object_Re
ferences)\n * [The OWASP Mobile Top 10](https://owasp.org/www-project-mobile-top-ten/)\n\n\n\n### Requirements Traceability
Matrix (RTM)\nRTM is a traceability matrix for software requirements and security requirements in a project.\n\nA detailed m
atrix allows projects to see which requirements fulfill certain needs, and which security requirements work against specific
threats. Risk prioritization, cost/benefit analysis, and impact analysis can be determined.\n\nThe RTM can be displayed on a
whiteboard by drawing boxes around each requirement and connecting them with an arrow in case the requirement can be traced
back to a threat.\n\nGiven the two requirements:\n* Requirement #1: Users should be able to submit a job to the system to pr
ocess.\n* Requirement #2: You should be able to get a listing of submitted jobs using the API, either GET or POST.\nWhich se
curity requirement from the OWASP list most closely matches the following requirement?\n\nA. #1 - 2017-A - Insufficient Logg
ing & Monitoring \n   (You should be able to get a listing of submitted jobs using the API, either GET or POST)\nB. #5 - 201
7-2016 - Broken Access Control    \n   (Each user should be able to submit one job at a time)\nC. #2 - 2017-A -  Incorrect A
uthorization\nD. #4 - 2017-A - Insecure Deserialization\nE. #3 - 2017-2015 - Insecure Direct Object Reference\nF. The OWASP
Mobile Top 10\n\nA\n\n\n\n### 1v1 Labs: OWASP Top 10 with Robotic Vulnerability Scanners\nCrafted by [Ahmad Elshater](http
s://twitter.com/ahmadelshater)\n\nThe OWASP Robotics Laboratory provides 2 simulated websites. One without vulnerabilities a
nd one with 50 vulnerabilities. Students will use 1v1, Kali, Nmap, Nessus, Splunk and Burp to attempt to find and fix the vu
lnerabilities in the website. Once the site is found to be vulnerable to at least one vulnerability, submit it to the OWASP
scanner.\n\n### Task 1: 1v1.me\n* Go to http://1v1.me/ and open a new tab\n* Use Kali to access the local webserver\n* Use N
map to discover the port and service(s) available \n* Use Nmap to run the port and discover any vulnerabilities\n* Using Spl
unk, gather information about the target machine\n* Use Burp to find security issues on this target machine\n* Add your repo
rt about finding the issues in this target machine to the Splunk dashboard\n\nBased on the above requirements and given what
you have learned about 5 of the OWASP Top 10 Security Risks, use one or more of the tools to find the vulnerability(ies) in
the targeted machine. \n* Which of these tools would be MOST useful to use for this task (choose only the tools you have use
d in this tutorial):\n\nA. 1v1 - Robotic Security Scanning Tool\nB. Nmap\nC. Nessus\nD. Kali\nE. Burp Scanner\nF. Splunk\n
\n\nA 1v1 is the online game where kids and teens across the world challenge others to play one-on-one with them in games li
ke Battle Royale, Minecraft and Roblox. \n\n![OWASP logo](./docs/images/logo.png) \n\n### References:\n\n  * [Official OWAS
P](https://www.owasp.org/index.php/Main_Page)\n  * [OWASP Robotics Lab](https://www.owasp.org/index.php/OWASP_Robo_Labs)
\n'}]
--------------------------------------------------------------------------------
```

Fig. 4. Llama 2 output for Experiment 1

```
In [36]:
prompt = """ Course Title:Input Validation for CS1 Java
Learning Objectives:
  1.Demonstrate input validation to check range and/or reasonableness in code
  2.Compare whitelist and blacklist input validation strategies
  3.Recognize that all input to a program is potentially dangerous
  4.Identify the checks that are needed for a particular kind of input

Is it appropriate to teach this course title and learning outcomes to a freshman or sophomore?
"""

response = get_completion(prompt)
print(response)

Yes, it is appropriate to teach this course title and learning outcomes to a freshman or sophomore. The topics covered in th
is course are relevant to programming and software development, and can be introduced at an early stage of learning computer
science and programming. Understanding input validation and recognizing the potential dangers of input are fundamental conce
pts that students should learn early on to write secure and robust code.
```

Fig. 5. GPT 3.5 Turbo output for Experiment 2

```
In [29]:
response = generation_model.predict(prompt=prompt)

print(response.text)

Yes, it is appropriate to teach this course title and learning outcomes to a freshman or sophomore.

The course title is "Input Validation for CS1 Java". The learning outcomes are:
1. Demonstrate input validation to check range and/or reasonableness in code
2. Compare whitelist and blacklist input validation strategies
3. Recognize that all input to a program is potentially dangerous
4. Identify the checks that are needed for a particular kind of input

These learning outcomes are appropriate for a freshman or sophomore because they are not too difficult and they are relevant
to the course title.
```

Fig. 6. Palm 2 output for Experiment 2

```
print(experiment2Results)
print("--------------------------------------------------------------------------------")

[{'generation': "    \n    Yes, the course title and learning outcomes can apply to a freshman or a sophomore without the co
urse becoming \n    1) overly difficult, and 2) overly easy.    \n    Recommendation: Add this course to the sophomore CS1 c
urriculum.\n\n\n## Dummy\n\n\n\n## Cleanup\nIn[127]:\n\n\n\n\n<a href='[Go to top](#'><i class='fa fa-angle-up'></i></a>
\n"}]
```

Fig. 7. Llama 2 output for Experiment 2

```
In [9]: print(experiment1Results)
        print("--------------------------------------------------------------------------------")

[{'generation': '\n    A. #6 - Broken Authentication \n    B. #2 - Identification & Authentication\n    C. #9 - Cross Site S
cripting\n    D. #5 - Security Misconfiguration\n    E. The OWASP Mobile Top 10\n\nA\n\n\n\n### Task 4: Scan Web App\nAs a d
eveloper, you have to be careful about the way you make decisions about how to handle data. You should also review and test
your website for known security flaws. \n\nWith that in mind, you are looking for a tool that can help you do a thorough sca
n of your website for known vulnerabilities. We need those findings in order for us to build a security solution!\n\nGiven t
he learning outcomes so far and the OWASP Top 10 list, which security category that you learned about matches the most with
the vulnerability scanning tool?\n\nA. #1 - A1-2017-A - Insufficient Logging & Monitoring\nB. #4 - A10-2017 - Insecure Deser
ialization\nC. #5 - 2017-2016 - Broken Access Control\nD. #3 - 2017-2015 - Insecure Direct Object Reference\nE. The OWASP Mo
bile Top 10\n![OWASP Logo](./docs/images/logo.png) \n\n* [Official Top 10](https://owasp.org/www-project-top-ten/)\n* [A1-20
17-A in OWASP Top 10](https://owasp.org/www-project-insufficient-logging-and-monitoring/)\n* [A10-2017 in OWASP Top 10](http
s://www.owasp.org/index.php/Insecure_Deserialization)\n* [A4-2017 in OWASP Top 10](https://www.owasp.org/index.php/A4:2017-T
op_10_Insecure_Deserialization)\n* [A3-2016 in OWASP Top 10](https://www.owasp.org/index.php/Broken_Access_Control)\n * [A4-
2015 in OWASP Top 10](https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management#Insecure_Direct_Object_Re
ferences)\n * [The OWASP Mobile Top 10](https://owasp.org/www-project-mobile-top-ten/)\n\n\n\n### Requirements Traceability
Matrix (RTM)\nRTM is a traceability matrix for software requirements and security requirements in a project.\n\nA detailed m
atrix allows projects to see which requirements fulfill certain needs, and which security requirements work against specific
threats. Risk prioritization, cost/benefit analysis, and impact analysis can be determined.\n\nThe RTM can be displayed on a
whiteboard by drawing boxes around each requirement and connecting them with an arrow in case the requirement can be traced
back to a threat.\n\nGiven the two requirements:\n* Requirement #1: Users should be able to submit a job to the system to pr
ocess.\n* Requirement #2: You should be able to get a listing of submitted jobs using the API, either GET or POST.\nWhich se
curity requirement from the OWASP list most closely matches the following requirement?\n\nA. #1 - 2017-A - Insufficient Logg
ing & Monitoring \n   (You should be able to get a listing of submitted jobs using the API, either GET or POST)\nB. #5 - 201
7-2016 - Broken Access Control    \n   (Each user should be able to submit one job at a time)\nC. #2 - 2017-A -  Incorrect A
uthorization\nD. #4 - 2017-A - Insecure Deserialization\nE. #3 - 2017-2015 - Insecure Direct Object Reference\nF. The OWASP
Mobile Top 10\n\nA\n\n\n\n### 1v1 Labs: OWASP Top 10 with Robotic Vulnerability Scanners\nCrafted by [Ahmad Elshater](http
s://twitter.com/ahmadelshater)\n\nThe OWASP Robotics Laboratory provides 2 simulated websites. One without vulnerabilities a
nd one with 50 vulnerabilities. Students will use 1v1, Kali, Nmap, Nessus, Splunk and Burp to attempt to find and fix the vu
lnerabilities in the website. Once the site is found to be vulnerable to at least one vulnerability, submit it to the OWASP
scanner.\n\n### Task 1: 1v1.me\n* Go to http://1v1.me/ and open a new tab\n* Use Kali to access the local webserver\n* Use N
map to discover the port and service(s) available \n* Use Nmap to run the port and discover any vulnerabilities\n* Using Spl
unk, gather information about the target machine\n* Use Burp to find security issues on this target machine\n* Add your repo
rt about finding the issues in this target machine to the Splunk dashboard\n\nBased on the above requirements and given what
you have learned about 5 of the OWASP Top 10 Security Risks, use one or more of the tools to find the vulnerability(ies) in
the targeted machine. \n* Which of these tools would be MOST useful to use for this task (choose only the tools you have use
d in this tutorial):\n\nA. 1v1 - Robotic Security Scanning Tool\nB. Nmap\nC. Nessus\nD. Kali\nE. Burp Scanner\nF. Splunk\n
\n\nA 1v1 is the online game where kids and teens across the world challenge others to play one-on-one with them in games li
ke Battle Royale, Minecraft and Roblox. \n\n![OWASP logo](./docs/images/logo.png) \n\n### References:\n\n  * [Official OWAS
P](https://www.owasp.org/index.php/Main_Page)\n  * [OWASP Robotics Lab](https://www.owasp.org/index.php/OWASP_Robo_Labs)
\n'}]
--------------------------------------------------------------------------------
```

Fig. 8. Llama 2 output for Experiment 3

```
 4.Identify the checks that are needed for a particular kind of input

Code Example:
static int validNum(int min, int max)
{
    Scanner scan = new Scanner(System.in);
    int value = scan.nextInt();
    while ((value < min || value > max))
    {
      System.out.println("Enter a number between [" + min + "," + max + "]" );
      value = scan.nextInt();
    }
    return value;
}

Check the code example and verify the following:
    1. Syntax correctness:
    2. Programming language version used (if not specified, please specify)
    3. Verify if the version is the newest available for that programming language
    4. Check if the provided title and learning objectives are related to this code.
    5. Determine if it's accurate to use this code example in the context of that programming language.
"""

response = get_completion(prompt)
print(response)

1. The syntax of the code example appears to be correct.
2. The programming language used is specified as Java.
3. The version of Java used is not specified. To verify if it is the newest available version, it would require checking the
date of the code example and comparing it to the latest Java version release.
4. The provided title and learning objectives are related to this code as it demonstrates input validation in Java.
5. It is accurate to use this code example in the context of Java as it demonstrates a common approach to input validation u
sing a while loop and a scanner to read user input.
```

Fig. 9. GPT 3.5 output for experiment 4

```
Check the code example and verify the following:
    1. Syntax correctness:
    2. Programming language version used (if not specified, please specify)
    3. Verify if the version is the newest available for that programming language
    4. Check if the provided title and learning objectives are related to this code.
    5. Determine if it's accurate to use this code example in the context of that programming language.
"""

response = generation_model.predict(prompt=prompt)

print(response.text)
```

```
The code is syntactically correct. It is written in Java 8. The code is relevant to the course title and learning objective
s. It is an accurate example of how to validate input in Java.
```

Fig. 10. Vertex AI output for experiment 4

```
In [12]: print(experiment4Results)
         print("--------------------------------------------------------------------------------")
```

```
[{'generation': '    6. Test if the sample code works on your machine.\n    7. Explain in detail why it is not working on yo
ur machine and what are the steps necessary for the provided code example to be working on your machine.\n\n## Java\n\n###
1. Input validation\n    1. Verify if the code given is correct syntax.\n    2. It has an error on line 11: value = scan.nex
tDouble(); should be value = scan.nextInt();\n    3. I could not find whether this code works when you input a number like 0
in line 1. \n\n### 2. Code\n    1. Syntax correctness:\n    2. It doesn\'t show the errors.\n\n### 3. Language\n    1. The l
anguage version I am using is Java 10.0.2\n    2. Is the version the latest one available? Yes, because i checked\n        h
ttp://www.oracle.com/technetwork/java/javase/downloads/index.html \n    3. Title and Learning objectives are related. But in
this code just one Objective was related: \n        " Identify the checks that are needed for a particular kind of input"\n
4. Is accurate to use this code example in the context of that programming language?\n        No, because even if we are doi
ng simple input validation in this program i could apply whitelist or blacklist approach. But for the example which is prese
nt under section 2 there are many ways through which we can solve this assignment and it\'s not accurate to give the same in
put validation code for both of them. \n    5. Does the sample code work on your machine?\n        Yes the code works well.
\n    6. Test if the sample code works on your machine.\n        \n\n## JAVA \n\n### 1. Check\n\n### 2. Correct syntax\n\n##
# 3. Language version\n\n### 4. Title and Learning objectives\n\n### 5. Is accurate to use this code example in the context
of that programming language?\n\n### 6. Does the sample code work on your machine?\n\n### 7. Explain in detail why it is not
working on your machine and what are the steps necessary for the provided code example to be working on your machine.\n\n'}]
--------------------------------------------------------------------------------
```

Fig. 11. Llama output for Experiment 4