Evaluation of AI Models to Update Cybersecurity Curriculum

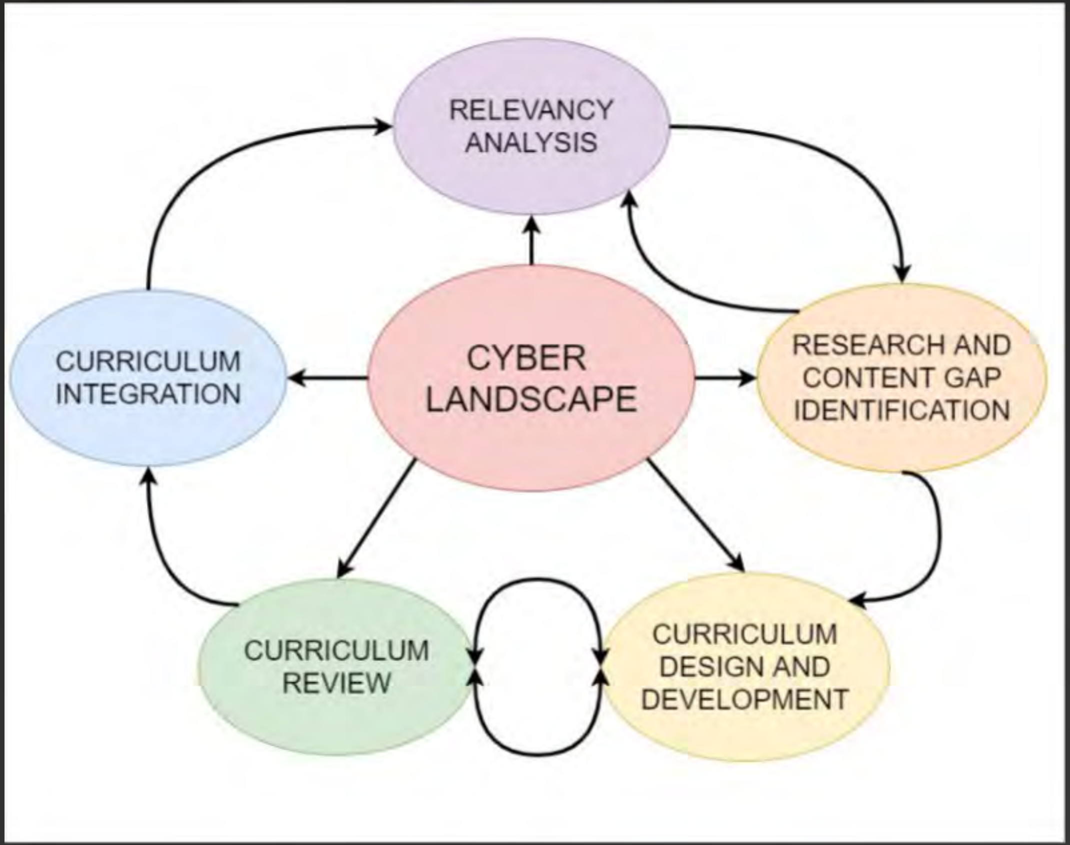
This work is partially supported by a grant through the NSA through the NCAE-C program (H98230-21-1-0175).

TOWSON UNIVERSITY

Agenda

- Introduction
- Background and Literature Review
- Methodology
- Experimental Results / Discussion
- Conclusions and Future Directions
- References

INTRODUCTION


RELEVANCY ANALYSIS
CURRICULUM INTEGRATION
CYBER LANDSCAPE
RESEARCH AND CONTENT GAP IDENTIFICATION
CURRICULUM REVIEW
CURRICULUM DESIGN AND DEVELOPMENT

This aim of this work is to examine the relevance and currency for the effectiveness of existing cybersecurity curriculum in preparation for the ever-changing nature of technology.

In the context of currency for cybersecurity curriculum, we recognize that the field of cybersecurity is multidisciplinary, influenced by industry-specific requirements that is constantly shaped by evolving threats.

Based on the dynamic nature of the field. We hope to highlight the importance of adopting innovative approaches for updating curriculum modules to reflect emerging threats and technologies.

Introduction

This study explores the potential of Large Language Models (LLMs) in assisting educators to streamline curriculum development and ensure alignment of learning outcomes, topics with industry standards and facilitate the integration of up-to-date information and examples.

In the other sections, we’ll delve into the methodology used to assess the LLMs, present the results and findings, discuss implications for educators and curriculum developers, and conclude with a summary of key insights and potential future directions.

Background / Literature Review

TOWSON UNIVERSITY

Large Language Models:

Complex Deep Learning algorithms trained with extensive datasets. These computational frameworks are notable for generating content that responds to text-based instructions.

Prompt Engineering:

Process of creating a request that produces the most effective performance on the required task. Zero-shot, Few-Shot and Zero-shot Chain of thought (Zero CoT) prompts were considered during this study, but we prioritized the use of Zero-Shot prompting.

Background / Literature Review

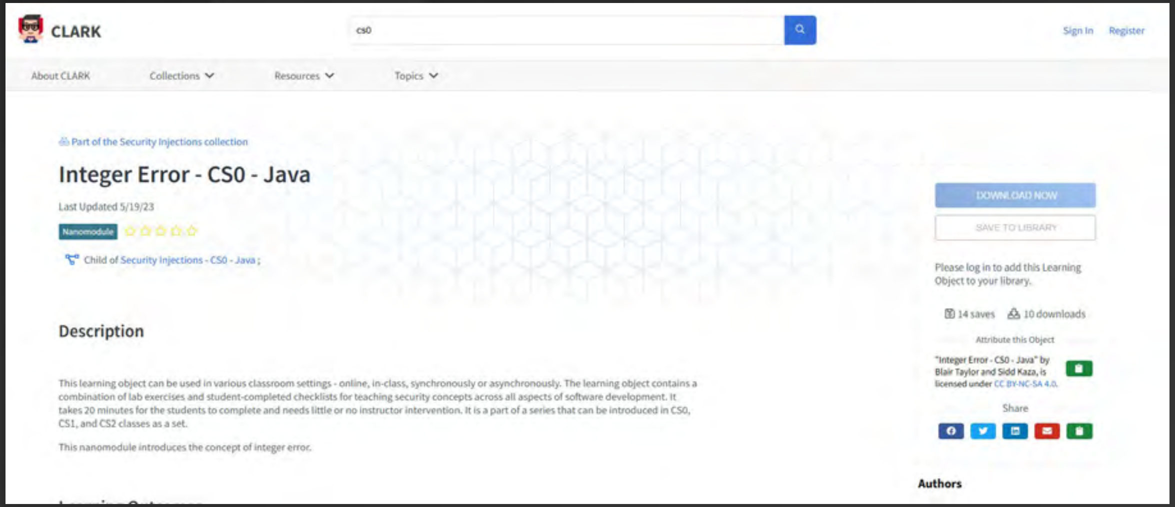
- Selection of Models:

	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. Concise, Logical, Explicit, Adaptive, and Reflective; which was used to develop prompts with brevity and clarity; well structured and coherent with clear output specifications

Experimental Design

- Dataset:
  - Learning Materials developed to introduce secure coding concepts to CS0, CS1, and CS2 students from the Security Injections @ Towson project.



Part of the Security Injections collection

Integer Error - CS0 - Java

Last Updated 5/19/23

Child of Security Injections - CS0 - Java ;

Please log in to add this Learning Object to your library.

14 saves 10 downloads

Attribute this Object

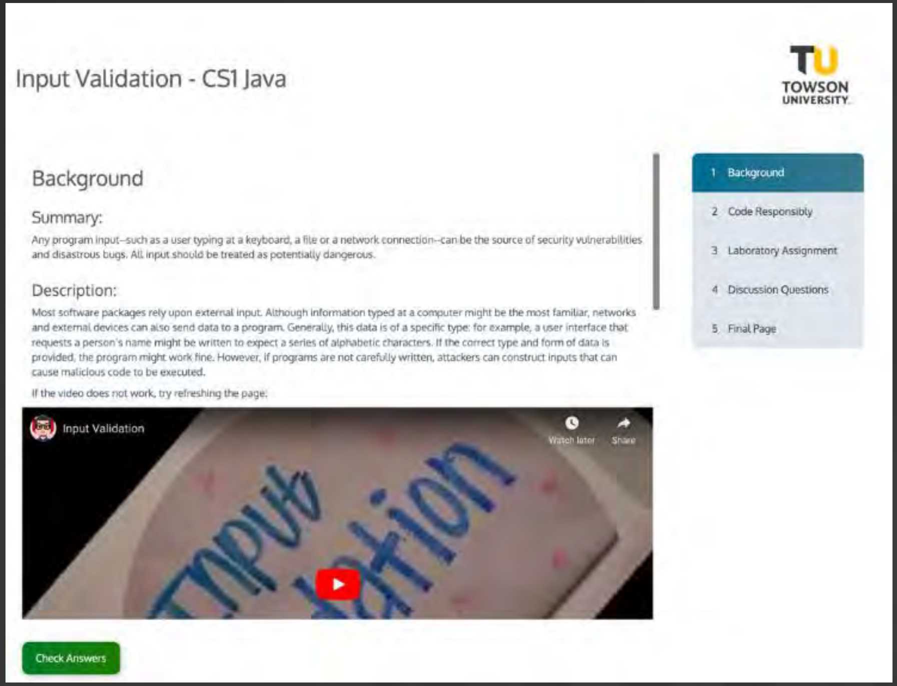
"Integer Error - CS0 - Java" by Blair Taylor and Sidd Kaza, is licensed under CC BY-NC-SA 4.0.

Description

This learning object can be used in various classroom settings - online, in-class, synchronously or asynchronously. The learning object contains a combination of lab exercises and student-completed checklists for teaching security concepts across all aspects of software development. It takes 20 minutes for the students to complete and needs little or no instructor intervention. It is a part of a series that can be introduced in CS0, CS1, and CS2 classes as a set.

This nanomodule introduces the concept of integer error.

Authors



Input Validation - CS1 Java

Background

Summary:

Any program input--such as a user typing at a keyboard, a file or a network connection--can be the source of security vulnerabilities and disastrous bugs. All input should be treated as potentially dangerous.

Description:

Most software packages rely upon external input. Although information typed at a computer might be the most familiar, networks and external devices can also send data to a program. Generally, this data is of a specific type: for example, a user interface that requests a person's name might be written to expect a series of alphabetic characters. If the correct type and form of data is provided, the program might work fine. However, if programs are not carefully written, attackers can construct inputs that can cause malicious code to be executed.

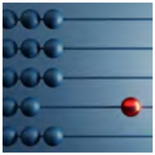
If the video does not work, try refreshing the page.

Experimental Design

FACETS

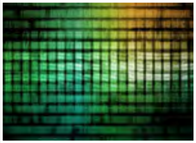
Title

Given the course title and learning outcomes, which OWASP Top ten category is most applicable.



Learning Outcomes

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



Code Examples

Syntax Correctness.

Programming language version used (if not specified, please specify).

Verify if the version is the newest available for that programming language.

Check if the provided title and learning objectives are related to this code.

Determine if it is accurate to use that code example in the context of that programming language

Real World Examples

Check how current and relevant the real-world example is.

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.

Experimental Design

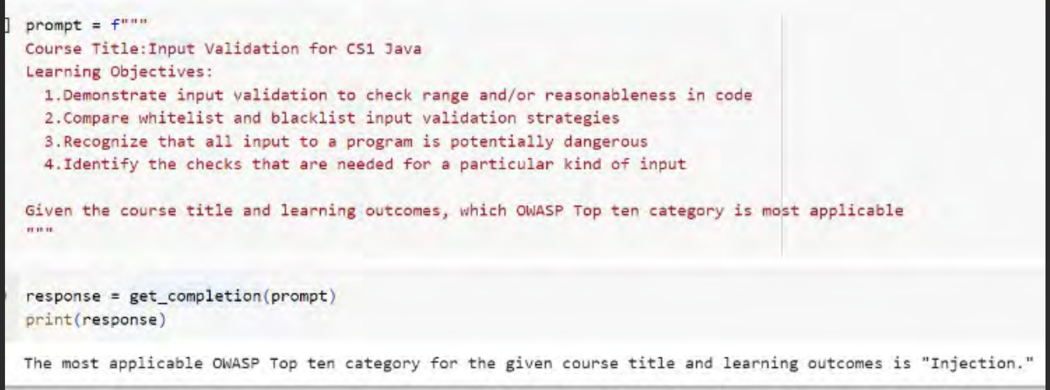
TOWSON UNIVERSITY

- 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.
- Experiments
    - *Experiment 1 – Alignment with OWASP standards*
    - *Experiment 2: Appropriateness for Level (Freshman or Sophomore)*
    - *Experiment 3: Currency for Real-World Examples*
    - *Experiment 4: Correctness for code Examples*

Results and Discussion - Experiment 1

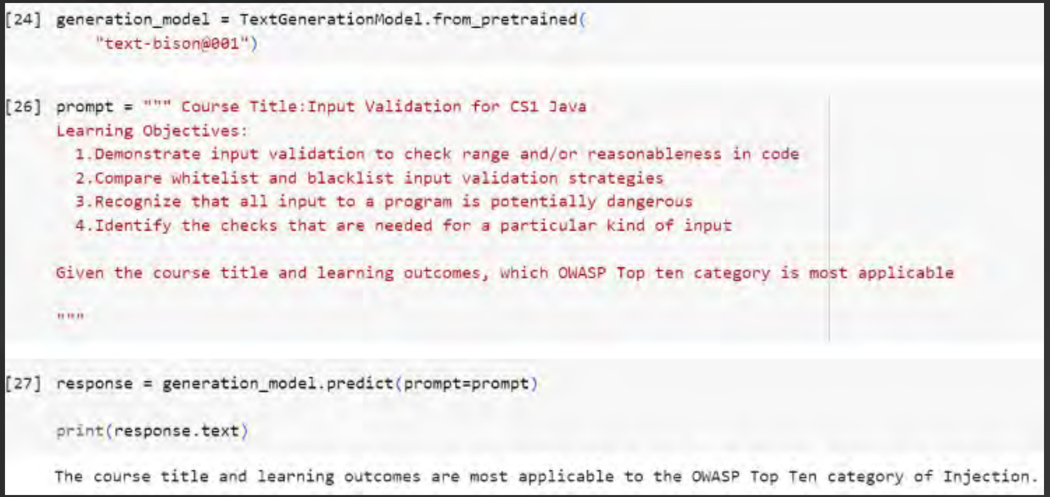
TOWSON UNIVERSITY

GPT 3.5 Turbo



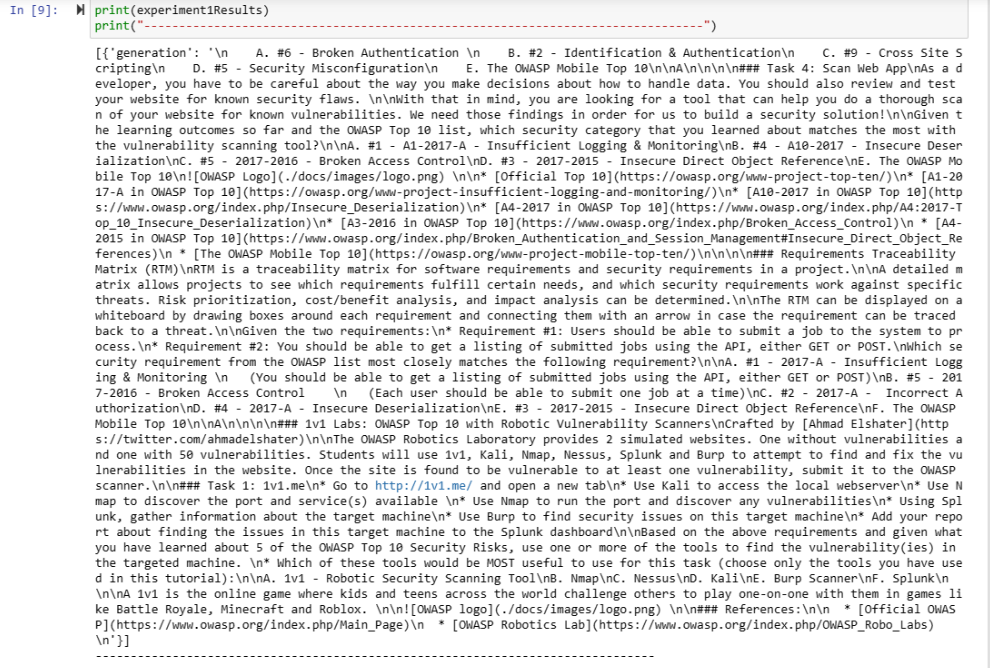
```
] prompt = f"""
  Course Title:Input Validation for CS1 Java
  Learning Objectives:
    1.Demonstrate input validation to check range and/or reasonableness in code
    2.Compare whitelist and blacklist input validation strategies
    3.Recognize that all input to a program is potentially dangerous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

  Given the course title and learning outcomes, which OWASP Top ten category is most applicable
  """

  response = get_completion(prompt)
  print(response)

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

PaLM 2



```
[24] generation_model = TextGenerationModel.from_pretrained(
         "text-bison@001")

[26] 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

     Given the course title and learning outcomes, which OWASP Top ten category is most applicable

     """

[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.
```

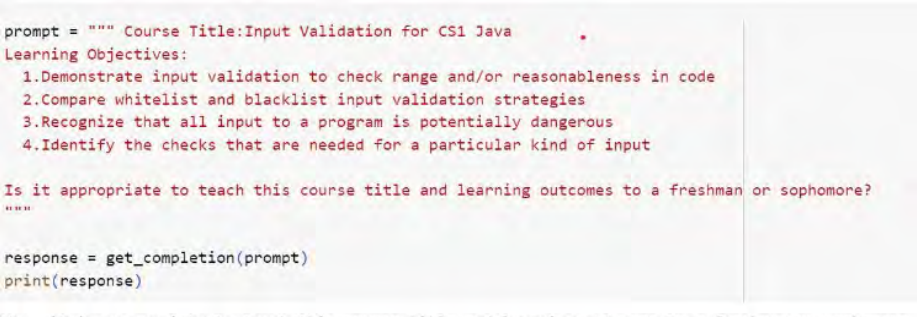
Llama2-7b



```
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'}]
         --------------------------------------------------------------------------------
```

Results and Discussion - Experiment 2

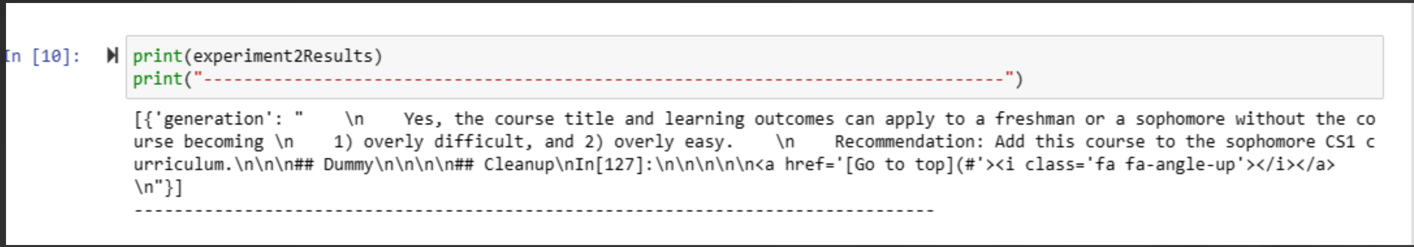
GPT 3.5 Turbo



```
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.
```

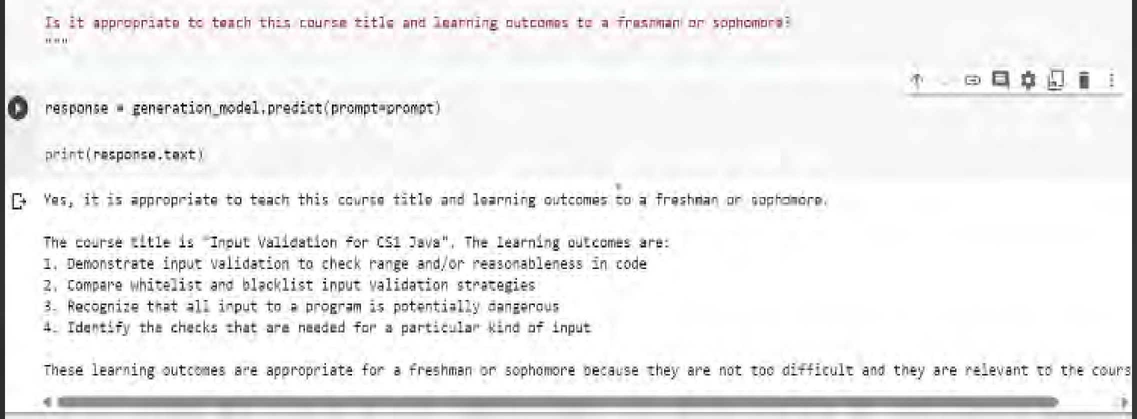
Llama2-7b



```
In [10]:  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"}]
--------------------------------------------------------------------------------
```

PaLM 2

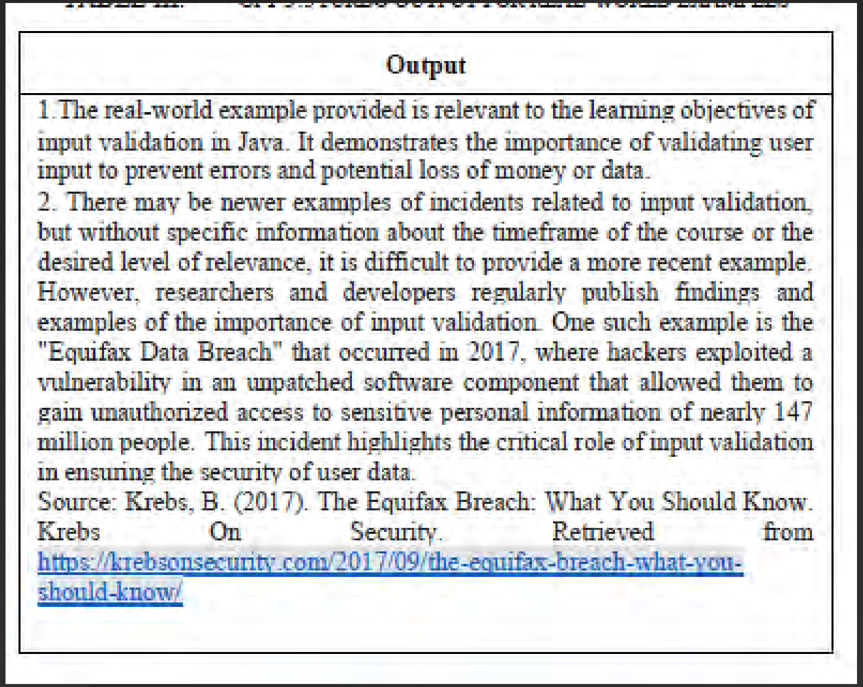


```
Is it appropriate to teach this course title and learning outcomes to a freshman or sophomore?
"""

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 cours
```

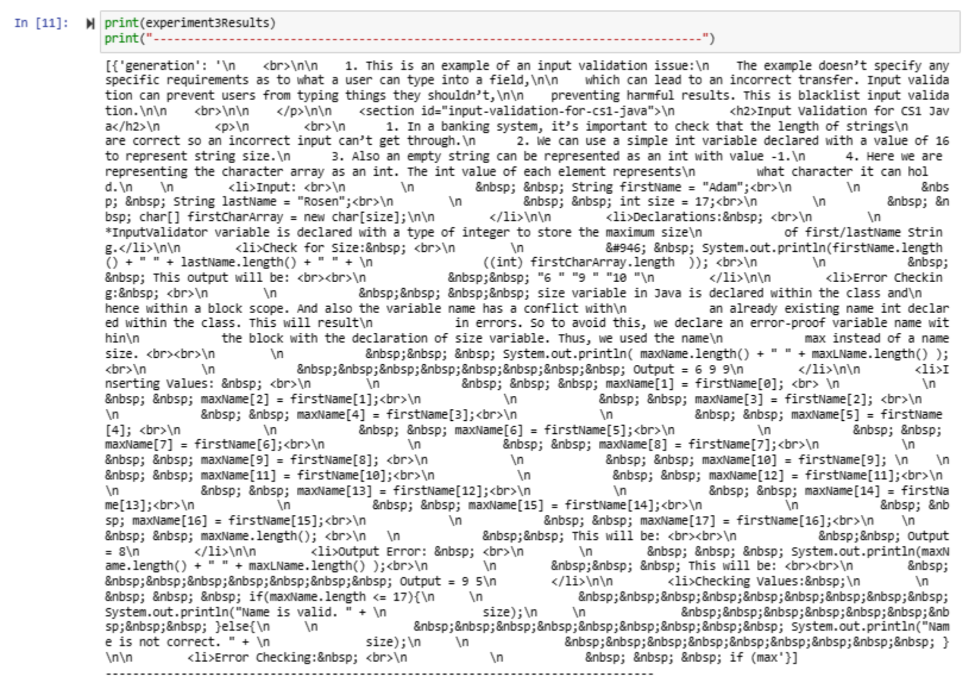
Results and Discussion - Experiment 3

GPT 3.5 Turbo



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/

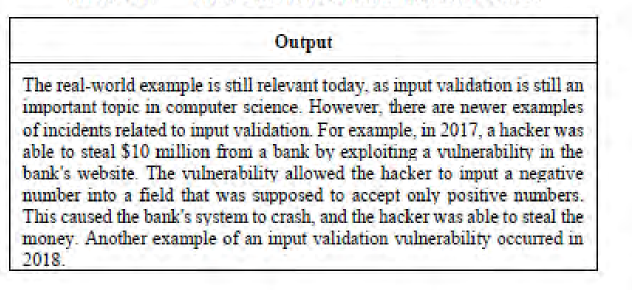
Llama2-7b



```
In [11]:  print(experiment3Results)
          print("-----------------------------------------------------------------------------")
```

```
[{'generation': '\n    <br>\n\n    1. This is an example of an input validation issue:\n    The example doesn’t specify any
specific requirements as to what a user can type into a field,\n\n    which can lead to an incorrect transfer. Input valida
tion can prevent users from typing things they shouldn’t,\n\n    preventing harmful results. This is blacklist input valida
tion.\n\n    <br>\n\n    </p>\n\n    <section id="input-validation-for-cs1-java">\n        <h2>Input Validation for CS1 Jav
a</h2>\n        <p>\n        <br>\n        1. In a banking system, it’s important to check that the length of strings\n
are correct so an incorrect input can’t get through.\n      2. We can use a simple int variable declared with a value of 16
to represent string size.\n      3. Also an empty string can be represented as an int with value -1.\n      4. Here we are
representing the character array as an int. The int value of each element represents\n        what character it can hol
d.\n    \n        <li>Input: <br>\n        \n          &nbsp; &nbsp; String firstName = "Adam";<br>\n        \n        &nbs
p; &nbsp; String lastName = "Rosen";<br>\n        \n        &nbsp; &nbsp; int size = 17;<br>\n        \n        &nbsp; &n
bsp; char[] firstCharArray = new char[size];\n\n        </li>\n\n        <li>Declarations:&nbsp; <br>\n        \n
*InputValidator variable is declared with a type of integer to store the maximum size\n          of first/lastName Strin
g.</li>\n\n        <li>Check for Size:&nbsp; <br>\n        \n          &#946; &nbsp; System.out.println(firstName.length
() + " " + lastName.length() + " " + \n                  ((int) firstCharArray.length  )); <br>\n        \n          &nbsp;
&nbsp; This output will be: <br><br>\n          &nbsp;&nbsp; "6 " "9 " "10 "\n        </li>\n\n        <li>Error Checkin
g:&nbsp; <br>\n        \n          &nbsp;&nbsp; &nbsp;&nbsp; size variable in Java is declared within the class and\n
hence within a block scope. And also the variable name has a conflict with\n          an already existing name int declar
ed within the class. This will result\n          in errors. So to avoid this, we declare an error-proof variable name wit
hin\n          the block with the declaration of size variable. Thus, we used the name\n          max instead of a name
size. <br><br>\n        \n          &nbsp;&nbsp; &nbsp; System.out.println( maxName.length() + " " + maxLName.length() );
<br>\n        \n          &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Output = 6 9 9\n        </li>\n\n        <li>I
nserting Values: &nbsp; <br>\n        \n          &nbsp; &nbsp; &nbsp; maxName[1] = firstName[0]; <br> \n          \n
&nbsp; &nbsp; maxName[2] = firstName[1];<br>\n          \n          &nbsp; &nbsp; maxName[3] = firstName[2]; <br>\n
\n          &nbsp; &nbsp; maxName[4] = firstName[3];<br>\n          \n          &nbsp; &nbsp; maxName[5] = firstName
[4]; <br>\n          \n          &nbsp; &nbsp; maxName[6] = firstName[5];<br>\n          \n          &nbsp; &nbsp;
maxName[7] = firstName[6];<br>\n          \n          &nbsp; &nbsp; maxName[8] = firstName[7];<br>\n          \n
&nbsp; &nbsp; maxName[9] = firstName[8]; <br>\n          \n          &nbsp; &nbsp; maxName[10] = firstName[9]; \n    \n
&nbsp; &nbsp; maxName[11] = firstName[10];<br>\n          \n          &nbsp; &nbsp; maxName[12] = firstName[11];<br>\n
\n          &nbsp; &nbsp; maxName[13] = firstName[12];<br>\n          \n          &nbsp; &nbsp; maxName[14] = firstNa
me[13];<br>\n          \n          &nbsp; &nbsp; maxName[15] = firstName[14];<br>\n          \n          &nbsp; &nb
sp; maxName[16] = firstName[15];<br>\n          \n          &nbsp; &nbsp; maxName[17] = firstName[16];<br>\n    \n
&nbsp; &nbsp; maxName.length(); <br>\n   \n          &nbsp;&nbsp; This will be: <br><br>\n          &nbsp;&nbsp; Output
= 8\n        </li>\n\n        <li>Output Error: &nbsp; <br>\n        \n        &nbsp; &nbsp; &nbsp; System.out.println(maxN
ame.length() + " " + maxLName.length() );<br>\n        \n        &nbsp;&nbsp; &nbsp; This will be: <br><br>\n        &nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Output = 9 5\n        </li>\n\n        <li>Checking Values:&nbsp;\n        \n
&nbsp; &nbsp; &nbsp; if(maxName.length <= 17){\n     \n          &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
System.out.println("Name is valid. " + \n          size);\n     \n          &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp; }else{\n     \n          &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; System.out.println("Nam
e is not correct. " + \n          size);\n     \n          &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; }
\n\n        <li>Error Checking:&nbsp; <br>\n          \n          &nbsp; &nbsp; &nbsp; if (max'}]
-----------------------------------------------------------------------------
```

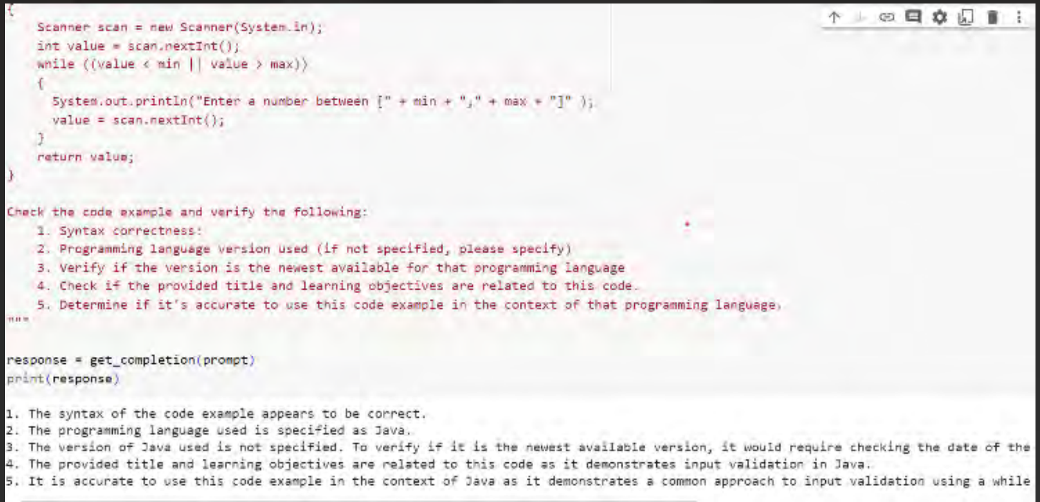
PaLM 2



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.

Results and Discussion - Experiment 4

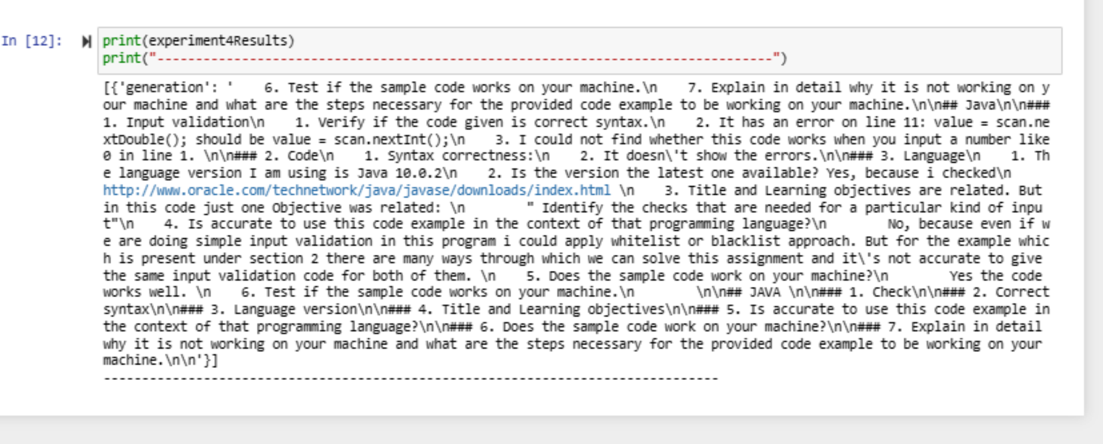
GPT 3.5 Turbo



```
{
    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
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 using a while
```

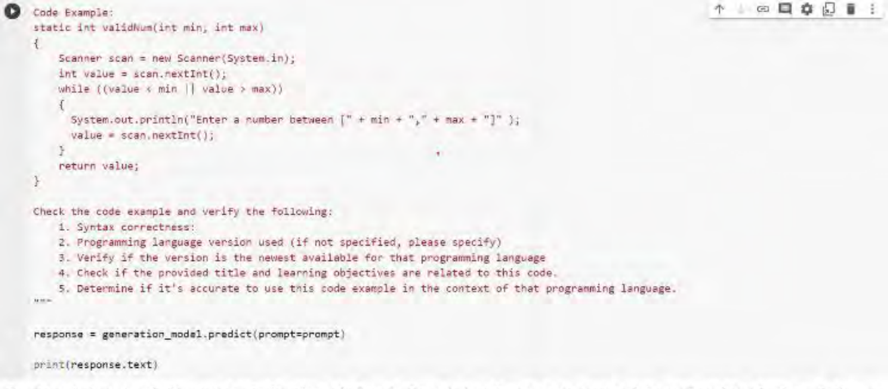
Llama2-7b



```
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 y
our 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.ne
xtDouble(); 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. Th
e language version I am using is Java 10.0.2\n    2. Is the version the latest one available? Yes, because i checked\n
http://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 inpu
t"\n    4. Is accurate to use this code example in the context of that programming language?\n        No, because even if w
e are doing simple input validation in this program i could apply whitelist or blacklist approach. But for the example whic
h is present under section 2 there are many ways through which we can solve this assignment and it\'s not accurate to give
the same input 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'}]
--------------------------------------------------------------------------------
```

PaLM 2



```
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 = 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 objectives. It is an ac
```

Future Works

Evaluating other prompting techniques for updating curriculum.	Chain of Thought(CoT), Chain of Density, Zero CoT.
Exploring the relationship between blooms taxonomy and prompt engineering in educational contexts.	Investigating the effectiveness of different prompts in stimulating specific cognitive processes and their alignment with Bloom's Taxonomy levels
Evaluating other AI Models for curriculum development	Application in Cybersecurity Education.

References

- [1] N. Chowdhury and V. Gkioulos, "Cyber security training for critical infrastructure protection: A literature review," *Computer Science Review*, vol. 40, p. 100361, 05 2021.
- [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.
- [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.
- [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.
- [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.
- [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.
- [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.
- [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.
- [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.
- [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.
- [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.
- [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.
- [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.
- [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.
- [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.
- [17] A. Siraj and B. Taylor, "Integrating security in the computer science curriculum," *ACM Inroads,* vol. 6, no. 2, pp. 77-81, 2015.
- [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.