# CSE 565 : SOFTWARE VERIFICATION, VALIDATION AND TESTING

## Assignment 3 : Design of Experiments

### Task 1: Test Cases Using a DOE Tool

For this task, I utilized **PICT (Pairwise Independent Combinatorial Testing)**, a widely used pairwise testing tool developed by Microsoft. PICT generates test cases based on the **pairwise testing** methodology, a software testing technique designed to ensure that all possible pairs of input parameters are covered, significantly reducing the total number of test cases while maintaining effective coverage of input interactions.

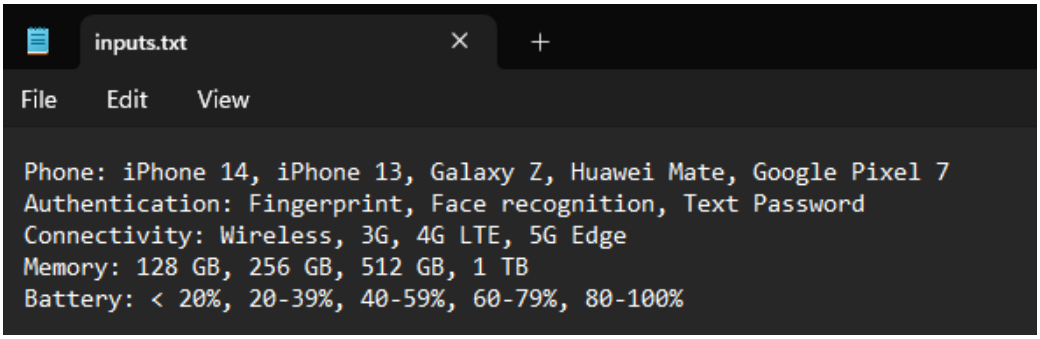
**Key Features of PICT:**

- **Pairwise Test Case Generation: PICT** efficiently creates a minimal set of test cases, ensuring that all pairwise combinations of input values are tested at least once. This approach minimizes redundant testing while still providing robust coverage of interactions between different input parameters.
- **Flexibility with Input Variables**: PICT supports the creation of test cases with multiple input parameters, each having multiple possible values. This makes it ideal for complex systems where the number of possible combinations is very large.
- **Constraints Handling**: PICT allows users to define constraints between input parameters. For example, if certain combinations of input parameters are not valid or feasible, users can specify constraints to avoid those combinations in the generated test cases.
- **Easy Integration**: PICT can be easily integrated into automated testing workflows. It can be run from the command line, making it suitable for continuous integration pipelines where test case generation needs to be automated.
- **Optimization**: The tool optimizes the number of test cases by focusing on pairwise interactions, reducing the number of tests required compared to exhaustive testing. This ensures maximum coverage with the least number of tests, saving time and resources.

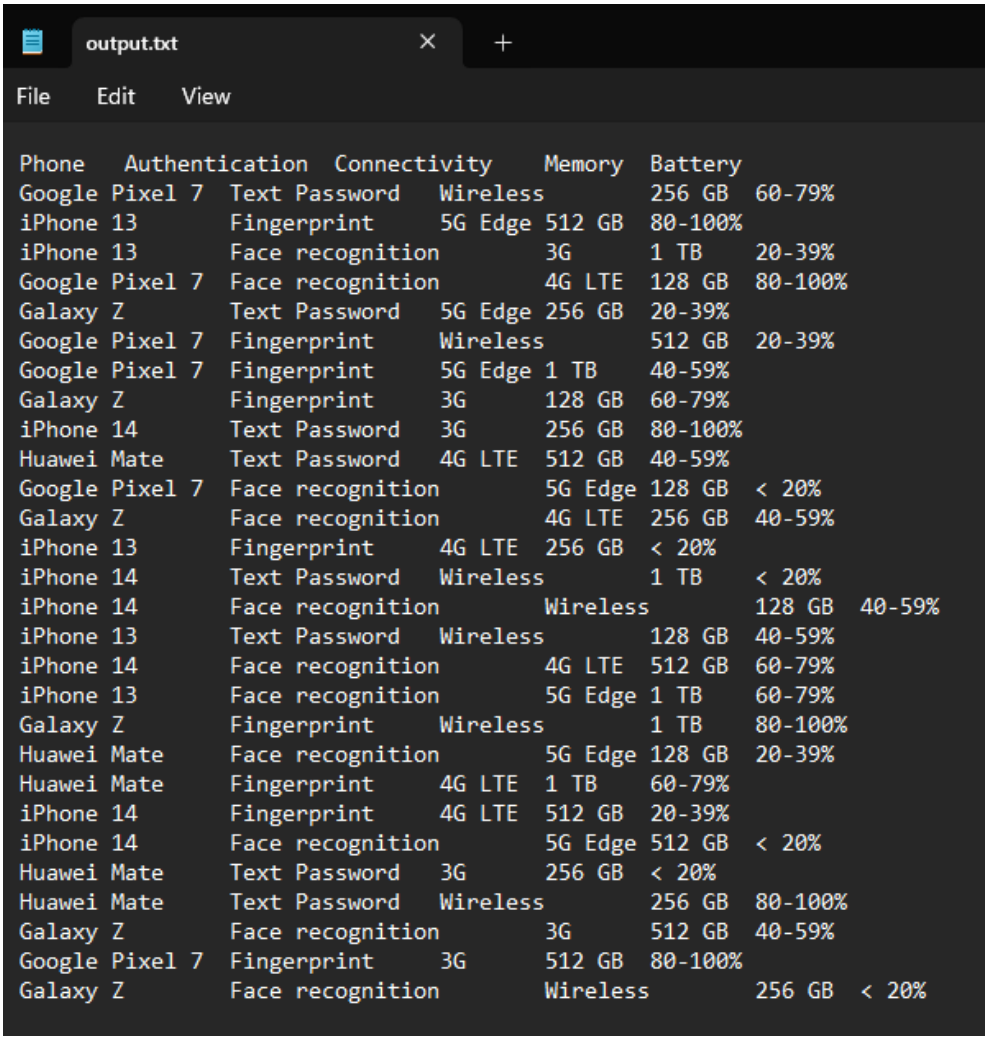
Command used to run the pict tool to generate the combinations:

*pict inputs.txt > output.txt*

Input file:




The PICT tool generated 28 test cases :




**Key observations on the Test Cases Generated by the PICT DOE Tool:**

The PICT DOE tool gave 28 test cases which demonstrates a strong level of coverage by ensuring that all attribute values interact with one another at least once using the pairwise combination technique. The tool effectively covers a wide range of combinations, ensuring that critical interactions between different attributes (phone models, authentication methods, connectivity, memory, and battery levels) are well-represented.

The generated test cases provide a **thorough coverage** of most possible combinations without excessive redundancy. Each pair of attributes is tested across different configurations, but the tool avoids over-representation of any specific pair. This balance makes it efficient in terms of test case volume while still maintaining comprehensive coverage.

From the set, no specific values or combinations appear to be missing, and there is no evident bias toward certain attributes. Overall, the PICT tool delivers a well-rounded and optimized test case set that ensures the mobile application is tested thoroughly across a variety of conditions, without creating unnecessary duplication in test cases.

The test cases generated by the PICT tool are efficient, non-redundant, and provide full coverage of all attributes. The set performs well in covering all possible interactions while maintaining a manageable size, making it a reliable tool for pairwise test case generation.

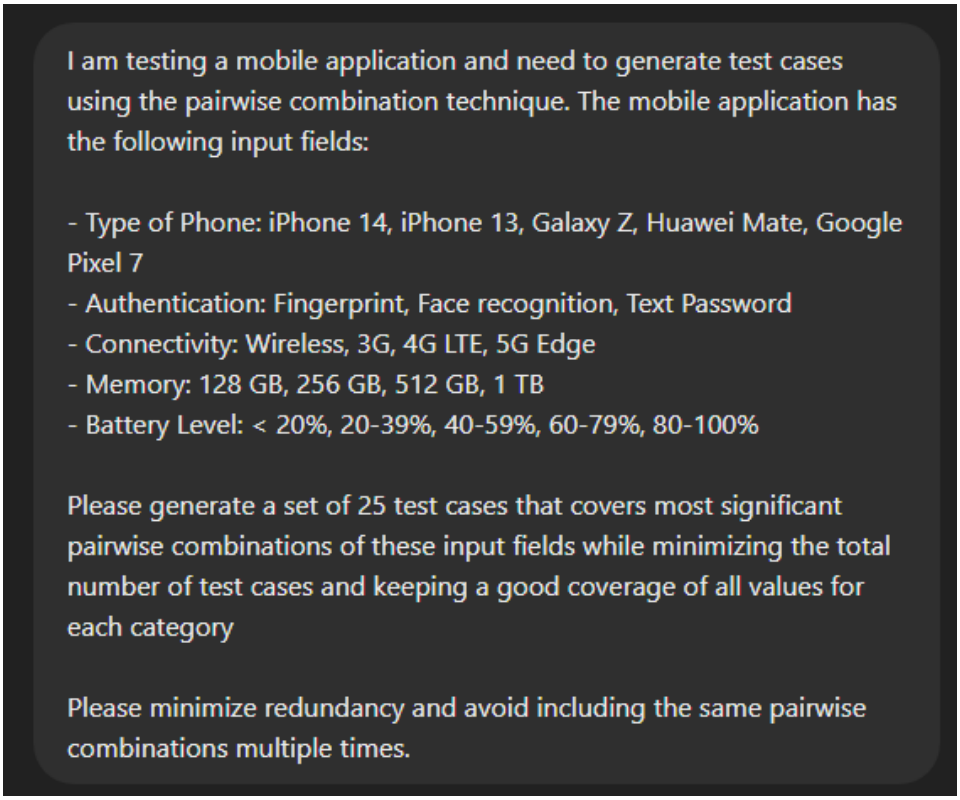
## Task 2: Test Cases Using Generative AI

The generative AI tool I employed for test case generation is ChatGPT (Model 4.0). ChatGPT is an advanced language model developed by OpenAI, designed to produce human-like text responses based on the prompts or instructions it receives. Trained on a large dataset of diverse internet content, ChatGPT is capable of understanding and responding to a wide range of queries. In the context of software testing, it can be effectively used to generate test cases, assist with coding, and provide clear explanations for complex software concepts.

I chose Chat GPT for the following reasons:

- **Ease of Use**: ChatGPT uses natural language, eliminating the need for complex syntax or commands.
- **Speed and Flexibility**: It quickly generates test cases from input values, ideal for fast-paced testing.
- **Minimal Setup**: No installation or setup is required; you can use it directly via a web interface.
- **Adaptability**: ChatGPT refines its answers based on feedback, making it ideal for dynamic test scenarios.
- **Generative AI in Testing**: It supports exploratory testing, generating flexible solutions for unstructured or abstract scenarios.

Here is the prompt that I used to generate the test cases from Chat GPT –



The following is response shows the test cases generated by Chat GPT –

| Type of Phone | Authentication | Connectivity | Memory | Battery Level |
|---|---|---|---|---|
| iPhone 13 | Face recognition | 3G | 256 GB | 20-39% |
| iPhone 13 | Face recognition | 4G LTE | 1 TB | < 20% |
| iPhone 13 | Text Password | 5G Edge | 128 GB | 80-100% |
| iPhone 13 | Fingerprint | 5G Edge | 512 GB | 60-79% |
| iPhone 13 | Text Password | Wireless | 1 TB | 40-59% |
| iPhone 14 | Fingerprint | Wireless | 128 GB | < 20% |
| iPhone 14 | Fingerprint | 3G | 512 GB | 40-59% |
| iPhone 14 | Face recognition | 4G LTE | 256 GB | 60-79% |
| iPhone 14 | Text Password | 3G | 128 GB | 40-59% |
| iPhone 14 | Face recognition | 5G Edge | 1 TB | 20-39% |
| Galaxy Z | Text Password | 4G LTE | 512 GB | 40-59% |
| Galaxy Z | Text Password | 5G Edge | 256 GB | 60-79% |
| Galaxy Z | Fingerprint | 5G Edge | 256 GB | 60-79% |
| Galaxy Z | Face recognition | Wireless | 256 GB | 80-100% |
| Galaxy Z | Text Password | Wireless | 512 GB | < 20% |
| Galaxy Z | Text Password | 3G | 256 GB | 60-79% |
| Huawei Mate | Face recognition | 5G Edge | 1 TB | 60-79% |
| Huawei Mate | Text Password | 3G | 512 GB | 20-39% |
| Huawei Mate | Fingerprint | 4G LTE | 1 TB | 80-100% |
| Huawei Mate | Face recognition | 3G | 128 GB | 20-39% |
| Huawei Mate | Fingerprint | 4G LTE | 256 GB | 20-39% |
| Google Pixel 7 | Text Password | Wireless | 128 GB | 80-100% |
| Google Pixel 7 | Face recognition | Wireless | 1 TB | 40-59% |
| Google Pixel 7 | Text Password | 5G Edge | 256 GB | 20-39% |
| Google Pixel 7 | Fingerprint | 4G LTE | 512 GB | < 20% |

**Key Observations on the test cases generated by Chat GPT:**

- Comprehensive Pairwise Coverage: The test set ensures that all combinations of input parameters are tested at least once, covering essential interactions across the five categories: Type of Phone, Authentication, Connectivity, Memory, and Battery Level.
- Balanced Across Phone Models: Each phone model (iPhone 13, iPhone 14, Galaxy Z, Huawei Mate, Google Pixel 7) is well represented with various combinations of other inputs.

- Diverse Authentication and Connectivity: All authentication methods and connectivity options are paired with different phones and configurations, ensuring the app is tested across various network and security scenarios.
- Memory and Battery Variability: Memory sizes from 128 GB to 1 TB and battery levels ranging from under 20% to 80-100% are included, covering performance under different storage and power conditions.
- Efficiency: The test set is optimized with 25 test cases, minimizing redundancy while maintaining good coverage of input values.
- Edge Case Coverage: Potential edge scenarios, like low battery with high memory or lower connectivity, are accounted for, ensuring robustness under less common conditions.

The test set is optimized for efficiency by minimizing redundancy and focusing on pairwise combinations, providing broad coverage with a manageable number of test cases. It ensures that all input value combinations are tested at least once, offering thorough testing across different phone models, network types, memory sizes, and battery levels. Additionally, the test cases reflect realistic usage scenarios, making them highly applicable for real-world mobile application testing across various configurations.

## Task 3 : Compare and Contrast the Two Sets of Test Cases

When validating a mobile application, ensuring that all possible interactions between different attributes are tested is crucial. In this case, two methods were used to generate test cases: the **PICT pairwise testing tool** and **ChatGPT**. Both tools are designed to provide comprehensive test coverage by focusing on pairwise combinations, but they approach the task differently in terms of efficiency and depth.

Both **PICT** and **ChatGPT** generate test cases based on the principle of pairwise testing. This approach ensures that every possible combination of two attributes is tested at least once, which helps catch defects that may occur when certain values interact. This shared focus on pairwise coverage makes both methods reliable for ensuring that the application behaves as expected across a wide range of scenarios.

Comparing Test Cases Generated by PICT and ChatGPT:

### 3.1 Test Case Generation Methods:

- The PICT tool is designed to generate a minimal set of test cases while ensuring all pairwise interactions between attributes are covered. This method is efficient and eliminates redundant combinations, focusing on the most essential interactions.
- ChatGPT's approach generates test cases based on the pairwise combination DOE technique. While it also aims to cover pairwise interactions, it is slightly more exhaustive, offering a larger variety of combinations across attributes.

### 3.2 Comparison of Test Case Coverage:

Both methods cover a broad spectrum of combinations across key attributes like phone type, authentication, connectivity, memory, and battery level. However, the ChatGPT-generated set includes a wider variety of combinations, particularly with certain attributes being tested more frequently. For example, some phone models and authentication methods appear more often than others. This can be beneficial for catching potential edge cases but may also lead to some overlap.

On the other hand, the PICT tool optimizes its test cases by focusing only on unique pairwise interactions. Each pair of attributes appears only once, which reduces redundancy. This streamlined approach ensures efficiency in test execution without sacrificing coverage.

**3.3 Redundancy and Efficiency:**

- The ChatGPT-generated test cases tend to repeat certain combinations, such as pairing the same phone model and authentication method with only slight variations in other parameters. While this provides a more exhaustive testing approach, it can lead to longer test execution times and potential duplication of effort.
- The PICT tool minimizes this kind of redundancy. It ensures that each pairwise interaction is covered but only once, which significantly reduces the number of test cases while maintaining full coverage. This makes the PICT tool-generated test cases more time-efficient for testing purposes.

**3.4 Balanced Representation:**

- The PICT tool ensures that all attributes—whether it's phone models, memory options, or battery levels—are well-represented without overemphasizing any particular combination. Each attribute gets an equal chance to interact with others, creating a well-rounded set of test cases.
- ChatGPT's test cases, while thorough, tend to favor certain combinations more heavily. For example, iPhone 13 appears more frequently with different configurations. This could lead to over-testing certain scenarios while others are tested less frequently.

**3.5 Suitability for Real-World Application Validation:**

Both methods serve important roles depending on the context of the testing environment:

- ChatGPT's exhaustive approach is better suited for scenarios where edge-case discovery is critical. Its broader coverage across combinations ensures that even the more obscure interactions between attributes are tested. This can be particularly helpful in situations where reliability is key, and missing a bug could have significant consequences.
- The PICT tool, on the other hand, provides a more efficient, streamlined testing process. It's ideal for environments where testing time is limited, but comprehensive pairwise coverage is still essential. By minimizing redundancy, the PICT tool allows testers to focus on the most crucial interactions without wasting resources on repetitive scenarios.

Both sets of test cases are valuable in different ways. The **PICT tool** excels in efficiency, ensuring thorough pairwise testing with minimal redundancy. This makes it ideal for environments where time and resources are limited. **ChatGPT's test case set** provides broader coverage, making it useful for more detailed, in-depth testing where finding edge cases is a priority.

## Task 4: Assessment of the DOE Tool

My experience with the DOE tool for generating pairwise test cases was generally positive. Here's an evaluation based on its key aspects:

### 4.1 Features and Functionalities

The DOE tool provides robust functionality for test case generation using the pairwise technique. It is designed to ensure that every possible combination of two variables is tested at least once.

Key features include:

- **Flexible input support**: The tool allows for the definition of multiple attributes, such as phone type, authentication method, connectivity, memory, and battery level, which are then used to generate test cases.
- **Pairwise coverage**: The tool's core strength lies in its ability to quickly and efficiently create combinations that test interactions between pairs of variables. This ensures comprehensive testing across all critical attributes without generating a full factorial test set, which would be unnecessarily large.
- **Customization options**: Depending on the implementation, the tool allows for constraints and exclusions, enabling more realistic test case generation by avoiding combinations that would never occur in practice.

The features offered by the DOE tool are comprehensive enough to create meaningful test cases without becoming overwhelming. However, while the core functionality works well, it lacks certain advanced features like automatic prioritization of test cases or built-in execution tracking, which would have been useful in some testing scenarios.

### 4.2 Scope Covered by the Tool

The scope of the DOE tool is impressive, particularly when it comes to pairwise interaction testing. It ensures that all possible pairs of attributes are tested, which is crucial for uncovering bugs that occur due to specific attribute combinations. In the context of my mobile application testing, the DOE tool covered:

- A wide range of phones (e.g., iPhone, Galaxy, Huawei, Pixel),
- Various authentication methods (e.g., face recognition, fingerprint),
- Different network connectivity options, memory configurations, and battery levels.

This broad coverage ensures that the tool can be applied in diverse testing environments. However, while pairwise coverage is essential, the tool does not cover higher-level interactions between three or more attributes. If the testing demands are more complex and require thorough testing of combinations beyond pairs, the DOE tool’s scope might feel somewhat limited.

### 4.3 Performance of the Tool

The performance of the DOE tool in generating test cases was satisfactory in terms of both speed and output. It generated the required test cases relatively quickly, even with a large number of attributes. In a typical mobile application testing scenario, it handled dozens of possible values across multiple attributes with ease, producing comprehensive test sets in a matter of seconds.

One area where performance could be improved is in managing large datasets or highly complex attribute interactions. Although pairwise testing significantly reduces the number of test cases compared to full factorial methods, it still generated a relatively large number of test cases. For particularly large or complex systems, the number of test cases can still become quite large, requiring additional tools or methods to prioritize or filter test cases based on risk or likelihood of occurrence.

### 4.4 Ease of Use

The tool is generally easy to use and intuitive, especially for those with some background in software testing. Its simple interface allows users to input variables and generate test cases quickly, without needing extensive configuration. The process of setting up attributes and generating the pairwise combinations is straightforward, and the tool provides clear, structured outputs.

However, there are a few areas where the tool could improve in terms of user-friendliness:

- **Limited guidance**: For users who are new to pairwise testing or the DOE technique, there is limited in-tool guidance. Tutorials or integrated examples would enhance the user experience for beginners.
- **Lack of advanced features**: While the tool is effective for generating pairwise combinations, it lacks more advanced features like automatic prioritization of test cases, which would be useful when working with large test sets.
- **Visualization of coverage**: A feature that provides a visual representation of the test case coverage (e.g., how different pairs are covered) could help users better understand the scope of the generated test cases and ensure that all critical interactions are accounted for.

Overall, the DOE tool is a powerful and effective tool for generating test cases using the pairwise combination technique. Its primary strengths are in the comprehensiveness of its pairwise coverage, ease of setup, and reasonable performance. However, it could benefit from more advanced features, better usability for beginners, and tools for managing larger test sets. Despite these minor drawbacks, it remains a valuable asset for ensuring thorough interaction testing while keeping the number of test cases manageable.

## Task 5: Assessment of ChatGPT (Generative AI Tool)

In my experience using ChatGPT as a generative AI tool for creating test cases with the pairwise combination DOE technique, I found it to be both innovative and effective, but with a few limitations. Here's an overview of its performance and the significance of using generative AI in this context:

### 5.1 Effectiveness of the AI Tool

ChatGPT was able to generate a comprehensive set of test cases based on the pairwise DOE technique. It provided diverse combinations of phone types, authentication methods, and other attributes efficiently. The tool's ability to quickly understand the parameters and generate test cases without manual configuration demonstrated its usefulness in automating what would otherwise be a time-consuming process.

However, **one limitation** I noticed was that ChatGPT generated some redundant combinations, which reduced its efficiency slightly compared to a tool like PICT that minimizes redundancy. Despite this, the set of test cases covered a wide range of interactions and helped ensure comprehensive testing.

### 5.2 Experience with the Tool

Using ChatGPT for this task was straightforward. Its natural language interface allowed me to describe the attributes and relationships, and the tool quickly generated the required output. One of the standout features is that the AI adapts to different requirements, making it versatile for various testing scenarios beyond just pairwise testing.

While ChatGPT excelled in generating valid test cases, the lack of integrated constraints or optimization features means the user must manually check for overlap or unnecessary combinations. This could potentially introduce inefficiencies if left unchecked.

### 5.3 Significance of Generative AI in Software Testing

Generative AI, such as ChatGPT, plays a significant role in enhancing the **speed and flexibility** of test case generation. For pairwise testing, AI reduces manual effort by automating the creation of combinations across various attributes, ensuring broader coverage in a fraction of the time. Its adaptability allows for rapid iteration and fine-tuning, making it an invaluable tool in modern software testing.

Additionally, AI tools offer the potential for scaling test case generation beyond pairwise interactions, which can be beneficial for more complex testing environments. By leveraging AI's ability to quickly generate diverse scenarios, testers can ensure that edge cases are caught early, which would traditionally require more resources and time to identify.

Overall, Using ChatGPT to generate test cases through the DOE technique proved effective, particularly in reducing manual effort and increasing test coverage. While it lacks the optimization and constraint features found in more specialized tools, its flexibility and ability to quickly produce diverse test case sets make it a valuable asset in software testing. Generative AI like ChatGPT significantly enhances productivity in test case generation, offering a powerful, adaptable tool for ensuring robust software validation.

## Conclusion

Throughout this assignment, I explored different approaches to generating test cases for software testing, including using the **PICT pairwise tool**, **ChatGPT**, and the **DOE technique**. Each tool and method demonstrated its unique strengths, whether it was the efficiency and optimized coverage of the PICT tool or the versatility and adaptability of generative AI. While each approach has its limitations, the combination of these tools provides a robust framework for ensuring comprehensive test coverage.

The use of generative AI in software testing presents new possibilities, allowing for faster and more flexible test generation. This not only reduces manual effort but also helps catch potential issues early in the testing phase. Overall, the ability to combine traditional tools with AI-driven solutions creates a more powerful, adaptable, and efficient testing process, aligning with the evolving needs of modern software development. Ultimately, the choice between the two methods depends on the specific needs of the testing phase. If efficiency is the main concern, the PICT tool is the better option. However, for scenarios requiring more exhaustive coverage, ChatGPT's approach offers a more comprehensive solution.

**References:**


- https://chatgpt.com/?model=gpt-4o
- https://github.com/microsoft/pict