With the rising popularity of Generative AI models such as ChatGPT, and as mentioned in Section <x> there is potential to harness generative Artificial Intelligence for the task of generating natural language descriptions of structured data formats such as JSON. We seeked to explore this possibility using ChatGPT <reference for ChatGPT>. Using the files used in the user study, we asked ChatGPT to generate descriptions for each file using the following prompt:

“Describe this JSON file: <JSON file>”.

Each prompt was sent in a new “chat” window.

**File A description generated from ChatGPT:**

This is a JSON (JavaScript Object Notation) file, which is a lightweight data interchange format. The file contains an object with two key-value pairs.

The first key is "store", which has a nested structure. Within "store", there are two key-value pairs. The first is "book", which is an array of four objects representing books. Each book object has properties such as "category", "author", "title", "isbn" (for some books), and "price".

The second key within "store" is "bicycle", which is another object with properties "color" and "price".

The second key-value pair in the JSON object is "expensive", which has a value of 10.

Overall, this JSON file represents a simple inventory of a store that sells books and bicycles, with details on each item's category, author, title, ISBN (where applicable), color, and price.

**File B description generated from ChatGPT:**

The given code is a JSON file, which stands for JavaScript Object Notation. It is a lightweight format for storing and exchanging data between different systems. The JSON file contains a set of key-value pairs, where each key is a string and each value can be a string, number, boolean, array, or another JSON object.

In this particular example, the JSON object has several key-value pairs:

"name": "Jane Doe" - This key-value pair stores the name of the person as a string value.

"email": "jane.doe@example.com" - This key-value pair stores the email address of the person as a string value.

"address": {...} - This key-value pair stores an object that represents the address of the person. The address object itself contains several key-value pairs that store the street, city, state, and zip code as string values.

"phoneNumbers": [...] - This key-value pair stores an array of objects that represent the person's phone numbers. Each object contains two key-value pairs, one for the type of number (home or work) and one for the phone number itself.

"age": 35 - This key-value pair stores the age of the person as a number value.

"isMarried": true - This key-value pair stores a boolean value that indicates whether the person is married or not.

"hobbies": [...] - This key-value pair stores an array of strings that represent the person's hobbies.

**File C description generated from ChatGPT:**

This is a JSON (JavaScript Object Notation) file that contains data in a structured format. It consists of three main properties: "title", "description", and "items".

The "title" property is a string that provides a title for the JSON file, and the "description" property is another string that provides a brief description of the file.

The "items" property is an array of objects, where each object represents a different item. Each item has three properties: "name", "quantity", and "price". The "name" property is a string that identifies the name of the item, the "quantity" property is an integer that represents the quantity of that item, and the "price" property is a number that represents the price of that item.

In this example, there are three items listed: "item1", "item2", and "item3", each with their respective quantity and price values. This JSON file might be used to store and transmit data about items in a shopping cart, for example.

Looking at the descriptions produced for each file, we make some observations.

**Difference in descriptions**

Firstly, despite executing the same prompt in the same conditions, the descriptions are all quite different. Description A is very detailed and specific, focusing on the structure and data contained within the JSON file. It provides a comprehensive overview of each book object, describing the properties and values within each one. Description B, on the other hand, takes a more general approach to describing JSON files. It provides an explanation of what JSON is and what it is used for, as well as a broad overview of the key-value pairs contained within the file. Description C strikes a balance between the two, providing both a general overview of the file's structure and a more detailed look at its contents. It introduces the main properties of the file and explains their purpose, before diving into the specific details of the "items" property.

The fact that three distinct descriptions were generated by ChatGPT for three different JSON files using the same prompt demonstrates the versatility and adaptability of generative AI like ChatGPT. However, we need to consider whether different styles of description is positive or negative.

Consistency in descriptions of JSON files can be beneficial for screen-reader users who rely on such descriptions to understand the structure and content of the file. Having a consistent format and style of description across different JSON files can help users with visual impairments to quickly and easily understand the information contained in the file, especially if they are already familiar with the structure and properties of JSON objects.

On the other hand, more context aware descriptions based on the content of JSON files, may be beneficial in providing users more relevant details. In particular, providing less information for a simple JSON file, vs providing more detailed explanation of a more complicated JSON file.

This raises an interesting research question: “Do users prefer consistent descriptions or summarizations when using code or structured text summary tools?” Understanding user preferences can help improve the design of code or structured text summary tools, making them more effective and user-friendly.

**Inference made about the what the JSON file represents**

At the end of all 3 dsecriptions, an inference about what the JSON file represents is made. The JSONtalk tool is unable to make this inference. Further research could be done to determine if this inference is meaningful for users of a JSON summarising tool, such as interviews or surveys. There may be certain use cases where this inference could be meaningful, and others where it is not required. This inference made illustrates one of the main differences between using generative AI for producing a description and using a predefined parse and description generation method. JSON files are often not used on their own, and a programmer will rarely just look at a JSON file in isolation. Generally, JSON files form part of a wider picture, for example, a NoSQL database or a RESTful API. The context of the JSON file will already give programmers some sort of inference of what it is about.

However, if generative AI is used to generate descriptions of JSON files to assist screenreader users, it is important to ensure that the descriptions are accurate and consistent in their style and tone. Inaccurate or inconsistent descriptions could potentially confuse or mislead screenreader users, leading to errors or misinterpretations. It is worth carefully considering and testing the optimum prompts given to users. There may be

Additionally, it is important to consider the potential biases that may be present in the generative AI model and its training data, as these could inadvertently influence the descriptions generated for the JSON files. It is therefore crucial to carefully evaluate the outputs of the generative AI model and ensure that they are appropriate for the intended audience, especially in the context of accessibility for individuals with disabilities.

The decision to use generative AI or predefined parsing visitor methods, such as using ANTLR4, to generate natural language descriptions of JSON files depends on the specific needs and goals of the project. Each method has its own strengths and weaknesses that should be carefully considered.

Generative AI can produce descriptions without predefined rules, allowing for more flexible and adaptable output. This can be especially useful in cases where the structure and content of the JSON file may be highly variable, making it difficult to create predefined parsing rules. However, generative AI can also produce less accurate and less meaningful output if not trained properly, and it may require significant computational resources to develop and train the model.

Predefined parsing visitor methods, such as using ANTLR4, require more upfront effort to define parsing rules and structures, but they can produce more accurate and meaningful descriptions. This method can be especially useful in cases where the structure and content of the JSON file is more predictable and stable, and where more control over the output is needed. However, it may be less flexible and adaptable compared to generative AI, and changes to the structure or content of the JSON file may require updating the parsing rules.