April 2023

GENERIC SYSTEM AND METHOD FOR JSON DATA PARSING

VISHESH KAIN
Visa

TITLE: “GENERIC SYSTEM AND METHOD FOR JSON DATA PARSING”

APPLICANT: VISA INTERNATIONAL SERVICE ASSOCIATION

ADDRESS: P.O. Box 8999, San Francisco, CA, 94128, USA

Nationality: USA

INVENTOR: VISHESH KAIN

TECHNICAL FIELD

[001] The present subject matter relates to a field of data parsing, more particularly, but not exclusively to a generic system and a method for JavaScript Object Notation (JSON) data parsing.

BACKGROUND

[002] Conventional parsing tools lack framework or open-source libraries for parsing JSON data or JSON files to flat file/data frame. JSON is a data exchange format that is lightweight, text-based, language-independent and easy for humans and machines to read and write. Certain JSON data are very complex with multiple nested hierarchy with some exceptions. Also, not every JSON data has the same schema structure. Further, some JSON files may contain multiple JSON records with different schemas. If a specific JSON schema structure is tried for parsing, then the parsing can take long time because of need to write specific logic. Further, in case the JSON data/schema is changed from upstream, then changing/updating the code manually can make process very time consuming.

[003] There may be some conventional parsing tools or logic available to parse the JSON data for each specific schema or the JSON data with limited nested hierarchy. However, these tools or logic fail to provide a generic framework which can parse any JSON data into a flat file. Sometimes the conventional parsing tools also require a schema file to parse the JSON data to represent what the JSON data may look like logically. This approach may prove to be complex and time-consuming. In addition, the conventional parsing tools may not be configured for both parsing and generation of files. Further, in case the data changes, the same schema cannot be used to parse the data. Thus, there is a need for a generic JSON data parser.

[004] The information disclosed in this background of the disclosure section is only for enhancement of understanding of the general background of the invention and should not be taken as an acknowledgement or any form of suggestion that this information forms the prior art already known to a person skilled in the art.

BRIEF DESCRIPTION OF THE DRAWINGS

[005] The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and, together with the description, serve to explain the disclosed principles. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The same numbers are used throughout the figures to reference like features and components. Some embodiments of device or system and/or methods in accordance with embodiments of the present subject matter are now described, by way of example only, and with reference to the accompanying figures, in which:

[006] Figures 1a and 1b illustrate a method for automatic JSON data parsing, in accordance with some embodiments of the present disclosure;

[007] Figures 2a and 2b illustrate an example JSON data for parsing automatically into flat file, in accordance with some embodiments of the present disclosure;

[008] Figure 3 illustrates a system for automatic JSON data parsing, in accordance with some embodiments of the present disclosure; and

[009] Figure 4 illustrates a block diagram of an exemplary computer system for implementing embodiments consistent with the present disclosure.

[010] The figures depict embodiments of the disclosure for purposes of illustration only. One skilled in the art will readily recognize from the following description that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the disclosure described herein.

DESCRIPTION OF THE DISCLOSURE

[011] In the present document, the word "exemplary" is used herein to mean "serving as an example, instance, or illustration." Any embodiment or implementation of the present subject matter described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other embodiments.

[012] While the disclosure is susceptible to various modifications and alternative forms, specific embodiment thereof has been shown by way of example in the drawings and will be described in detail below. It should be understood, however that it is not intended to limit the disclosure to the particular forms disclosed, but on the contrary, the disclosure is to cover all modifications, equivalents, and alternative falling within the spirit and the scope of the disclosure.

[013] The terms “comprises”, “comprising”, or any other variations thereof, are intended to cover a non-exclusive inclusion, such that a setup, device, or method that comprises a list of components or steps does not include only those components or steps but may include other components or steps not expressly listed or inherent to such setup or device or method. In other words, one or more elements in a device or system or apparatus proceeded by “comprises… a” does not, without more constraints, preclude the existence of other elements or additional elements in the device, system, or apparatus.

[014] The terms "an embodiment", "embodiment", "embodiments", "the embodiment", "the embodiments", "one or more embodiments", "some embodiments", and "one embodiment" mean "one or more (but not all) embodiments of the invention(s)" unless expressly specified otherwise.

[015] The terms "including", "comprising", “having” and variations thereof mean "including but not limited to", unless expressly specified otherwise.

[016] The present disclosure discloses a method (also, referred as utility) and a system that is generic for JavaScript Object Notation (JSON) data parsing. The method of the present disclosure (hereinafter referred as the method) provides an automatic way to parse the JSON data with different structures having “N” number of hierarchy and flatten them so that it can be stored in table or a file and further used for data analysis and development. The method can parse any JSON data without any schema configuration or manual intervention into flat file or data frame (tables). During parsing of the JSON data, the method automatically identifies the field/column names from the JSON data and stores parsed values in their columns respectively. The method allows the flattened JSON data to be segregated into different facts and dimension tables as per project requirements. Further, the method has the capability to detect any schema changes if the JSON structure changes from upstream in existing data pipeline and filters out

the JSON record. Also, the method allows analysis of specific JSON record with new changes for validity and actions based on analysis. The method also detects corrupted JSON records and filters them out without parsing into separate folders for analysis.

[017] Figures 1a and 1b illustrate a method for automatic JSON data parsing, in accordance with some embodiments of the present disclosure. Initially, the method receives various complex JSON data structures as input for parsing as shown in Figure 1a. Parsing is process of extracting necessary information from the JSON data, most often textual data. With reference to Figure 1b, the method filters/validates JSON data records to remove corrupt JSON records. Similarly, the method analyses JSON data for identifying JSON records with changed schema. Further, the method can analyse specific JSON record/data to identify if new changes are valid or not and can take respective action based on analysis. In an embodiment, if the JSON data is not valid or corrupt then the method filters out that JSON data and stores at a different location. The method parses the JSON data from top to bottom after analysing each key and stores the values of each key. During the parsing, a schema of the JSON data is created by the method which may be utilized for further processing. The method derives unique names of column after analysing the schema and stores the value corresponding to its column name. Upon deriving the column name and values, the method stores them in a data frame, or a file.

[018] Figures 2a and 2b illustrates an example JSON data for parsing automatically into flat files, in accordance with some embodiments of the present disclosure. Consider Figure 2a showing an example of the JSON data/records that is received as input for parsing by the method. The JSON data/record is parsed from top to bottom node by the method. At each node/key, the method identifies the type of key whether it's a dictionary, a list or a string. If the key is a string, then the method captures the value of the key. If it is a list or a dictionary, then the method recursively parses and check the child nodes/keys. In an embodiment, if the key is a list, then the method tries to create a new row and append all the parent keys to it so that the JSON data can be flattened. In Figure 2a, "name" and "employee_id" are strings, and "project" is the key, and it may be a list, dictionary or a string. In Figure 2a, the "project" is represented as list in first part inside the square brackets, and the "project" is represented as dictionary in second part inside the curly brackets. Further, after capturing every data and its corresponding keys, the method creates a unique column name and map the values with the corresponding column names. For example, in Figure 2a, the data may be "Mark", "1",

“DTMS”, “Mathew”, “2022-10-01”, “50%” and the like. While the corresponding keys for these data maybe “name”, “employee_id”, “name”, “supervisor_name”, “start_date”, “allocation” and the like. Further, the method captures these data and the corresponding keys and creates unique column name and map values as shown in Figure 2b. In an embodiment, the method can handle the keys with same name in the JSON data and create unique columns names by appending parent key to column names so that they can be generated uniquely every time. Once, the values are mapped, the method creates a data frame which can be used to store the data in a specific table in a hive or a file as shown in the Figure 2b. Further, the method can segregate the flatten JSON data into tables with different dimensions as per requirement of a project. Additionally, the method can normalize the JSON data using various normalization techniques.

[019] Figure 3 illustrates a system for automatic JSON data parsing, in accordance with some embodiments of the present disclosure. The environment 300 includes a system 301. The system 301 may be configured to implement the method for automatic JSON data parsing as explained with respect to Figures 1a and 1b. The system 301 may include one or more processor 302, Input/Output (I/O) interface 303 and a memory 304. In some embodiments, the memory 304 may be communicatively coupled to the one or more processors 302. The memory 304 stores instructions, executable by the one or more processors 302, which, on execution, may cause the system 301 to automatically parse the JSON data, as disclosed in the present disclosure. In an embodiment, the memory 304 may include one or more modules 305 and data 306. The one or more modules 305 may be configured to perform the steps of the present disclosure using the data 306, to provide automatic parsing of the JSON data. In an embodiment, each of the one or more modules 305 may be a hardware unit which may be present outside the memory 304 and coupled with the system 301. The system 301 may be implemented in a variety of computing systems, such as, a laptop computer, a desktop computer, a Personal Computer (PC), a notebook, a smartphone, a tablet, e-book readers, a server, a network server, a cloud-based server, and the like. In an embodiment, the system 301 may be a dedicated server or may be a cloud-based server.

Computing System

[020] Figure 4 illustrates a block diagram of an exemplary computer system 400 for implementing embodiments consistent with the present disclosure. In an embodiment, the computer system 400 is used to implement the system 301 for automatic parsing of the JSON data. The computer system 400 may include a central processing unit ("CPU" or "processor") 402. The processor 402 may include one or more data processors and/or specialized processing units such as, integrated system (bus) controllers, memory management control units, floating point units, graphics processing units, digital signal processing units, etc.

[021] The processor 402 may be disposed in communication with one or more input/output (I/O) devices 409 and 410 via I/O interface 401. The I/O interface 401 may employ communication protocols/methods such as, without limitation, audio, analog, digital, monaural, RCA, stereo, IEEE-1394, serial bus, universal serial bus (USB), infrared, PS/2, BNC, coaxial, component, composite, digital visual interface (DVI), high-definition multimedia interface (HDMI), radio frequency (RF) antennas, S-Video, VGA, IEEE 802.n /b/g/n/x, Bluetooth, cellular (e.g., code-division multiple access (CDMA), high-speed packet access (HSPA+), global system for mobile communications (GSM), long-term evolution (LTE), WiMax, or the like), etc.

[022] Using the I/O interface 401, the computer system 400 may communicate with one or more I/O devices 409 and 410. For example, the input devices 409 may be an antenna, keyboard, mouse, joystick, (infrared) remote control, camera, card reader, fax machine, dongle, biometric reader, microphone, touch screen, touchpad, trackball, stylus, scanner, storage device, transceiver, video device/source, etc. The output devices 410 may be a printer, fax machine, video display (e.g., cathode ray tube (CRT), liquid crystal display (LCD), light-emitting diode (LED), plasma, Plasma Display Panel (PDP), Organic light-emitting diode display (OLED) or the like), audio speaker, etc.

[023] In some embodiments, the computer system 400 may consist of the system 301. The processor 402 may be disposed in communication with a communication network 411 via a network interface 403. The network interface 403 may communicate with the communication network 411. The network interface 403 may employ connection protocols including, without limitation, direct connect, Ethernet (e.g., twisted pair 10/100/1000 Base T), transmission control protocol/internet protocol (TCP/IP), token ring, IEEE 802.11a/b/g/n/x, etc. The

communication network 411 may include, without limitation, a direct interconnection, local area network (LAN), wide area network (WAN), wireless network (e.g., using Wireless Application Protocol), Internet, etc. Using the network interface 403 and the communication network 411, the computer system 400 may communicate with a server 412 for receiving/obtaining JSON data. The network interface 403 may employ connection protocols include, but not limited to, direct connect, Ethernet (e.g., twisted pair 10/100/1000 Base T), transmission control protocol/internet protocol (TCP/IP), token ring, IEEE 802.11a/b/g/n/x, etc.

[024] The communication network 411 may include, but is not limited to, a direct interconnection, an e-commerce network, a peer to peer (P2P) network, local area network (LAN), wide area network (WAN), wireless network (e.g., using Wireless Application Protocol), Internet, Wi-Fi, and such. The communication network 411 may either be a dedicated network or a shared network, which represents an association of the different types of networks that use a variety of protocols, for example, Hypertext Transfer Protocol (HTTP), Transmission Control Protocol/Internet Protocol (TCP/IP), Wireless Application Protocol (WAP), etc., to communicate with each other. Further, the communication network 411 may include a variety of network devices, including routers, bridges, servers, computing devices, storage devices, etc.

[025] In some embodiments, the processor 402 may be disposed in communication with a memory 405 (e.g., RAM, ROM, etc. not shown in Figure 4) via a storage interface 404. The storage interface 404 may connect to memory 405 including, without limitation, memory drives, removable disc drives, etc., employing connection protocols such as, serial advanced technology attachment (SATA), Integrated Drive Electronics (IDE), IEEE-1394, Universal Serial Bus (USB), fibre channel, Small Computer Systems Interface (SCSI), etc. The memory drives may further include a drum, magnetic disc drive, magneto-optical drive, optical drive, Redundant Array of Independent Discs (RAID), solid-state memory devices, solid-state drives, etc.

[026] The memory 405 may store a collection of program or database components, including, without limitation, user interface 406, an operating system 407, web browser 408 etc. In some embodiments, computer system 400 may store user/application data, such as, the data,

variables, records, etc., as described in this disclosure. Such databases may be implemented as fault-tolerant, relational, scalable, secure databases such as Oracle ® or Sybase®.

[027] The operating system 407 may facilitate resource management and operation of the computer system 400. Examples of operating systems include, without limitation, APPLE MACINTOSH® OS X, UNIX®, UNIX-like system distributions (E.G., BERKELEY SOFTWARE DISTRIBUTION™ (BSD), FREEBSD™, NETBSD™, OPENBSD™, etc.), LINUX DISTRIBUTIONS™ (E.G., RED HAT™, UBUNTU™, KUBUNTU™, etc.), IBM™ OS/2, MICROSOFT™ WINDOWS™ (XP™, VISTA™/7/8, 10 etc.), APPLE® IOS™, GOOGLE® ANDROID™, BLACKBERRY® OS, or the like.

[028] In some embodiments, the computer system 400 may implement a web browser 408 stored program component. The web browser 408 may be a hypertext viewing application, such as Microsoft Internet Explorer, Google Chrome, Mozilla Firefox, Apple Safari, etc. Secure web browsing may be provided using Hypertext Transport Protocol Secure (HTTPS), Secure Sockets Layer (SSL), Transport Layer Security (TLS), etc. Web browsers 408 may utilize facilities such as AJAX, DHTML, Adobe Flash, JavaScript, Java, Application Programming Interfaces (APIs), etc. In some embodiments, the computer system 400 may implement a mail server (not shown in Figure 4) stored program component. The mail server may be an Internet mail server such as Microsoft Exchange, or the like. The mail server may utilize facilities such as ASP, ActiveX, ANSI C++/C#, Microsoft .NET, Common Gateway Interface (CGI) scripts, Java, JavaScript, PERL, PHP, Python, WebObjects, etc. The mail server may utilize communication protocols such as Internet Message Access Protocol (IMAP), Messaging Application Programming Interface (MAPI), Microsoft Exchange, Post Office Protocol (POP), Simple Mail Transfer Protocol (SMTP), or the like. In some embodiments, the computer system 400 may implement a mail client (not shown in Figure 4) stored program component. The mail client may be a mail viewing application, such as Apple Mail, Microsoft Entourage, Microsoft Outlook, Mozilla Thunderbird, etc.

[029] Some of the advantages of the present disclosure are listed below.

[030] Embodiments of the present disclosure discloses a generic system and method for automatic JSON data parsing. The method of the present disclosure provides an automatic way to parse the JSON data with different structures having “N” number of hierarchy and flatten

them so that it can be stored in table or a file and further used for data analysis and development. The method can parse any JSON data without any schema configuration or manual intervention into flat file or data frame (tables). The method also detects corrupted JSON records and filters them out without parsing into separate folders for analysis.

[031] Embodiments of the present disclosure reduces time consumption by automatically parsing the JSON data.

[032] Furthermore, one or more computer-readable storage media may be utilized in implementing embodiments consistent with the present disclosure. A computer-readable storage medium refers to any type of physical memory on which information or data readable by a processor may be stored. Thus, a computer-readable storage medium may store instructions for execution by one or more processors, including instructions for causing the processor(s) to perform steps or stages consistent with the embodiments described herein. The term "computer-readable medium" should be understood to include tangible items and exclude carrier waves and transient signals, i.e., be non-transitory. Examples include Random Access Memory (RAM), Read-Only Memory (ROM), volatile memory, non-volatile memory, hard drives, Compact Disc (CD) ROMs, DVDs, flash drives, disks, and any other known physical storage media.

[033] The described operations may be implemented as a method, system or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The described operations may be implemented as code maintained in a "non-transitory computer readable medium," where a processor may read and execute the code from the computer readable medium. The processor is at least one of a microprocessor and a processor capable of processing and executing the queries. A non-transitory computer readable medium may include media such as magnetic storage medium (e.g., hard disk drives, floppy disks, tape, etc.), optical storage (CD-ROMs, DVDs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, Flash Memory, firmware, programmable logic, etc.), etc. Further, non-transitory computer-readable media may include all computer-readable media except for a transitory. The code implementing the described operations may further be

implemented in hardware logic (e.g., an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc.).

[034] The illustrated steps are set out to explain the exemplary embodiments shown, and it should be anticipated that ongoing technological development will change the manner in which particular functions are performed. These examples are presented herein for purposes of illustration, and not limitation. Further, the boundaries of the functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternative boundaries can be defined so long as the specified functions and relationships thereof are appropriately performed. Alternatives (including equivalents, extensions, variations, deviations, etc., of those described herein) will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein. Such alternatives fall within the scope of the disclosed embodiments. Also, the words "comprising," "having," "containing," and "including," and other similar forms are intended to be equivalent in meaning and be open ended in that an item or items following any one of these words is not meant to be an exhaustive listing of such item or items or meant to be limited to only the listed item or items. It must also be noted that as used herein and in the appended claims, the singular forms “a,” “an,” and “the” include plural references unless the context clearly dictates otherwise.

[035] Furthermore, one or more computer-readable storage media may be utilized in implementing embodiments consistent with the present disclosure. A computer readable storage medium refers to any type of physical memory on which information or data readable by a processor may be stored. Thus, a computer readable storage medium may store instructions for execution by one or more processors, including instructions for causing the processor(s) to perform steps or stages consistent with the embodiments described herein. The term “computer readable medium” should be understood to include tangible items and exclude carrier waves and transient signals, i.e., are non-transitory. Examples include random access memory (RAM), read-only memory (ROM), volatile memory, non-volatile memory, hard drives, CD ROMs, DVDs, flash drives, disks, and any other known physical storage media.

[036] Finally, the language used in the specification has been principally selected for readability and instructional purposes, and it may not have been selected to delineate or

circumscribe the inventive subject matter. Accordingly, the embodiments of the disclosure are intended to be illustrative, but not limiting, of the scope of the disclosure.

[037] With respect to the use of substantially any plural and/or singular terms herein, those having skill in the art can translate from the plural to the singular and/or from the singular to the plural as is appropriate to the context and/or application. The various singular/plural permutations may be expressly set forth herein for sake of clarity.

GENERIC SYSTEM AND METHOD FOR JSON DATA PARSING

ABSTRACT

The present disclosure provides a method and system that is generic for automatic JSON data parsing. The system obtains the JSON data of various structures as input for parsing. During the parsing, if the JSON data is not valid or corrupt then the system filters out that JSON data and stores at a different location. Thereafter, the system parses the JSON data from top to bottom after analysing each key and stores the values of each key. During the parsing, a schema of the JSON data is created by the system. The system derives unique names of column after analysing the schema and stores the value corresponding to its column name. Upon deriving the column name and values, the system stores them in a data frame, or a file. The present disclosure prevents manually intervention and automatically parse the various JSON data.

Figure 1a

1/6

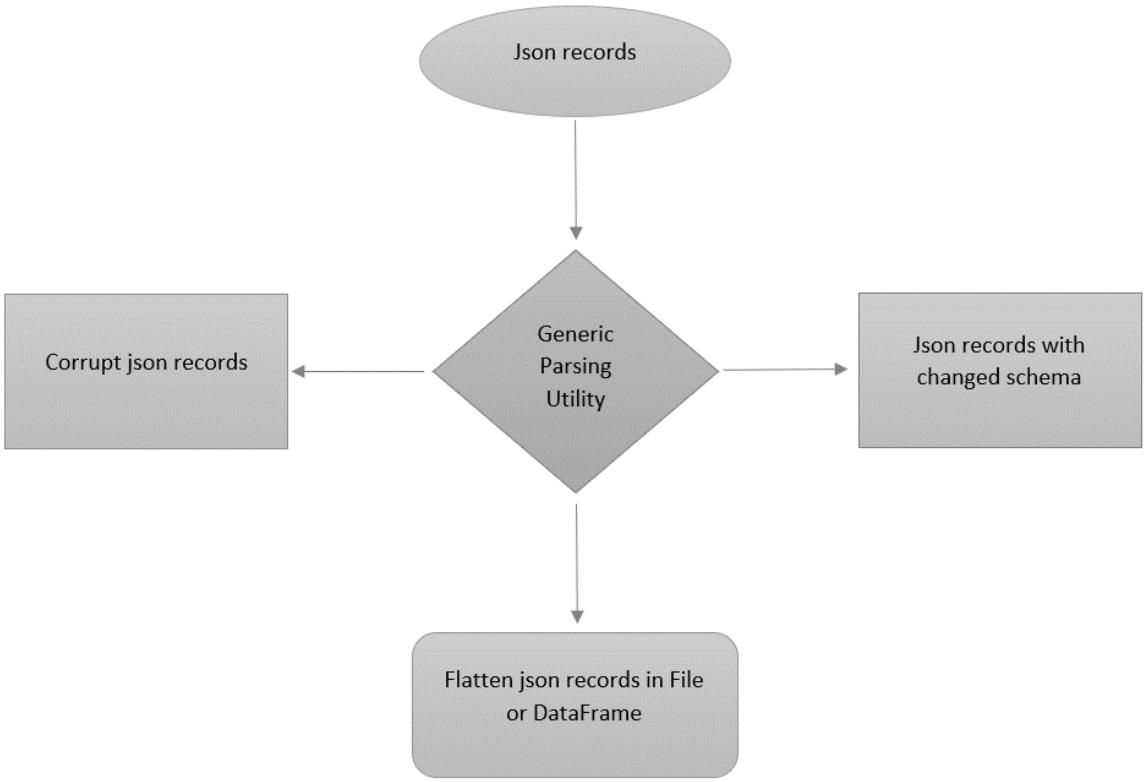

Json records
Generic Parsing Utility
Corrupt json records
Json records with changed schema
Flatten json records in File or DataFrame

Figure 1a

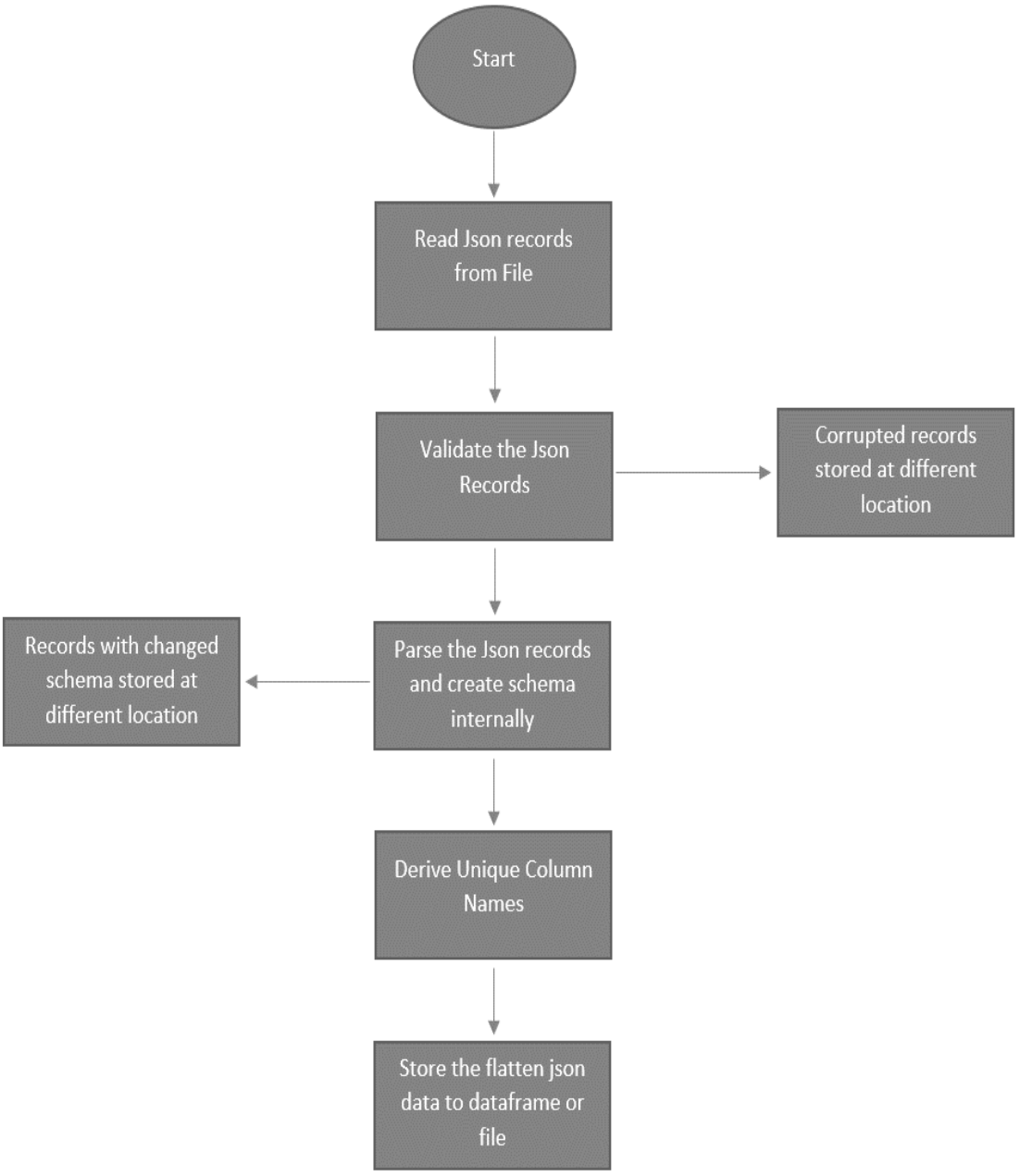

Start
Read Json records from File
Validate the Json Records
Corrupted records stored at different location
Records with changed schema stored at different location
Parse the Json records and create schema internally
Derive Unique Column Names
Store the flatten json data to dataframe or file

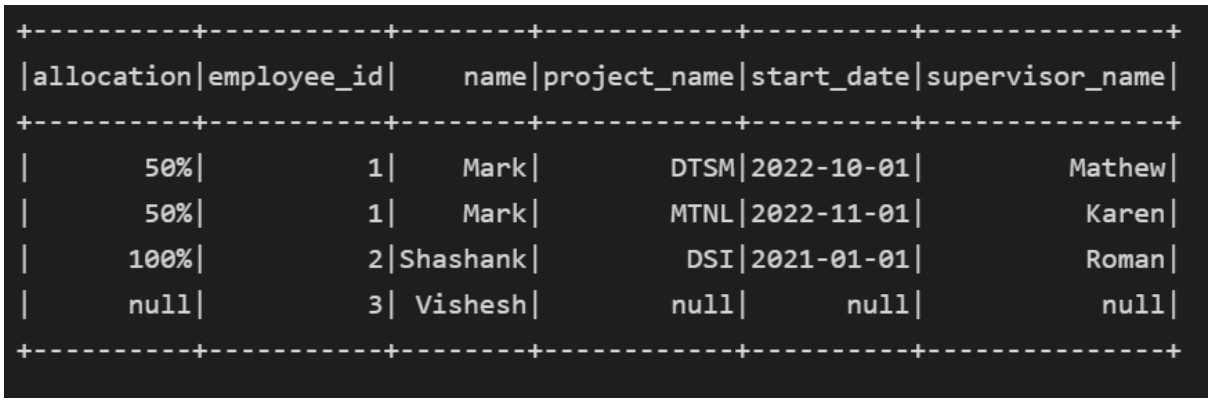
Figure 1b

3/6

```
[
    {
        "name": "Mark",
        "employee_id": "1",
        "project": [
            {
                "name": "DTSM",
                "supervisor_name": "Mathew",
                "start_date": "2022-10-01",
                "allocation": "50%"
            },
            {
                "name": "MTNL",
                "supervisor_name": "Karen",
                "start_date": "2022-11-01",
                "allocation": "50%"
            }
        ]
    },
    {
        "name": "Shashank",
        "employee_id": "2",
        "project": {
            "name": "DSI",
            "supervisor_name": "Roman",
            "start_date": "2021-01-01",
            "allocation": "100%"
        }
    },
    {
        "name": "Vishesh",
        "employee_id": "3"
    }
]
```


Figure 2a

4/6



allocation	employee_id	name	project_name	start_date	supervisor_name
50%	1	Mark	DTSM	2022-10-01	Mathew
50%	1	Mark	MTNL	2022-11-01	Karen
100%	2	Shashank	DSI	2021-01-01	Roman
null	3	Vishesh	null	null	null

Figure 2b

5/6

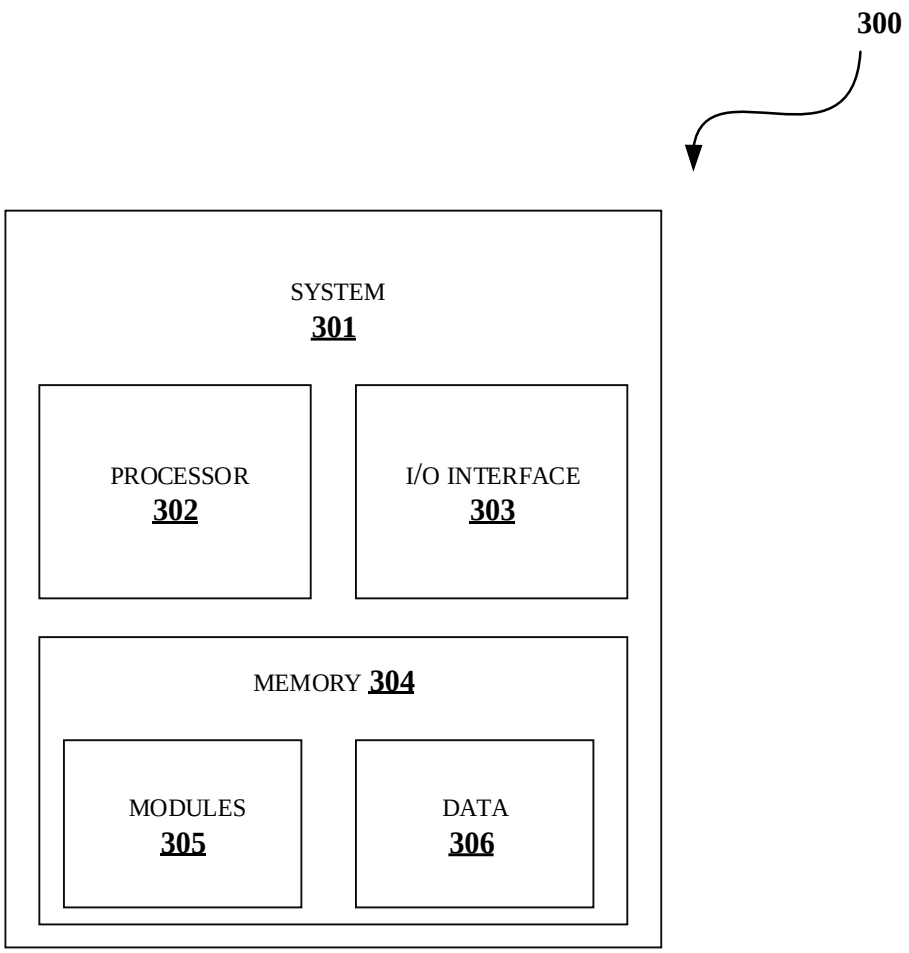

300
SYSTEM
301
PROCESSOR
302
I/O INTERFACE
303
MEMORY 304
MODULES
305
DATA
306

Figure 3

6/6

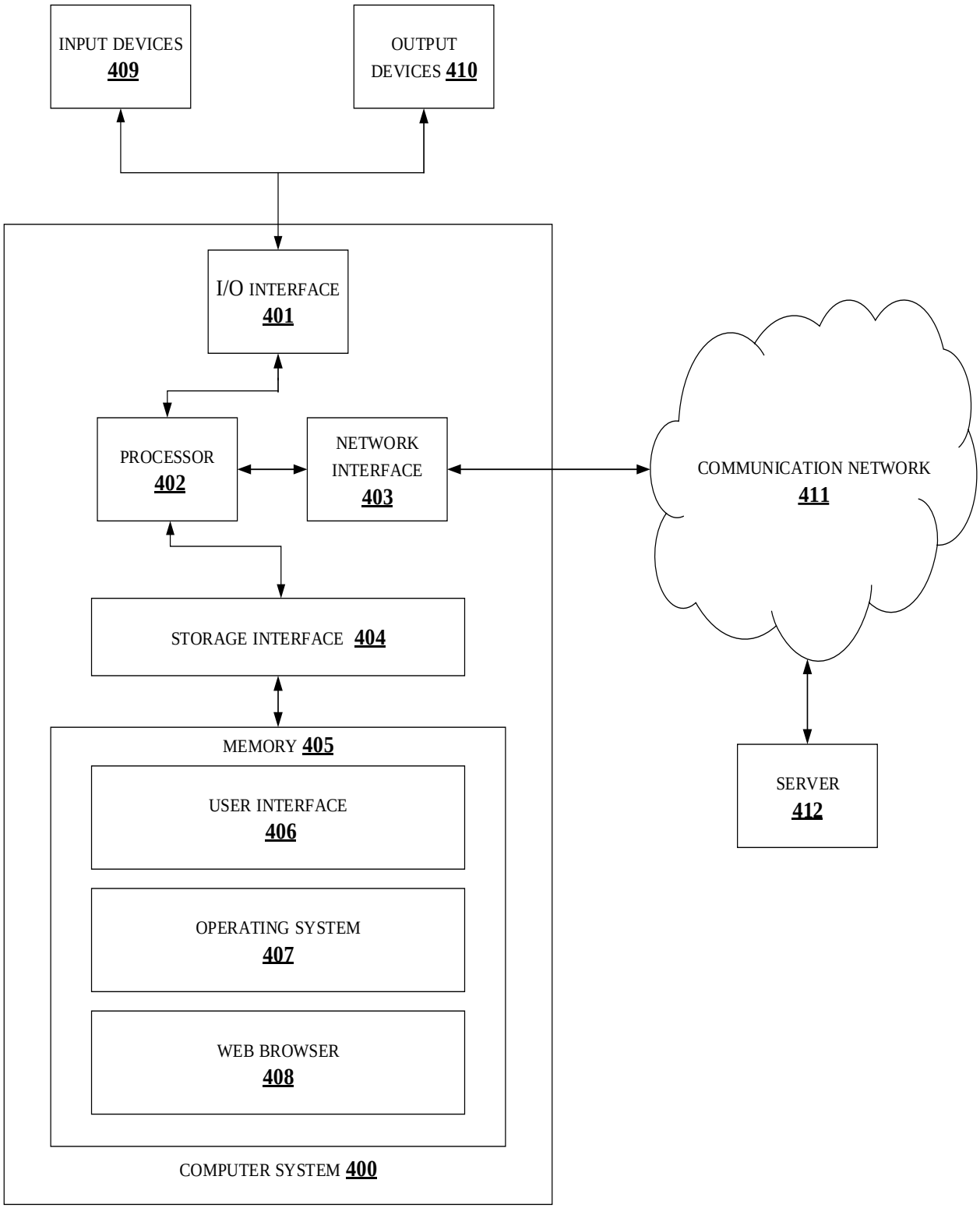


Figure 4

22