PATENT SEARCH REPORT

SEARCH TYPE: PATENTABILITY SEARCH

TITLE: Time File

PREPARED FOR: John Powers
Powers IP Law
3160 Day Court
Murrysville, PA 15668

CLIENT REFERENCE NUMBER: None

REPORT DATE: 02/03/2026

CARDINAL PROJECT MANAGER: Richard Ito

CARDINAL REFERENCE NUMBER: 5810.20

CARDINAL® INTELLECTUAL PROPERTY | IP Services
Trusted Name In IP Services

1603 Orrington Avenue
20th Floor
Evanston, IL 60201
www.cardinal-ip.com

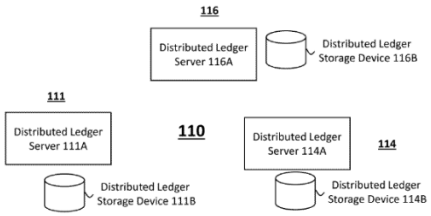
(847) 905-7122 phone
(847) 905-7123 fax
mail@cardinal-ip.com

We appreciate feedback
feedback@cardinal-ip.com

All References

#	Reference #	Title	Company	Authors	Pub/Issue Date
1	**US11687495**	System and method for managing collaborative multiuser document editing via a distributed ledger	CAPITAL ONE SERVICES, LLC	WYLIE, STEPHEN; TANG, Qiaochu; PRICE, MICAH; DAGLEY, GEOFFREY; HOOVER, JASON; HOOSHMAND, HABEEB	06/27/2023

Abstract	An electronic device may include logic to provide a trust credential for linking to a permissioned network over a local network, comprising a plurality of user devices; retrieve a document in an initial status for editing, corresponding to a first state of a distributed ledger, maintained by the permissioned network; generate a first change in the document, wherein the document is in a second status; send, to the permissioned network via the local network, the first change for storage in the distributed ledger, in a second state; retrieve the document in a third status, including a second change, performed subsequently to the first change, the second change being generated by a user device, external to the electronic device, and coupled to the local network; and link to the distributed ledger to retrieve a third state of the distributed ledger, the third state comprising the first change and the second change.
col 1, ln 38- col 2, ln 36	his disclosure presents various systems, components, and methods related to using a blockchain for collaborative document editing. Each of the systems, components, and methods disclosed herein provides one or more advantages over conventional systems, components, and methods. In one embodiment, an electronic device may include a storage device, and logic, at least a portion of the logic implemented in circuitry coupled to the storage device. The logic may provide a trust credential for linking to a permissioned network over a local network, the local network comprising a plurality of user devices; retrieve a document at a first instance for editing, the document being in an initial status, corresponding to a first state of a distributed ledger, maintained by the permissioned network; generate a first change in the document, wherein the document is in a second status after the first change; send, to the permissioned network via the local network, the first change for storage in the distributed ledger, in a second state; retrieve the document at a second instance when the document is in a third status, the third status including a second change, performed subsequently to the first change, the second change being generated by a user device, external to the electronic device, and coupled to the local network; and link to the distributed ledger to retrieve a third state of the distributed ledger, the third state comprising the first change and the second change. In another embodiment, a method may include providing a trust credential for linking an electronic device to a permissioned network; retrieving a document at a first instance, in an initial status, the initial status being represented in a first state of a distributed ledger maintained by the permissioned network; and generating a first change in the document, wherein the document is in a second status after the first change. The method may also include sending the first change to the permissioned network via a local network, for storage in the distributed ledger as a second state, derived from the first state; retrieving the document in a third status at a second instance, the third status including a second change, performed subsequently to the first change, the second change being generated by a user device, external to the electronic device, and coupled to the local network; and retrieving the distributed ledger from the permissioned network via the local network, the distributed ledger being in a third state comprising the first change and the second change. In a further embodiment at least one non-transitory computer-readable medium is provided, comprising a set of instructions that, in response to being executed on a computing device, cause the computing device to: provide a trust credential for linking an electronic device via a local network to a permissioned network; retrieve a document in an initial status; the initial status being represented in a first block of a blockchain maintained by the permissioned network; and generate a first change in the document, wherein the document is in a second status after the first change. The set of instructions cause the computing device to send the first change for storage, via the local network to the permissioned network, in a second block of the blockchain, derived from the first block; retrieve, the document, the document being in a third status, the third status including a second change, performed subsequently to the first change, the second change being generated by a user device coupled to the local network, external to the electronic device; and retrieve from the permissioned network via the local network a third block of the blockchain, the third block comprising the first change and the second change.
col 3, ln 28-39	Overall, communications over the permissioned network 120 may be encrypted and therefore secure represents a secure network. In particular embodiments the permissioned network 120 may represent a blockchain network, where the blockchain network may store edits to a collaborative document that is edited by one or more users through the user device 102, user device 104, and user device 106. In particular embodiments, the permissioned network 120 may be based on Hyperledger® Fabric (Hyperledger is a registered trademark of the Linux Foundation). As such, the permissioned network 120 may represent a private, permissioned network, or a private channel of network.
col 3, ln 40- col 4, ln 7	FIG. 2 depicts a general topology for one example of distributed ledger nodes 110, showing three different nodes, merely for the purposes of illustration. A given node may be maintained in a device, such as a computing device, such as a mainframe computer, a personal computer, laptop computer, notebook computer, and so forth. The embodiments are not limited in this context. In FIG. 2, a distributed ledger node 111 is represented by a distributed ledger server 111A and distributed ledger server storage device 111B. Likewise, a distributed ledger node 114 is represented by a distributed ledger server 114A and distributed ledger storage device 114B, while a distributed ledger node 116 is represented by a distributed ledger server 116A and distributed ledger storage device 116B. Notably, in the configuration of FIG. 2, a distributed ledger server and distributed ledger storage device may be embodied in the same device. In accordance with some embodiments of the disclosure, the different user devices may establish trust with one another in order to participate in collaborative editing of a document, as represented by collaborative document 130. In one implementation, the user device 106, user device 104, and user device 102 may be operated by users of a common organization, such a governmental organization, an educational organization, a business or other commercial entity. The embodiments are not limited in this context. In particular embodiments, the collaborative document 130 may represent any suitable document that may require or may receive inputs from multiple users. Said differently, the collaborative document 130 may represent a type of document that generally is changed, updated, formatted by more than one party. In some embodiments, the user device 102, user device 104, and user device 106 may establish trust with one another by sharing cryptographic information, such as sharing a private cryptographic key.

Fig. 2	 *FIG. 2*
col 4, ln 42-56	During editing of a document, information that is propagated to the distributed ledger or blockchain for storage in new block will represent any actions or operations performed on the document by any user of any of the user devices coupled to the local network, such as local network 112. Non-limiting examples of such operations include editing areas of text of the document, including coordinates of those text areas, placing objects in the document, such as images or non-text objects, and so forth. In accordance with some embodiments of the disclosure, a given change made to a collaborative document may be stored in a block of a blockchain as a “delta” or difference between the collaborative document in a first state and the collaborative document in a second state, immediately after the given change is entered. In accordance with other embodiments of the disclosure, a given change made to a collaborative document may be stored in a block of a blockchain as the entire collaborative document, in the second state, immediately after the given change is made.
col 4, ln 62- col 5, ln 21	FIG. 3A-3C shows one use scenario for collaborative editing, in accordance with embodiments of the disclosure. In the example of FIG. 3A, a group of users may form a team within a commercial organization, such as a lending institution, to process a loan application document 150, which document may represent a real estate loan document or an auto loan document in different non-limiting embodiments. The loan application document 150 may be made available for retrieval by authorized users in any suitable form. In the context of blockchain editing, such as using Hyperledger Fabric, various users may be granted access to a channel for creating a dedicated distributed ledger for storing and tracking edits to a given collaborative document. When user device 102 retrieves the loan application document 150 for editing, the user of user device 102 may find the loan application document 150 in an initial state. After performing an operation or group of operations on the loan application document 150, the changes to the loan application document 150 performed by the user of user device 102 may be saved as edit 1A to a ledger 201. Notably, the ledger 201 may be simultaneously propagated to multiple nodes of a blockchain system or similar distributed ledger system. For example, a whitelisted MAC address or whitelisted IP address may provide the appropriate trust credential for editing and storing changes to the loan application document 150 to a ledger that is operated by the permissioned network 120.
col 5, ln 54- col 6, ln 28	In embodiments employing a Hyperledger Fabric-based blockchain, the data for a collaborative document may be stored both as a blockchain ledger and a so-called world state, where the world state represents a current value of the document in question. Notably, the illustration of FIG. 3A and FIG. 3B present just general features of a ledger, where the different ledger blocks of the ledger 201 may represent the blockchain portion of data that is stored with respect to the edits to the loan application document 150. Different than the world state, in embodiments where the ledger 201 represents a blockchain ledger, the ledger blocks 201A and 201B are immutable and cannot be changed. Said differently, the ledger 201 may represent a blockchain, where the blockchain builds a continually increasing series of blocks. As known in the art, because the individual blocks of the blockchain are immutable, the different blocks of the blockchain (such as ledger 201) will effectively present a record that stores the chronological order of edits performed on the loan application document 150 in an unmutable fashion. In the example of FIG. 1B, the ledger block 201B lists the edit 1A and edit 2B, where the position of the edits may represent the order received for purposes of illustration. Thus, the ledger block 201B shows that the edit 1A to loan application document 150 was recorded firstly, while the edit 2B was recorded subsequently. In one use scenario according to embodiments of the disclosure, when a first user of user device 102 attempts to undo or revise a change made to the loan application document 150, made by a second user, such as the user of user device 106, the loan application document may be changed to the state that reflects the revision by the user of the user device 102. For example, starting from an initial state, the loan application document may undergo a series of edits, performed by different user devices, where the loan application appears the same as the initial state after the series of edits. However, each of the series of edits may be propagated as a new ledger entry in a new block of a blockchain. Thus, a series of three edits made to the loan application document 150 may be stored as three successive blocks in a blockchain, where the last block presents an immutable chronological record of the three edits, regardless of the present state of the loan application document 150.
col 6, ln 59- col 7, ln 3	In a collaborative mode where multiple users may interact with the same document over the same time span, the ledger 201 preserves the chronological record of the updating of the loan application document 150 so that details of the loan application process may be better and more accurately understood in a timely fashion. As noted, the collaborative editing procedures of the present embodiments, such as illustrated in FIGS. 3A-3C do not require connections to a single third party, where trusted user devices may communicate updates to the loan application document 150 for blockchain storage via mechanisms such as Bluetooth or TCP/IP.

2	**US20240275790**	DESCENDENT CASE ROLE ALIAS	OPEN TEXT CORPORATION	REDDY, SATYAPAL P.; JAYAKUMAR, MUTHUKUMARAPPA; HJORTSHOJ, JULIAN M.; MEENAKSHISUNDARAM, RAVIKUMAR	08/15/2024

Abstract	Case management systems and techniques are disclosed. In various embodiments, a definition is received that associates a descendant case role alias with a first case node at a first hierarchical level of a hierarchical data model, the definition further associating a permission with the descendant case role alias and referencing a referenced case role associated with a second case node at a second hierarchical level of the hierarchical data model. The definition is used to extend the permission to a user assigned to the referenced case role with respect to a case instance comprising the hierarchical data model.
para [0032]	A case model typically describes a case management system. Using a case model, one can model ad hoc actions with mini workflows, for example, as opposed to very structured process that defines an end-to-end business workflow. In various embodiments, a case model comprises a hierarchical/nested container model (sometimes referred to herein as a "hierarchical data model"), and may in addition define case roles, case phases (states), and/or permissions. In some embodiments, permissions may be defined for each case node and/or level in the hierarchy, and may vary in some embodiments based at least in part on the respective phases (states) of a state machine defined for a case node.
para [0037]	In various embodiments, a case model definition is embodied in an extensible Markup Language (XML) or other structured data file. A case management system and/or platform is provided, which is configured (e.g., by software) to load a case model definition, parse the definition, and create an instance of the case model based on the definition. Instance-specific attributes and/or state information or other metadata may be stored in a case model instance data store, e.g., a database. At runtime, the case model definition file and the case model instance data for a given instance are used by the disclosed case management system to implement the case model instance, including by performing processing and managing case model instance associated content per the case model definition, in light of the current values of the case model instance data for that instance.
para [0047]	FIG. 6 is a flow chart illustrating an example embodiment of a process to receive and store a case model. In some embodiments, the process of FIG. 6 is included in step 506 of FIG. 5. In the example shown, a definition of a hierarchical/nested data model is received (602). For example, a user interface that enables a developer to drag and drop case nodes onto a canvass and to indicate hierarchical relationships between case nodes may be provided and used by the developer to define a hierarchical/nested data model. A definition of case roles is received and stored (604). For example, a "loan application" case model may include user roles such as "loan initiator," "underwriter," "appraiser," etc. For each case node in the hierarchical/nested data model, a definition of metadata, behaviors, content (e.g., documents), states/phases (and transitions between states/phases), and/or permissions (e.g., by case role) is received (606). For example, in various embodiments a developer interface may be provided to enable a developer to select a case node and be presented with an interface to define a state machine for that case node.
para [0050]-[0054]	[0050] FIG. 9 is a block diagram illustrating an example of a hierarchical data model and associated state machine in an embodiment of a case management system. In various embodiments, the hierarchical data model and associated state machine of FIG. 9 may be included in a case model definition defined and/or deployed via a case management system such as case management system 206 of FIGS. 2 and 3. In the example shown, a state machine 902 has been defined for and associated with case node 704 of hierarchical/nested container model 700 of FIG. 7. [0051]In various embodiments, for any case node within the hierarchical/nested container model, a state machine can be defined and the actions that can be used to transition between different phases/states of the state machine defined for that case node may be specified. These actions could be used during runtime to transition between states. [0052]In the example shown in FIG. 9, a state machine 902 has been defined and associated with a specific case node in the hierarchical model shown in FIG. 7, specifically node "F11" (704). In various embodiments, a document or other content associated with node "F11"; traits, such as metadata and/or associated behavior associated with node "F11"; etc. may be transformed, reviewed, and/or otherwise involved with processing that may result, in a given case model instance, in transitions being made between states of the state machine 902 defined for case node "F11" in this example. [0053]In various embodiments, enabling a state machine to be defined and associated with a case node comprising a hierarchal/nested container model provides a flexible, dynamic framework within which ad hoc actions and/or information can be responded to, in a manner determined dynamically based on the circumstances of a given instance of a case, with the result that the actions and/or processing performed at a given case node, and/or the consequences of such actions and/or processing, may be different for one instance of the case model than for another instance of the case model. [0054]In various embodiments, a state machine engine may be included in a case management system, such as case management system 206 of FIG. 2, to enable a state machine defined for a case node, such as state machine 902 of FIG. 9, to be implemented and associated functionality to be provided. For example, in some embodiments, case management module 310 of FIG. 3 may include a state machine engine. In some embodiments, the state machine engine may receive and parse state machine definition portions of a case model definition, and may use such portions to create and manage runtime data structures associated with the respective defined states (phases) of the state machine and transitions between them. In some embodiments, state variables associated with a current state of a case node-specific state machine for a given instance of a case model may be stored persistently with other case management instance data, for example in a case instance data store such as data store 312 of FIG. 3.
para [0065]	In various embodiments, permissions set for a case role with respect to a case node may be defined dynamically, e.g., by reference to the respective phases/states of a state machine associated with the case node. For example, a case role may have a first set of permissions at a case node when the case node's state machine is in a first phase/state, and a second (different) set of permissions at the case node when the case node's state machine is in a second phase/state. In some embodiments, in each state one or more permissions associated with causing transitions to be made to one or more other states may be set based on case role. In some embodiments, permission may be set by case role to indicate which case roles will be able to assign users to case roles and/or to change such assignments.
para [0066]-[0067]	[0066] FIG. 13 is a flow chart illustrating an example embodiment of a process to define hierarchical permissions conditioned on case node state. In some embodiments, the process of FIG. 13 may be implemented by a case management system, such as case management system 206 of FIG. 2. In the example shown, definition of permissions by case node phase/state begins at a first phase/state of a state machine associated with a case node (1302). Case role(s) is/are associated with the phase state (1304), and for each case role associated with the case node phases/state permissions are set for the case role with respect to the case node when the state machine of the case node is in that phase/state (1306). Processing continues with respect to the respective phases/states of the state machine of the case node, until permissions have been defined for each phase/state (1308, 1310). [0067] In various embodiments, hierarchical and/or conditional (e.g., by case node phase/state) permissions defined as described in connection with FIGS. 12 and 13 may be embodied in a case model definition. At runtime, when a case instance is instantiated based on the case model definition, data comprising the case model definition is parsed to determine and create runtime data structures reflecting the hierarchical data model of the case model, and with respect to each node permissions are set by case role (and, as applicable, conditioned on case node phase/state). Permission-related services associated with a runtime environment in which the case instance is realized are used in some embodiments to enforce hierarchical and/or conditional (e.g., by phase/state) permissions as defined in the case model definition. For example, in some embodiments, the ability of a given user to access and/or perform requested operations with respect to metadata and/or content, to initiate transitions between states, and/or to assign or modify the assignment of users to case roles with respect to the case instance is determined by the runtime environment based on which (if any) case role(s) the user has been assigned with respect to the

	case instance, and the permission(s), if any, associated with such case role(s) with respect to an applicable case node in a current phase/state of a state machine associated with the case node.

3	**US12166858**	System and method for storing contract data structures on permissioned distributed ledgers	ROYAL BANK OF CANADA	KOMANDUR, VENKATADRI KAUSHIK; ZUGIC, GORAN	12/10/2024

Abstract	A computer implemented method for maintaining a synchronized distributed ledger data structure across a plurality of computing nodes each configured to enforce a synchronization protocol for coordinating updates of the synchronized distributed ledger data structure to establish an inter-group reconciliation mechanism across one or more trusted entities having immutable transaction records stored in the form of blockchain data structures recording approvals from one or more parties corresponding to the plurality of computing nodes. In some aspects, regulator nodes are contemplated.
col 1, ln 41- col 2, ln 64	A distributed ledger-based system is described that is adapted to manage a reconciliation-free, contract-based workflow between multiple parties who do not trust each other. A blockchain-based protocol for contract creation, agreeance, and performance, adapted to handle both processes that on a path to execution and includes paths where the execution of a multi-party contract can fail. The specific approach is adapted such that contract performance can be achieved successfully between untrusting parties. The distributed ledger-based system is provided as a computational approach and corresponding systems for implementing electronic reconciliation of contracts on permissioned distributed ledgers. Distributed ledgers are utilized to store and track status transitions of an agreement at various points of time, using a state machine that models the contract statuses and possible states. For example, a state change can include 11 possible states, where a last state is a final/accepting state, and a number of intermediate states are possible. At various events and states, there may be various invocations of chaincode and updates representing events occurring in relation to the agreement. These state changes are reflected in a blockchain data structure stored on a distributed ledger mechanism distributed across various computing nodes.
col 2, ln 36-67	When an input set of data fields is received at a first node (e.g., through a user interface), the input set of data fields is processed as the fields represent parameters of an agreement between a plurality of parties. For example, NY would like to establish an agreement with London in respect of sharing the cost of an underlying services contract with a hundred licenses (NY only needs fifty, so it wishes to share fifty with London and the cost should later be reconciled between the offices). In this case, NY could serve as the first node, and the input set of data fields can be received in NY. The first node then instantiates an agreement represented in the blockchain data structure incorporating at least the input set of data fields, a series of one or more required approval signals, each required approval signal corresponding to a hidden unique primary key. As the network is a trusted network, blocks storing, among others, transaction data, approvals, and sets of input fields can simply be added to the distributed ledger without proof of work or proof of stake protocols. The set of data elements representing the initial fields of the agreement, when generated, includes a hidden unique primary key (e.g., a seed) that can be sequentially or pseudo-randomly generated. The hidden unique primary key can be used with a one-way hash to expose information regarding the initial agreement, and in some embodiments, the hidden unique primary key is never accessible by any individual human. The computing nodes may interact with the blockchain by providing approvals (e.g., if a corresponding approval is required), through providing an approval signal signed using a private key corresponding to a second party of the plurality of parties.
col 3,l n 1-29	When all the approvals are obtained, the blockchain may be rendered immutable yet accessible, for example, through a consensus mechanism that does not allow further block transactions on the blockchain data structure. In another aspect, responsive to receiving, at the second computing node of the plurality of computing nodes, a disapproval signal signed using the private key corresponding to a second party of the plurality of parties and including a new set of input data fields, a separate agreement is instantiated. In another aspect, the synchronized distributed ledger data structure maintains one or more cryptographically immutable records of transactions established between business entities associated with a corresponding node of the plurality of computing nodes, each node potentially representing a different business unit. In another aspect, each node of the plurality of computing nodes is a trusted computing device. In another aspect, each node of the plurality of computing nodes includes at least two trusted computing devices: a first computing device configured for actively conducting operations on the distributed ledger data structure, and a second computing device configured to store a redundant copy of the distributed ledger data structure and to switch over to conducting the operations on the distributed ledger data structure responsive to an event of systems failure of the first computing device.
col 21, ln 62- col 23, ln 6	Approach 2: Lock editable data using Redis in the Node layer: In this approach, data (e.g Agreement) that can be updated simultaneously by multiple users will be intercepted by the Node layer. The UUID of the agreement data along with the userId will be cached in Redis with a time-to-expire value before delivering to the UI. This way the second request for the same data will receive a lock error and will not be allowed to make updates until either the first user has completed editing the data and the cached key in Redis is cleared or the lock has expired. With this approach, the user will not receive an error after submitting the transaction. The disadvantage of this approach is that data is cached in Redis and there is an additional dependency on it. Another cons belongs to the fact that when the chaincode is accessed via interfaces that enables direct chaincode calls (e.g., CLI) the locking will not be provide. Note that there is still a very small probability based on timing of the request messages that Redis might not be able to set the lock which will result in two users attempting to update the same data but on submission, Fabric's MVCC as described in approach 1 will allow only one of the user to update the data. The locking for Agreement updates will support UI interface. Also the global lock for creation and update of any object during the batch processing within the maintenance window will also be supported. Redis takes care of a lock expiration.

	Approach 3: Add version information in the chaincode query response: In this approach, when the UI user requests data for updating or deleting, the chaincode will also return a version information. The version is incremented every time an object is updated. When the UI layer updates the data, the chaincode compares the version received in the request with the version currently in the ledger. If the version does not matches, an error will be generated by the chaincode and will be returned to the UI user. This approach ensures data consistency. If two users concurrently try to update the same version of an agreement but they save their updates at different time the last one who saves will receive an error since chaincode increments version value for each save.

4	**US10796086**	Selectively controlling modification states for user-defined subsets of objects within a digital document	MICROSOFT TECHNOLOGY LICENSING, LLC	LEWBEL, HANNAH REBECCA	10/06/2020

Abstract	The disclosed technologies enable users to view a particular modification state of select objects of a document without impacting the display of unselected objects. For example, a user can create a number of objects located at different sections of a document. The user may perform modifications to each object that result in a sequence of modification states for each object. The user may select a subset of the objects by the use of an input gesture. The user can then view a specific modification state of the selected objects that existed at a first point in time, while viewing a modification state of other objects that existed at another point in time. In this way, the user is enabled to concurrently view different versions of each object even if these particular versions did not coexist at any point in time during the user's editing process.
col 2, ln 3-50	The disclosed technologies enable users to view a particular modification state of selected objects of a document without impacting the display of unselected objects. For example, a user can create a number of objects, e.g., drawing objects or text, located at different sections of a document. The user may perform modifications to each object that result in a sequence of modification states for each object. The user may select a subset of the objects by the use of an input gesture, which can be an inking gesture, a voice command, or any other suitable input for selecting objects. The user can then view a specific modification state of the selected objects that existed at a particular point in time, while viewing a modification state of other objects, e.g., the unselected objects, that existed at another point in time. In this way, the user is enabled to concurrently view different versions of each object even if the particular versions did not coexist at any point in time during the user's editing sequence. In some configurations, the user may select a particular object or a group of objects and selectively rollback (e.g., “undo”) modifications that were applied to the selected objects without rolling back modifications that were applied to other objects. In one illustrative example, a system causes a display device to render multiple objects at different sections of a digital document. For example, a user may be operating an application to edit a first sketch that is located at a first content section of the digital document and a second sketch that is located at a second content section of the digital document. The application may cause the display device to concurrently render current versions of both the first sketch and the second sketch. While the user is editing the sketches, the application may generate historical modification data that defines modification states in association with individual objects at various times throughout the user's editing process. The individual objects may be graphical elements that form the first sketch and the second sketch. For example, the individual objects may be raster-based and/or vector-based objects that graphically represent lines, curves, and/or points of one or more colors. The system may receive a first user input that indicates a particular content section of the digital document. For example, the user may want to return the first sketch to a previous version (i.e., some version that existed prior to the current version that is being displayed) and, therefore, may select the first content section at which the first sketch is located. The first input may be in the form of an inking gesture, voice command, or other input indicating a selection of the first sketch.
col 10, ln 50- col 11, ln 18	FIGS. 2A and 2B correspond to an implementation in which the user input received via the UI control 118 controls the modification state of only those individual graphical objects 112 that are located within some user-defined content section 116. However, in other implementations, implementation parameters may be reversed so that user input received via the UI control 118 controls the modification state of only those individual graphical objects that are located outside of some user-defined content section 116. FIGS. 3A and 3B convey details of such an alternative implementation. Although implementations of the present disclosure are predominantly described in the context of a content section being a single content section defined by a single boundary 206, it is within the scope of the present disclosure that a content section be defined by multiple boundaries 206 that enclose multiple non-contiguous portions of a digital document. For example, a user may define three non-contiguous boundaries that each fully encompass separate portions of a digital document 104. The aggregate of the graphical objects 112 included within these three separate portions may then be identified as the subset of graphical objects with respect to which a user is then able to control the modification state. For example, a single user input that is received in association with a single UI control 118 may cause the modification states of all objects 112 within each discretely defined boundary to change accordingly. Additionally, or alternatively, different user-defined boundaries 206 may be each individually associated with a corresponding UI control. For example, if a user has defined three discrete boundaries 206 within a digital document, then in some implementations a user may be provided with an individual UI control in association with each individual discrete boundary 206 so that the user can control the specific modification state that objects within each separate boundary are rendered in accordance with.
col 11, ln 38-55	Turning now to FIG. 3B, illustrated is an exemplary UI control 118 that facilitates user input that indicates a time within a modification history of the digital document 104 at which to render the determined subset of graphical objects 112 that reside outside of the content section 116. FIG. 3B is similar to FIG. 2B with the exception that user input that is received in association with UI control 118 directly affects the rendered modification state of the first sketch 110(1) due to the subset of graphical objects 112 forming the first sketch 110(1) residing outside of the content section 116. Stated plainly, in FIG. 3B the graphical objects 112 within the content section 116 that is defined by the user remain frozen (e.g., “locked”) whereas the other graphical objects 112 in the first sketch 110(1) that are outside of the content section 116 may be altered via manipulation of the UI control 118. In the illustrated example, the manipulation of the UI control 118 causes the first sketch 110(1) to revert from the fourth version back to the second version.
col 12, ln 14-51	Turning now to FIG. 4A, illustrated is an exemplary sequence of operations that are performed to duplicate graphical objects 112 that are contained within a content section 116 and to further duplicate modification history data that corresponds to these graphical objects 112. FIG. 4A is similar to FIG. 2A with the exception that rather than selecting a “Rollback Section Edits” option, in FIG. 4A the curser UI element 202 is used to select a “Copy w/ Modification History” option. In response to the user(s) creating the boundary 206 to define the content section 116 and then selecting the “Copy w/ Modification History” option, a duplicative instance of a sketch 110 is created within the digital document 104. Furthermore, modification history data that was recorded by an application during creation of the sketch 110 (and/or the various versions thereof) remains tied to the duplicative instance of the sketch 110. FIG. 4B illustrates both an original instance of the sketch (labeled 110) and a duplicative instance of the sketch (labeled 110′). In various implementations, one or more UI controls 118 may be displayed in association with one or both of the original instance of the sketch 110 or the duplicative instance of the sketch 110′. For example, as illustrated, a first UI control 118(1) is shown in association with the original instance of the sketch 110 and a second UI control 118(2) is shown in association with the duplicative instance of the sketch 110′. In

	FIG. 4B the first UI control 118(1) has been placed at a position along a first graphical timeline object 208(1) to control the modification state at which the display device 108 renders the original instance of the sketch 110. Here, the rendered modification state of the original instance of the sketch 110 corresponds to a point in time that is between T1 and T2. At the same time, the second UI control 118(2) has been placed at a position along a second graphical timeline object 208(2) to control the modification state at which the display device 108 renders the duplicative instance of the sketch 110′. Here, the rendered modification state of the duplicative instance of the sketch 110′ corresponds to T4.
col 13, ln 36- col 14, ln 31	Turning now to FIG. 6, illustrated is an exemplary implementation in which a histogram 600 is displayed in association with a defined content section 116 to visually indicate modification activity owners that correspond to editing activity that occurred during an editing process. An exemplary histogram 600 may include one or more activity owner indicators 602 that are displayed in association with a graphical timeline object 208. In the specifically illustrated example, the histogram 600 includes a first activity owner indicator 602(1) through a fourth activity level indicator 602(4). The first activity owner indicator 602(1) corresponds to a first range of time along the graphical timeline object 208 during which a first user was making user modifications with respect to the graphical objects 112 within the content section 116. The second activity owner indicator 602(2) corresponds to a second range of time along the graphical timeline object 208 during which a second user was making user modifications with respect to the graphical objects 112 within the defined content section 116. As illustrated, individual ones of the activity owner indicators 602 may include graphical representations such as, for example, avatars that are visually representative of the individual users. In this way, if a user is interested in reviewing user modifications that were entered by a specific user (e.g., a boss or colleague), then that user can simply slide the UI control 118 into a region of the graphical timeline object 208 that is adjacent to (e.g., below, above, to the right of, to the left of) a graphical representation of the specific user. In some implementations, the histogram 600 is configured to indicate ranges of time in which one or more users collaborated with respect to making user modifications to graphical objects 112 within the defined content section 116. As illustrated, for example, the third activity owner indicator 602(3) corresponds to a third range of time along the graphical timeline object 208 during which both the first user and the second user were concurrently making user modifications with respect to the graphical objects 112 within the defined content section 116. In this way, a user will be able to quickly and visually identify portions of a modification history of the digital document 104 during which collaborative modifications occurred. In some implementations, one or more indicators may be rendered to graphically indicate regions of the digital document 104 at which user(s) were making modifications to objects 112 outside of the defined content section 116 at the specific time where the UI control 118 is set and/or within a threshold amount of time from the specific time where the UI control 118 is set. For example, in the illustrated example, the UI control 118 is set to a specific time within the aforementioned third range of time during which both the first user and the second user were concurrently making user modifications with respect to the graphical objects 112 within the defined content section 116. Suppose that at this time the second user was also making modifications to the graphical objects 112 that are located outside of the defined content section 116. To graphically indicate this information, the fourth activity owner indicator 602(4) is rendered outside of the defined content section 116 and identifies one or more specific regions of the digital document 104 that was concurrently being edited by the second user during the third range of time.
col 16, ln 9-43	Turning now to FIG. 8A, illustrated is an exemplary implementation in which a first section 802(1) of a word processing digital document 804 is locked (e.g., frozen) while a second section 802(2) is unlocked with respect to a review modification history tool 806. In the illustrated example, the first section 802(1) of the word processing digital document 804 corresponds to a “Detailed Analysis” section and the second section 802(2) of the word processing digital document 804 corresponds to a “Executive Summary” section. To visually indicate that the first section 802(1) is locked, the illustrated implementation includes a hatching pattern rendered over the first section 802(1) along with a “locked padlock” symbol. In contrast, the second section 802(2) includes no such hatching pattern and includes an “unlocked padlock” symbol. As illustrated, a user may dynamically change a currently rendered modification state of the unlocked section(s) of the word processing digital document 804 by moving a curser UI control 202 to manipulate a position of a UI control 118 with respect to a graphical timeline object 208. In various implementations, an application that is rendering the word processing digital document 804 may be configured to emphasize differences between a most recently saved version of the word processing digital document 804 and the version selected via the UI control 118. For example, as illustrated, a user has manipulated the UI control 118 to cause the application to render a version of the digital document 804 as it existed at 3:04 PM on Jul. 21, 2018. In response, the application is rendering a comparison between the user selected version and the most current version. More specifically, text characters that have since been deleted from the second section 802(2) are shown with strike-through font formatting whereas text characters that have since been added to the second section 802(2) are shown with underlined font formatting.

5	**US20250291512**	System and Method for Decentralized Data Storage and Retrieval with Intelligent Sharding and Provenance Tracking	Vannadium, Inc.	GILCHRIST, RICHARD; KREBS, Randy; LYONS, Skylar; SCOTT, Jonathan Boyd	09/18/2025
Abstract	Systems and methods for decentralized data storage and retrieval within a distributed computing system include partitioning datasets into encrypted data shards that are each associated with a unique identifier. A processing system operating within the network may distribute these shards across storage nodes selected based on performance metrics such as latency, storage utilization, and reliability. Metadata for each shard, including its identifier, unique cryptographic key, and node location, may be recorded in a tamper-evident distributed ledger. Upon receiving a retrieval request, the system may query the ledger to identify shard locations, retrieve the shards in parallel, validate their integrity by comparing ledger metadata with metadata from storage nodes, decrypt the shards using their respective cryptographic keys, and reconstruct the dataset. These systems and methods enhance data security, integrity, and fault tolerance while allowing efficient and verifiable decentralized storage and retrieval in dynamic, distributed environments.				
para [0005]-[0012]	[0005]Various aspects include methods performed by a processing system operating within a distributed computing system for decentralized data storage and retrieval. In some aspects, the methods may include receiving a dataset for storage, partitioning the dataset into a plurality of data shards, in which each data shard may be associated with a unique shard identifier and may include a portion of the dataset, encrypting each data shard using a unique cryptographic key, distributing the plurality of data shards across a plurality of storage nodes in the distributed computing system, in which each storage node may be selected based on network performance metrics which may include latency, storage utilization, and reliability, recording metadata associated with each data shard in a tamper-evident distributed ledger, the metadata which may include the unique shard identifier, the unique cryptographic key, and the storage node location, receiving a retrieval request specifying the dataset, querying the tamper-evident distributed ledger to determine the storage node locations of the data shards associated with the dataset, retrieving the data shards from the identified storage nodes in parallel, validating the integrity of each retrieved data shard by comparing metadata stored in the tamper-evident distributed ledger with metadata retrieved from the storage nodes, decrypting each validated data shard using the unique cryptographic key associated with the data shard, and reconstructing the dataset from the validated and decrypted data shards based on their shard identifiers.				

	[0006]In some aspects, recording metadata associated with each data shard in the tamper-evident distributed ledger further may include recording a lineage chain for each data shard. In some aspects, the lineage chain may include a record of each transformation or modification performed on the data shard, the entity responsible for the modification, and a cryptographic signature of the modification. In some aspects, distributing the plurality of data shards across the plurality of storage nodes further may include dynamically assigning data shards to storage nodes based on real-time network conditions and predefined performance policies that balance tradeoffs between data retrieval latency and fault tolerance. [0007]Some aspects may further include dynamically adjusting the size of the data shards based on real-time network conditions. In some aspects, validating the integrity of each retrieved data shard further may include detecting unauthorized modifications to the data shard by performing a cryptographic hash comparison between the metadata stored in the tamper-evident distributed ledger and the metadata retrieved from the storage node storing the data shard. Some aspects may further include storing operation-specific details in a tamper-evident distributed ledger to create a tamper-evident record of operations performed on the data shards. [0008]Some aspects may further include identifying a compromised data shard based on a mismatch between the metadata stored in the tamper-evident distributed ledger and the metadata retrieved from the storage node, retrieving a validated copy of the compromised data shard from an alternate storage node, and updating the tamper-evident distributed ledger to record the recovery operation. Some aspects may further include identifying data shards likely to experience high access demand based on historical access patterns and real-time network conditions, caching the identified data shards on strategically selected storage nodes to optimize retrieval efficiency, and recording updates to cache assignments in the tamper-evident distributed ledger. [0009]In some aspects, identifying data shards likely to experience high access demand further may include applying a machine learning model trained on historical access logs and network performance metrics to predict future access patterns for the data shards. In some aspects, distributing the plurality of data shards across the plurality of storage nodes further may include assigning a jurisdictional attribute to each data shard based on a unique identifier linking the data shard to an owning entity, and restricting storage node selection for the data shard to nodes located within a jurisdiction compliant with the laws and regulations applicable to the owning entity. [0010]In some aspects, encrypting each data shard using the unique cryptographic key may include generating, for each data shard, a cryptographic hash representing the content of the data shard, the cryptographic hash being linked to a provenance record identifying an origin and a modification history of the data shard, and reconstructing the dataset from the validated and decrypted data shards based on their shard identifiers may include reconstructing the dataset by retrieving and combining at least a subset of the plurality of data shards from the decentralized network based on their respective shard identifiers and cryptographic hashes, in which the integrity of the shards may be validated during reconstruction. In some aspects, encrypting each data shard using the unique cryptographic key may include encrypting each data shard using an asymmetric cryptographic technique. In some aspects, the provenance record further may include a lineage chain linking the data shard to one or more prior versions, a timestamp indicating when the data shard was created or modified, and a unique identifier associated with the entity responsible for the creation or modification. [0011]In some aspects, the lineage chain may include cryptographic evidence of each transformation, access, and modification associated with the data shard. In some aspects, recording metadata in the tamper-evident distributed ledger further may include applying a consensus mechanism to validate updates to the metadata, the consensus protocol which may include proposer node selection based on node reliability and performance metrics, proposal generation for metadata updates, and cryptographic signature validation of the proposals. In some aspects, the consensus mechanism uses a proof-of-stake algorithm for validating updates to the distributed ledger, and may include periodic integrity checks to detect tampering before shard retrieval operations. [0012]In some aspects, encrypting each data shard using a unique cryptographic key further may include managing the cryptographic keys in a fault-tolerant key management system operatively connected to the distributed computing system, and securely distributing the cryptographic keys to authorized entities based on access control policies recorded in the tamper-evident distributed ledger. In some aspects, retrieving the data shards in parallel further may include prioritizing retrieval operations based on a weighted combination of latency metrics, bandwidth availability, and node reliability metrics for each storage node storing the data shards. Some aspects may further include generating a security audit report that may include access histories, lineage records, and transformation events for each data shard, and presenting the security audit report to authorized entities via a secure interface.
para [0117]	In some embodiments, the distributed ledger 214 may be, may include, or may be included in a decentralized database configured to store tamper-evident and immutable records of transactions, event data, and metadata across the distributed computing framework. The distributed ledger 214 may include a consensus component 242 that validates transactions and ensures consistency across the network. The distributed ledger 214 may also include immutable records 244 that store metadata and operational logs in a format that prevents overwriting or alteration. Each record may include cryptographic hashes and digital signatures to maintain traceability and verify data integrity. These immutable records 244 may support auditability and compliance with regulatory requirements by providing an accurate and verifiable history of system operations. In addition, the distributed ledger 214 may interact with other components, such as the tokenization platform 234 and shield component 238, to record token operations and system security events.
para [0329]	In some embodiments, the processing system may store an encrypted document in a decentralized storage network and control document access using permissions encoded in the distributed ledger. In some embodiments, the methods may further include detecting an anomaly in an entity's activity pattern, generating an anomaly alert, and updating an event memorialization structure with the detected anomaly. In some embodiments, the processing system may display a visual representation of a hierarchical provenance record associated with the entity's digital asset transactions and identity tokens. In some embodiments, the methods may further include executing multi-signature transactions following verification of cryptographic signatures recorded in the distributed ledger. In some embodiments, the methods may further include verifying entity authentication credentials prior to granting access to identity-based tokens stored in the distributed ledger. In some embodiments, the methods may further include controlling access to digital asset balances using role-based access control policies stored in the distributed ledger. In some embodiments, the processing system may detect attempted modifications of immutable transaction records and generate alerts indicating potential security anomalies. In some embodiments, event records may be timestamped using a consensus mechanism supported by the distributed ledger, ensuring chronological consistency.
para [0340]	In some embodiments, the processing system may detect anomalies in entity activities, generating alerts recorded as events within the event memorialization structures linked to identity-based tokens. For example, role-based access controls encoded in smart contracts may restrict viewing or editing of event memorialization data to authorized entities.
para [0346]-[0349]	[0346] FIG. 11A is a process flow diagram illustrating a method 1100 of securely storing electronic documents using decentralized storage and enforcing access control through multi-signature authentication in accordance with some embodiments. The method 1100 may be performed by a processing system including one or more processors and the components described in this application (e.g., FIGS. 1-6). The processing system may execute software or firmware to perform some or all operations of method 1100. References to a “processing system” encompass alternative hardware configurations implementing portions or all of method 1100.

	[0347]In block 1102, the processing system may receive a document uploaded by a user. For example, the processing system may accept electronic files including PDF documents, images, or text documents through a secure application interface accessed via a web browser or mobile device. In some embodiments, the processing system may retrieve documents automatically from cloud storage associated with a user account at scheduled intervals or when triggered by predefined events. For example, the processing system may periodically query cloud storage repositories linked to user credentials to collect recently modified documents. In some embodiments, the processing system may validate document formats, sizes, or metadata against predefined criteria to confirm compatibility with encryption and storage methods. In some embodiments, the processing system may be further configured to maintain audit records logging document uploads and associated user activities. [0348]In block 1104, the processing system may encrypt the document using a cryptographic key associated with the user. For example, the processing system may apply a symmetric encryption algorithm such as Advanced Encryption Standard (AES-256) to generate encrypted data accessible only using the user's cryptographic key. In some embodiments, the processing system may use asymmetric encryption to generate a public-private key pair for securing the user's identity and data. For example, the private key may reside securely within dedicated HSMs or user-controlled secure storage devices. In some embodiments, the processing system may incorporate timestamps, user identity attributes, or document-specific metadata into the encryption key generation process. In some embodiments, the processing system may be further configured to periodically rotate encryption keys according to defined security policies to mitigate data exposure risks. [0349]In block 1106, the processing system may store the encrypted document in a decentralized storage system. For example, the processing system may divide encrypted document data into segments and distribute these segments across decentralized nodes selected through a consensus mechanism or based on metrics such as availability, geographic location, and latency. In some embodiments, the processing system may replicate encrypted segments across geographically diverse nodes to achieve fault tolerance or jurisdictional compliance with data residency requirements. For example, multiple encrypted copies of a document may reside on nodes distributed across jurisdictions meeting specific regulatory standards. In some embodiments, the processing system may assign unique identifiers to encrypted segments and track these identifiers using metadata entries recorded in a distributed, tamper-evident ledger. In some embodiments, the processing system may be further configured to dynamically reallocate document segments from nodes identified as compromised or unresponsive to maintain secure and uninterrupted document availability.

6	**US20240370827**	DIGITAL PROCESSING SYSTEMS AND METHODS FOR EXTERNAL EVENTS TRIGGER AUTOMATIC TEXT-BASED DOCUMENT ALTERATIONS IN COLLABORATIVE WORK SYSTEMS	MONDAY COM LTD	ZIONPOUR, RON; HARAMATI, TAL; MANN, ROY	11/07/2024

Abstract	Systems, methods, and computer-readable media for automatically altering information within an electronic document based on an externally detected occurrence are disclosed. The systems and methods may involve accessing an electronic word processing document; displaying an interface presenting at least one tool for enabling an author of the electronic word processing document to define an electronic rule triggered by an external network-based occurrence; receiving, in association with the electronic rule, a conditional instruction to edit the electronic word processing document in response to the network-based occurrence; detecting the external network-based occurrence; and in response to the detection of the external network-based occurrence, implementing the conditional instruction and thereby automatically edit the electronic word processing document.
para [0130]	FIG. 1 is a block diagram of an exemplary computing device 100 for generating a column and/or row oriented data structure repository for data consistent with some embodiments. The computing device 100 may include processing circuitry 110, such as, for example, a central processing unit (CPU). In some embodiments, the processing circuitry 110 may include, or may be a component of, a larger processing unit implemented with one or more processors. The one or more processors may be implemented with any combination of general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), field programmable gate array (FPGAs), programmable logic devices (PLDs), controllers, state machines, gated logic, discrete hardware components, dedicated hardware finite state machines, or any other suitable entities that can perform calculations or other manipulations of information. The processing circuitry such as processing circuitry 110 may be coupled via a bus 105 to a memory 120.
para [0347]-[0348]	[0347]For example, in FIG. 2, a fifth entity may access a collaborative electronic document from user device 220-2 via Network 210 and stored in repository 230-1. After accessing the collaborative electronic document, the fifth entity may see alterations made on user device 220-1 by a first entity of a first collaborative group. The fifth entity may attempt to change the alterations made by the first entity. The computing device 100 may then determine, by looking up instructions stored in its memory whether the fifth entity (user device 220-2) has permission to change the first alteration. It may be determined that the fifth entity has permission to change, and the changes may then be stored in repository 230-1. The next entity that accesses the collaborative electronic document on user device 220-m, via Network 210, may see the changes made by the fifth entity to the collaborative electronic document. [0348]In some aspects of the disclosure, it may be determined that a fifth entity possesses the permission, based on a determination that the fifth entity is associated with the first collaborative group. Determination that the fifth entity is associated with the first collaborative group may include analyzing characteristics of the fifth entity, comparing them to the characteristics of the first collaborative group, and identifying a similar or matching characteristic such that the fifth entity is related to the first collaborative group, consistent with the discussion above. For example, it may be required that the fifth entity be a part of the first collaborative group (i.e., linked to the first entity and the second entity) to possess a change-enabling permission. Alternatively, for example, it may be determined that the fifth entity may possess a change-enabling permission and may not be a part of the first collaborative group. Alternatively, some embodiments may include at least one processor further configured to recognize a fifth entity as a member of a third collaborative group with permission to change alterations of a first collaborative group, to permit the change to a first alteration, and to tag the change with a third collaborative group indicator. Recognizing the fifth entity as a member of a third collaborative group may include determining that the fifth entity may be linked to an additional collaborative group (e.g., the third collaborative group) or that it may not belong to either the first collaborative group or the second collaborative group. For example, the system may analyze characteristics of the fifth entity and recognize that it is associated with Company3, different from Company1 that is associated with a first collaborative group, and different from Company2 that is associated with a second collaborative group. The system may determine that the fifth entity is associated with a third collaborative group and that the third collaborative group has permission to change alterations made by the first collaborative group (Company1). The changes may be tagged or otherwise associated with a third collaborative group indicator (e.g. the logo of Company3). Recognizing that the third collaborative group with permission to change alterations of the first collaborative group may include determining that the third collaborative group is authorized to change the first alteration based on default or configured permission settings, consistent with the disclosure discussed previously above. For example, the third collaborative group may be permitted to change alterations of the first collaborative group but may not be permitted to change alterations of the second collaborative group. Alternatively, the third collaborative group may be permitted to change alterations of the first and second collaborative group. Consistent with the discussion above, permitting the change to the first alteration may refer to allowing or authorizing an entity to edit, add, delete, comment on, change, subtract, rearrange, correct or any other granting of authority to modify the first alteration. Tagging the change with a third collaborative group indicator may include identifying, marking, labeling, associating, classifying, or otherwise indicating the source of the change through an identifier (e.g., the third collaborative group indicator). For example, tagging a change may include associating metadata with the change that may

	be stored in a repository and not necessarily displayed. In other examples, tagging the change may include associating an indicator (e.g., graphical, alphanumeric, or a combination thereof) that may be presented in a manner to indicate additional information relating to or otherwise identifying the collaborative group.
para [0522]-[0523]	[0522]Granular permissions, as used herein, may refer to any attribute or setting that may define how an entity or entities may interact with any amount of content associated with a section or portion of a shared electronic document. Content, as used herein, may refer to any information displayed in a section or portion of a shared electronic document or any other information that may be associated with a section or portion of a shared electronic document. For example, content may include data objects, alphanumerics, metadata, or any other data associated with a section or portion of a shared electronic document. Such interactions may involve viewing, editing, navigating, executing, or any other user task involving the content associated with the document. The sections or portions of the shared electronic document may include one or more data objects. Non-limiting examples of granular permissions may involve attributes or settings that authorize an entity or entities to view or edit a single character of text, a line of text, several lines of text, a table, a portion of a video, a portion of an audio file associated with the document, or any other selectable portion of the document. Permissions may be said to be configurable on a granular level because permissions may be configured for any selected segment of information (e.g., a block, as discussed in detail later) contained in a document, and not just a general permission setting for the entire document. Other non-limiting examples of granular permissions may involve attributes or settings that authorize an entity or entities to view or edit a sentence, paragraph, an image, or a chart associated with the document. In some embodiments, the granular permission settings may be reconfigured after a particular setting has been applied. For example, a single electronic document may contain a first portion of text with a first permission setting and a second portion of text with a second permission setting. The first permission setting may be configured to enable only the document author to view and access the first portion of text while the second permission setting may be configured to enable any user to view and access the second portion of text. As a result, the document author would be able to view and access both the first and second portions of text in the document, while a secondary user would only be able to view and access the second portion of text. The document author may then reconfigure the permission setting to access the first portion of text to authorize any secondary user to view that first portion of text at a later time. Similarly, the permission settings to access the second portion of text may be reconfigured to restrict access to certain users by predetermined users such as the document author. [0523]By way of example, FIGS. 59A and 59B illustrate examples of a shared electronic document with granular permissions. Referring to FIG. 59A, the shared electronic document 5900A may be a collaborative electronic word processing document. The document 5900A may include an indication of an entity 5902A accessing the document via an editing interface 5904A. The indicator 5906A may indicate to an accessing entity, such as entity 5902A, all of the entities accessing the shared electronic document at that time, or at any other time (e.g., the indicator may display a last accessed time stamp for a particular entity). The document 5900A may further include a first section 5908A (e.g., a first block) and a second section 5910A (e.g., a second block), each of which may include information such as a single string or multiple strings of text. The first section 5908A and the second section 5910A may, by way of example, have associated granular permissions authorizing entity 5902A to view, or view and edit, the content associated with both the first and second sections.

7	**IN202521098632A**	A SYSTEM FOR PROCESSING DOCUMENT AND A METHOD THEREOF	SAGE UNIVERSITY	DR PRASHANT JAIN; BHAVYA SINGH; MANISH PAL; KARANVEER RAJPUT; DR ATI JAIN; DR RITU TANDON	10/31/2025

8	**Translation-WO2025253260**	Translation- ELECTRONIC DOCUMENT CERTIFICATION METHOD	BANCO DAVIVIENDA S A	AGUDELO RINCON, Charly; Espinel Pinzón, Rafael Esteban; Espinosa Alarcón, José Israel; FONSECA SIERRA, Armando; GALVÁN TORRES, Daniel Ricardo; HENRIQUEZ AVILEZ, Elian; LEÓN GUTIERREZ, Oscar Andrés; OSORIO GARZON, Miller; OSPINA OROZCO, Jhon Gabriel; Padilla Yañez, Yesika; QUINTERO VELANDIA, Diego Alfredo	12/11/2025
pg 2, last 3 paragraphs	This disclosure relates to a computer-implemented method for generating certification data for an electronic document, comprising executing the following steps on a computing unit: a) authenticating a user through an identity verification process, where the identity verification process includes at least a first and a second authentication; where the user accesses an application from a terminal that connects the terminal to the computing unit; b) receiving, from a user terminal, a status certification request associated with a file, where the file is stored in a database, and where the file includes at least one electronic document; c) transmitting, to a validator server, a verification request that includes the file, a timestamp, and an identification data associated with the user; and d) receiving, from the validator server, a certification data associated with the verification request, where the validator server obtains the certification data through a blockchain-based process. In one embodiment of the method described herein, in the blockchain-based process for obtaining certification data, the validator server obtains the certification data through a blockchain-based process that includes: validating whether a digital wallet associated with the user exists on a blockchain network; if the validation is negative, then a digital wallet				

	associated with the user is created. This digital wallet includes a private key, a public key, and an identification number associated with the user. A hash is then generated using the blockchain network associated with the digital wallet. associated with the verification request data file, and the certification data is obtained from the hash data. The certification data includes the hash data, a status data of the electronic document of the file, and a timestamp data associated with the moment when the hash data is generated. In some forms of the method described herein, the validator server and the blockchain network are connected via an API gateway, where the API can integrate endpoints that provide services such as authentication, digital signature, notifications, blockchain timestamping, and combinations thereof, as well as making the connection to a backend service.
Abstract	The present disclosure describes a computer-implemented method for generating certification data for an electronic document, which comprises carrying out, in a computing unit, the steps of: a) authenticating a user by means of an identity verification process, the identity verification process including carrying out at least a first authentication and a second authentication; b) receiving, from a user terminal, a status certification request associated with a file, wherein the file is stored in a database, and wherein the file includes at least the electronic document; c) transmitting, to a validating server, a verification request that includes the file and a piece of identification data associated with the user; and d) receiving, from the validating server, a piece of certification data associated with the verification request.
pg 3, para 1-2	Referring to FIG. 1, this disclosure relates to a computer-implemented method for generating certification data (12) of an electronic document, comprising executing on a computing unit (1) the steps of: a) authenticating a user (2) through an identity verification process, where the identity verification process includes executing at least a first authentication and a second authentication; where the user (2) accesses from a terminal (3) an application (4) that connects the terminal (3) to the computing unit (1); b) receiving, from a terminal (3) of the user (2), a status certification request (5) associated with a file (6), where the file (6) is stored in a database (7), and where the file (6) includes at least one electronic document; c) transmitting, to a validator server (9), a verification request (8) that includes the file (6), a timestamp data (10) and an identification data (11) associated with the user (2), and d) receiving, from the validator server (9), a certification data (12) associated with the verification request (8), wherein the validator server (9) obtains the certification data (12) through a blockchain-based process (13). The computer-based method implemented for generating certification data (12) in this disclosure offers several distinct technical advantages: User authentication, which can include multiple layers of identity verification, adds a level of security to the process. Efficient communication between the user terminal (2) and the computing unit (1) helps ensure the integrity and protection of transmitted data, thereby improving computer security across system connections. Furthermore, the integration of a blockchain-based process (13) to obtain the certification data (12) provides additional security and verification. This results in the immutability of the certified data and traceability of the certification process, which enhances the transparency and reliability of the entire system. This makes the method a secure solution for generating certification data for electronic documents.

9	**Original-WO2025253260**	Original- ELECTRONIC DOCUMENT CERTIFICATION METHOD	BANCO DAVIVIENDA S A	AGUDELO RINCON, Charly; Espinel Pinzón, Rafael Esteban; Espinosa Alarcón, José Israel; FONSECA SIERRA, Armando; GALVÁN TORRES, Daniel Ricardo; HENRIQUEZ AVILEZ, Elian; LEÓN GUTIERREZ, Oscar Andrés; OSORIO GARZON, Miller; OSPINA OROZCO, Jhon Gabriel; Padilla Yañez, Yesika; QUINTERO VELANDIA, Diego Alfredo	12/11/2025

10	**US20190236124**	SYSTEMS AND METHODS FOR CREATING A DYNAMICALLY EDITABLE DOCUMENT TEMPLATE AND FOR DRAFTING, REVIEWING, NEGOTIATING, AND FINALIZING A DYNAMICALLY EDITABLE DOCUMENT	First Cut Technologies LLC	BUMBY, COLIN; CHARNEY, JAMES; BALDWIN, Adam	08/01/2019
Abstract	Systems and methods for reviewing, drafting, and negotiating a dynamically editable document are disclosed. Exemplary embodiments may: receive a template request, generate a drafter interface generate a drafter interface, the drafter interface comprising a dynamically editable document and one or more prompts and corresponding responses corresponding to the dynamically editable document; receive responses; invoke listeners; record and distribute the responses, and record edits. An exemplary embodiment may: generate a manager interface, the manager interface comprising one or more prompt section fields, one or more segment fields, one or more prompt fields, one or more response fields, and one or more listener fields; create one or more prompt sections; create one or more segments; create one or more prompts; create one or more responses; create one or more listeners that are invoked based on receiving the one or more responses; and generate a dynamically editable document template.				
para [0110]-[0115]	[0110]An edit interface may be displayed on the drafter interface such that a party may edit the text on the draft. The edit interface may allow a drafter to make more granular changes to the document, such as modifying punctuation, a letter, a number, a word, and/or other text. A first drafter's edits may be displayed in a first style. A second drafter's edits may be displayed in a second style. The first drafter and the second drafter may be from a first party. The first drafter and the second drafter may be from opposing parties. A first style may be different than a second style. The first style may be a different size, shape, color, highlight, underline, font, and/or other difference than the second style. [0111]At operation 206, the system receives responses to the prompts. Responses may include, for example, selections of radio buttons or drop-down menu items, answers to				

	prompts, responses to fill-in-the-blank prompts and so on. The drafter (or reviewer in the case of document revision), responds to the prompts in the sequence provided. As described below, the system uses the responses to build the document (e.g., by selecting or retaining segments associated with the response) and may also alter subsequent prompts based on the response. The received responses may be stored in electronic storage. [0112]An operation 208 may include invoking listeners based on the responses. In some embodiments, one or more listeners are associated with a response. When a response is received, its associated listener(s) may be invoked and the listener acts on the document. For example, a listener may act to retain all document segments associated with a selected response and to delete all document segments associated with the responses that were not selected. In this way, the document can be built from the segments corresponding to the drafter responses. Accordingly, listener actions may include adding, removing, and/or modifying a segment in the document. Listener actions may further include hiding, revealing or altering subsequent prompts and response options. Listener actions may also include otherwise affecting prompts, responses, and/or the document. [0113]Invoking listeners may include modifying the document. As described immediately above, the document may be modified based on the listener action. This operation may also include dynamically making any grammatical changes appropriate based on the document modification, such as appropriately placing or removing an oxford comma or semicolon, renumbering a list of items, relocating a conjunction, or other grammatical/stylistic adjustments. The modified document may be presented to the drafter on the drafter interface in real time as the document is modified based on responses. [0114]An operation 210 may include recording and distributing responses. Recorded responses may be available for all users of a first party associated with the document. Distributed responses may be available for all parties and users associated with the document. An operation 212 may include recording edits. Recorded edits may be available for all users of a first party associated with the document. An operation 214 may include distributing the modified document and the edits. Distributed edits may be available for all parties and users associated with the document. The distributed document may be available for all parties and users associated with the document. [0115]This process of providing prompts, receiving drafter responses and updating the document based on the responses may be repeated until the draft document is ready for review by an opposing party. Once the draft is finished to the drafter's satisfaction, the drafter may then send the dynamically editable document to the reviewer (e.g., the opposing party), and the reviewer may revise the document. This may begin the negotiation process as document versions are passed back and forth between the drafter and the reviewer.

11	**US11418497**	Synchronization in a file storage system using authentication-state-based permissions	MICROSOFT TECHNOLOGY LICENSING, LLC	RATHORE, Jyotsana; BARRETO, JOSE A.; CHAN, KEVIN ANDREW; PEMMARAJU, Deepak Sreenivas; TURNER, ROBERT C.; DESAI, RONAKKUMAR; MOULHAUD, PATRICK	08/16/2022

Abstract	A system is provided for facilitating access to data stored in a cloud-based storage service. Data associated with a user account is stored at the cloud-based storage service. A portion of the data is associated with a heightened authentication protocol. A request for an application to receive data that is associated with the heightened authentication protocol is received at the cloud-based storage service. In response to the request, the request is authenticated based on the heightened authentication protocol. In response to authenticating the request, permission is granted for the application to receive the data that is associated with the heightened authentication protocol. In response to a locking of the data that is associated with the heightened authentication protocol, an indication that the data is unavailable is sent to the application.
col 1, ln 65- col 3, ln 20	At least some embodiments of this disclosure are directed to providing a secure area in a distributed storage system. In some embodiments, the secure area may be referred to as a vault, which may be a special folder that requires additional authorization measures such as two-step verification to access the contents or to even view the vault. This folder provides an additional layer of security to files and documents in the event that a user's device or account is compromised. In some embodiments, the additional layer of security may be provided more generally for a class of action or activity rather than to activity tied to a secure area of storage, where the vault may be one type of such activity. It should be understood that while the example implementations described herein refer to the vault, the disclosed principles may be applied to actions or activities that are tied to the additional layer of security. In some implementations, the vault may be part of a cloud-based distributed storage service, such as OneDrive, Dropbox, Google Drive, etc., that may be accessible via a user's account on a mobile device, via a web-based application, on the user's personal computer, and may be extended to other endpoints that can connect to the distributed storage service. In some embodiments, functionality associated with implementing a vault as described herein may be performed by a vault service or a vault storage service. Once authenticated, the vault may function like a typical folder or other storage container that may be accessed using a file explorer or other application that provides an interface for accessing file systems. For example, documents and files stored in the vault may be viewed or opened, and added to or deleted from the vault. In some embodiments, once access to a user's vault is authorized, access to the vault may be allowed in a time-limited manner. The vault may lock after a specified time, requiring re-authentication to continue to access the vault. While the term “lock” and “unlock” are used to represent authentication states in one example, it should be understood that other authentication state semantics may be used, such as those that have long term and short term implications. In some embodiments, the user's computing device may execute a synchronization application that coordinates the synchronization of locally-stored files with a remote copy of the files stored in the vault at the distributed storage service. A user interface may be provided to allow a user to view the vault, control the vault, and make topological changes to the vault, such as renaming files in the vault and/or moving files into the vault or removing files from the vault. The user interface may be provided by a file explorer application or as part of an application configured to open and/or edit data stored in the vault. In some embodiments, the vault storage service may provide a graphical user interface (GUI) that facilitates changes to the items stored in the vault. User inputs can be provided in the form of a request, which can be communicated by a gesture, text input, voice input, or any other suitable form of input that can define an operation on a file. For example, a user might utilize the GUI to drag-and-drop files to the GUI for upload to distributed storage. The GUI may display a visual indication or other identifier that a folder or a document is stored in the vault. Information pertaining to files in the vault may be logged in metadata associated with the files. The metadata may include information such as the directory location and the file name.

	The techniques disclosed herein provide a number of improvements over existing systems. For instance, when the file is stored at a remote storage service, such as Google Drive, iCloud, or OneDrive, a secure area may be provided that can be used to selectively store files therein rather without having to apply additional security mechanisms to the entirety of the user's files or individually to selected files. The techniques disclosed herein further improve user interaction with a computer along with providing improvements with respect to processing resources, network resources, and memory resources. For instance, a user no longer has to manually apply security measures to individual files or open a separate secure account for storing sensitive files, causing the user to have to manage multiple storage services and accounts. A more simplified way to provide a secure area, as disclosed herein, can lead to the reduction of inadvertent inputs, and based on which other efficiencies can be improved. This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter. The term “techniques,” for instance, may refer to system(s), method(s), computer-readable instructions, module(s), algorithms, hardware logic, and/or operation(s) as permitted by the context described above and throughout the document.

12	**US11449911**	Blockchain-based document registration for custom clearance	Alipay Labs (singapore) Pte. Ltd.	FANG, HUI; CAO, Shengjiao; YANG, Weitao	09/20/2022
Abstract	Disclosed herein are methods, systems, and apparatus, including computer programs encoded on computer storage media, for document registration. One of the methods includes: at a service platform, receiving first information related to an order, wherein the first information is provided, or to be provided, to a service authority for clearance of the order; at the service platform, encrypting at least a portion of the first information to generate encrypted first data; and sending the encrypted first data to a blockchain network to store the encrypted first data on a blockchain managed by the blockchain network, wherein the encrypted first data are stored on the blockchain through consensus of blockchain nodes of the blockchain network.				

13	**US20250173446**	SYSTEMS AND METHODS FOR ELECTRONIC DOCUMENT EXECUTION, AUTHENTICATION, AND FORENSIC REVIEW	Entrust & Title (FZE)	HADI, SHAMSH	05/29/2025
Abstract	Embodiments disclosed herein include systems and methods for generating and editing digital documents, and applying electronic signatures to the documents. The system further performs forensic operations on digital documents for determining the authenticity of the document data or metadata and validating the status of document. The forensic operations may include, for example, security operations, audit trail analysis, and authenticity/verification analysis, which the system performs on an input document being scrutinized for authenticity and a master digital document purportedly corresponding to the input digital document. The system may apply a machine-learning architecture to identify discrepancies otherwise imperceptible to humans or to speed the review process. The system may generate a report for the user who submitted the forensics request indicating the results of the forensic operations. The available forensics operations and level of information about the master digital document are tailored to permissions afforded to the submitting-user.				

14	**US11276039**	Role-agnostic interaction management and workflow sequence generation	Venkatraman	VENKATRAMAN, Bhaskar Mannargudi; Chunduru, Pravin Sai; Raju, Sivarama Subramaniam	03/15/2022
Abstract	A method and a role-agnostic interaction management system (IMS) dynamically embedded into an editing application define codified decision rules (CDRs) associated with actions and inductively sequence a workflow(s) in a workflow-embedded content entity. The IMS automatically generates a satellite content entity (SCE) carrying an inductive action condition (IAC) for each allocation of a master content entity (MCE) to one or more participants for inducing the workflow sequence(s). Using the CDRs, the IMS dynamically generates a permission set for each SCE, and comments and regions within the SCE. The IMS contextually anonymizes the participants invoked to act on each SCE based on an anonymity requirement. The IMS tracks and maps actions performed along an interaction path of each SCE, while transmitting and executing the permission set, until each SCE reaches an intended closure and the comments reach a resolution to generate one or more re-deployable workflow sequences from the MCE.				

Search History

Patbase
Date Searched: 2/2/2026

1 CPC=(G06F16/1774 or G06F16/1837 or H04L9/0367 or G06Q40/03) 18,884
2 UC= 707/827 927
3 IC=(H04L9/06 or G06F21/60 or G06Q40/03) 120,792
4 (1 or 2 or 3) 133,668
5 4 and document* 24,365
6 5 and (state* w2 machine*) 1,805
7 6 and (state* w2 transition*) 222
8 7 and (document* w5 (chang* or edit* or version* or finaliz*)) 32
9 8 and (cryptograph* w5 key*) 19
10 9 and authoriz* 18
11 10 and (("without" or "no") w3 fork*) 1
12 4 and (("without" or "no") w3 fork*) 53
13 12 and ((state* w2 transition*) w5 document*) 0
14 12 and (document* w3 histor*) 1
15 7 and ((state* w2 transition*) w5 document*) 3

1 (document* w4 unmutab*) 1
2 document* w5 (chang* or edit* or version*) w5 lock* 303
3 2 and (state* w2 machine*) 26
4 3 and unedit* 2
5 3 and (freez* or unfreez*) 2
6 3 and histor* 11
7 6 and finaliz* 4
8 document* w5 (chang* or edit* or version*) 77,075
9 8 and (state* w2 machine*) 3,424
10 9 and (document* w5 (chang* or edit* or version* or finaliz*)) 3,424
11 10 and (document* w4 (lock* or finaliz*)) 94
12 11 and (state* w4 transition*) 25
13 8 and (document* w3 histor*) 2,716
14 13 and (state* w2 machine*) 156
15 14 and (document* w2 seal*) 1
16 (state* w2 transition*) and document* 37,992
17 16 and (state w2 machine*) 9,577
18 17 and authoriz* 2,915
19 18 and (("without" or "no") w3 fork*) 9
20 18 and authority 1,182
21 19 and (cryptograph* w5 key*) 4
22 (document* w4 immutab*) 366
23 22 and (state* w2 machine*) 38
24 23 and (document* w5 (chang* or edit* or version* or finaliz*)) 18
25 24 and (state* w4 transition*) 6

Google Patents
Date Searched 2/2/2026

document state change edit
More than 100,000 results

document state change edit lock
More than 100,000 results

document state change edit lock "state machine"
About 91,669 results

document state change edit lock "state machine" authorize
About 42,971 results

document state change edit lock "state machine" authorize version
About 31,128 results

document state change edit lock "state machine" authorize version (temporal OR time)
About 31,036 results

Google Scholar
Date Searched: 2/2/2026

document state change edit
About 916,000 results (0.20 sec)

document state change edit lock
About 93,000 results (0.23 sec)

document state change edit lock "state machine"
About 18,600 results (0.44 sec)

document state change edit lock "state machine" authorize
About 16,600 results (0.56 sec)

document state change edit lock "state machine" authorize version
About 16,200 results (0.60 sec)

document state change edit lock "state machine" authorize version (temporal OR time)
About 15,800 results (0.61 sec)

document state change edit lock "state machine" authorize version (temporal OR time) lifecycle
About 4,830 results (0.63 sec)

Google
Date Searched: 2/2/2026

document state change edit
About 1,180,000,000 results (0.33s)

document state change edit lock
About 149,000,000 results (0.33s)

document state change edit lock "state machine"
About 455,000 results (0.39s)

document state change edit lock "state machine" authorize
About 275,000 results (0.28s)

document state change edit lock "state machine" authorize version
About 278,000 results (0.28s)

document state change edit lock "state machine" authorize version (temporal OR time)
About 330,000 results (0.52s)

document state change edit lock "state machine" authorize version (temporal OR time) lifecycle
About 21,500 results (0.31s)

ProQuest
Date Searched: 2/2/2026

Databases: ABI/INFORM® Professional Advanced, Abstracts in New Technology & Engineering, AdisInsight: Drugs, AdisInsight: Trials, Adis Pharmacoeconomics & Outcomes News, Aerospace Database, AGRICOLA, AGRIS, Allied & Complementary Medicine™, Aluminium Industry Abstracts, Analytical Abstracts, Animal Behavior Abstracts, APA PsycInfo®, British Library Inside Conferences, British Nursing Index, Business & Industry, Business Monitor International, Ceramic Abstracts, Chemical Business Newsbase, Chemical Engineering & Biotechnology Abstracts, Chemical Safety Newsbase, Civil Engineering Abstracts, ClinicalTrials.gov, Computer and Information Systems Abstracts, Copper Technical Reference Library, Corrosion Abstracts, COVID-19 Research, DH-DATA: Health Administration, Medical Toxicology & Environmental Health, DIOGENES® FDA Regulatory Updates, Drug Information Fulltext, EconLit, Economist Intelligence Unit, Ei Compendex®, Electronics & Communications Abstracts, Embase®, Embase Preprints, EMCare®, Emerging Markets Direct, Energy Science and Technology, Engineered Materials Abstracts, Entomology Abstracts, ESPICOM Pharmaceutical & Medical Device News, FDAnews, FLUIDEX (Fluid Engineering Abstracts), Gale Group Computer Database™, Gale Group Health Periodicals Database, Gale Group New Product Announcements / Plus®, Gale Group Newsletter Database™, Gale Group PharmaBiomed Business Journals, Gale Group PROMT®, Gale Group Trade & Industry Database™, Global Health, Health Research Full Text Professional, HSELINE: Health and Safety, ICONDA - International Construction Database, IMS Company Profiles, IMS New Product Focus, IMS Pharma Trademarks, IMS R&D Focus, IMS R&D Focus Drug News, Inspec®, Jane's Defense & Aerospace News, King's Fund, KOSMET: Cosmetic Science, Lancet Titles, Materials Business File, Mechanical & Transportation Engineering Abstracts, MEDLINE®, METADEX, Morressier Life Science Conference Abstracts and Posters, New England Journal of Medicine, Northern Light Life Sciences Conference Abstracts, Paperbase, PAPERCHEM, ProQuest Advanced Tech & Aerospace Professional, ProQuest Biological & Health Science Professional, ProQuest Dissertations and Theses Professional, ProQuest Dissertations and Theses Professional, ProQuest Environmental Science Professional, ProQuest Materials Research Professional, ProQuest Newsstand Professional, ProQuest Research, ProQuest Technology Research Professional, Publicly Available Content, Registry of Toxic Effects of Chemical Substances (RTECS®), Solid State and Superconductivity Abstracts, TableBase, ToxFile®, Transport Research International Documentation, UBM Computer Full Text, Weldasearch®

S1 document* n/5 edit* 283527*
S2 1 and (state n/2 machine*) 292749*
S3 2 and (time or temporal*) 213712770*
S4 s1 and (state n/2 machine*) 2499°
S5 s4 and (time or temporal*) 2474°
S6 s5 and (freez* or lock*) 1545°
S7 s6 and ((freez* or lock*) n/4 document*) 26°
S8 s6 and lifecycle* 161°
S9 s8 and (unmut* or immut*) 34°
S10 s9 and (version* or histor*) 34°

Disclaimer

Cardinal Intellectual Property, Inc.'s search methodology includes steps and protocols aimed at a good faith effort to produce a reasonably complete and accurate search report, but such methodology is not without limitation. Searching, by its nature, relies at least in part on the subjective assessment of individual searchers. The databases on which Cardinal Intellectual Property, Inc. relies are maintained by third parties such as the United States Patent and Trademark Office, foreign governments, and commercial or public entities, for example, libraries or Internet database providers including associated cataloging techniques. Cardinal Intellectual Property, Inc. is not responsible for maintaining the accuracy or inclusiveness of these databases, which may be incomplete or contain errors. Accordingly, we cannot guarantee that a search is free of any error or omission, and thus, WE DISCLAIM ALL WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. This search is valid only for the mutually agreed upon scope of the search, and even minor changes in the basis or scope of the search can require a separate and distinct search. Any liability on the part of Cardinal Intellectual Property, Inc. arising out of the preparation of this search report is specifically limited to a refund of the search fee paid.

Cardinal Intellectual Property, Inc. is not a law firm and this search report is not a legal opinion. Any ranking of cited references, including ranking based on perceived relevance, any cited relevance of a reference, or any analysis provided as part of the search report is for the clientâ€™s convenience in reviewing the search results and is not intended to convey an opinion regarding the legal significance of any cited reference. Similarly, legal status, if provided, is an automated process from a third party database and is not created by, curated by, or validated by Cardinal Intellectual Property, Inc., and Cardinal Intellectual Property, Inc. is not responsible for its accuracy or inclusiveness.

Receipt or use of this search report acts as an acknowledgement of reasonable notice of the above terms, conditions, and limitations. Any questions regarding acceptance of the above terms, conditions, and limitations should be directed to Cardinal Intellectual Property, Inc. in writing.