(12) STANDARD PATENT APPLICATION (11) Application No. **AU 2024203136 A1**
(19) AUSTRALIAN PATENT OFFICE

(54) Title
Decentralized system for identification, authentication, data encryption, cloud and distributed cluster computing

(51) International Patent Classification(s)
G06Q 50/26 (2012.01)
G06F 9/50 (2006.01)
G06F 21/32 (2013.01)
G06F 21/41 (2013.01)
G06Q 10/00 (2023.01)
G06Q 40/04 (2012.01)
H04L 9/08 (2006.01)
H04L 9/32 (2006.01)
G06N 3/0464 (2023.01)

(21) Application No: **2024203136** (22) Date of Filing: **2024.05.12**

(30) Priority Data

(31) Number	(32) Date	(33) Country
2023901444	**2023.05.12**	**AU**

(43) Publication Date: **2024.11.28**
(43) Publication Journal Date: **2024.11.28**

(71) Applicant(s)
DET-IO PTY LIMITED

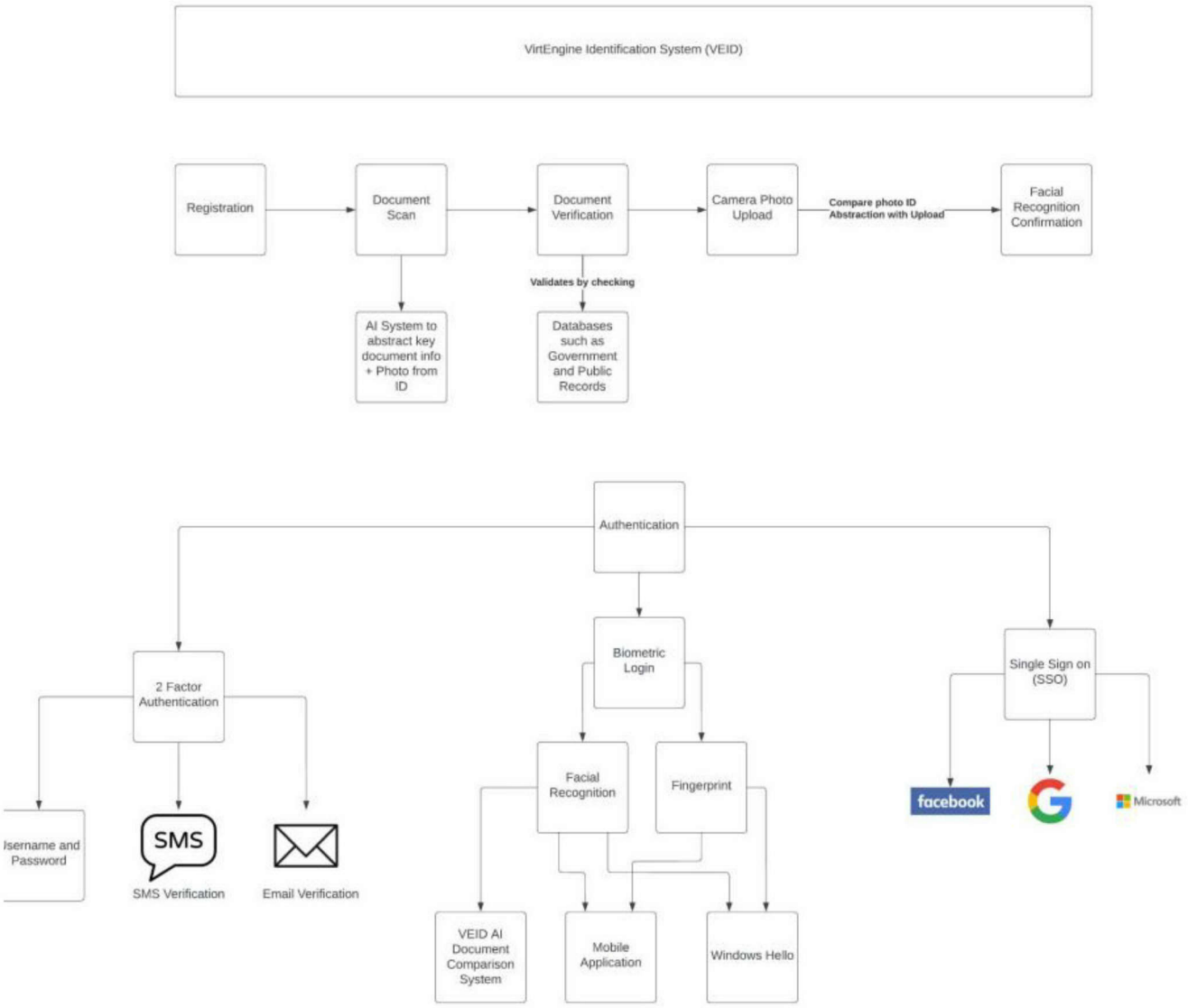
(72) Inventor(s)
Philipos, Jonathan;Livenson, Ilja;Poorun, Rohan

(74) Agent / Attorney
DET-IO PTY LIMITED, 2 Gipsy Street, Bungarribee, NSW, 2767, AU

Abstract

Decentralized system for identification, authentication, data encryption, cloud and distributed cluster computing is disclosed within this document referred to as 'VirtEngine'. The system provides a secure and verifiable way to establish and verify the identity of individuals and entities within a blockchain network as well as protect and encrypt sensitive data. These systems can be particularly useful for applications such as online marketplaces, where it is important to ensure the identity of participants and protect sensitive data. VirtEngine employs a mobile app that enables users to verify their identity and authenticate through document uploads, biometric sensors, facial recognition, and web-based scopes such as single sign-on, email verification, and SMS verification. The system also uses data encryption to ensure sensitive data is not stored on the public ledger and can only be accessed by authorized user accounts. VirtEngine also powers a Distributed Cluster Computing network and its own Cloud Marketplace system, allowing compute providers to convert computing power into tradeable currencies while allowing consumers to purchase computing power via Cloud Services, High Performance Compute, and other integrations.

Figure 2A:



Decentralized system for identification, authentication, data encryption, cloud and distributed computing

TECHNICAL FIELD

The following implementation generally relates to decentralized computing, distributed computing, cloud computing, identification utilizing artificial intelligence technologies and methods for integrating such technologies within an enclosed decentralized system.

BACKGROUND OF THE INVENTION

The following references and descriptions of prior proposals or products that follow are not intended to be, and should not be understood as, statements or admissions of commonly known information in the field. Specifically, the discussion of prior art that follows does not relate to what is commonly known or understood by experts in the field, but rather helps to clarify the innovative aspect of the present invention, which includes identifying relevant prior art proposals as one aspect.

AI-based Identification:

- Facial recognition: AI systems can be trained to recognize and identify individuals based on their facial features. This can be done through the use of machine learning algorithms that analyse images and compare them to a database of known individuals.
- Biometric identification: AI systems can be used to identify individuals based on their unique physical characteristics, such as fingerprints, iris scans, or facial recognition data.
- Document Validity: AI Systems can be used to identify and validate document authenticity such as ID Cards, Passports, and other legal identity documents.

Blockchain:

- The use of distributed ledger technology to create a decentralized and tamper-proof record of transactions.
- The use of cryptographic techniques to secure the data on the ledger and ensure the integrity of the transaction record.
- The use of consensus algorithms to validate and add new transactions to the ledger.

Cloud Computing:

- The use of virtualized resources and services to provide scalable and flexible computing capabilities over the internet.
- The use of resource pooling and allocation mechanisms to enable users to access and pay for only the computing resources they need.

- The use of software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) model to enable users to access different levels of computing resources and capabilities.

Distributed Computing:

- The use of distributed systems to enable multiple computers to work together on a common task.
- The use of message passing and communication protocols to enable the exchange of data and coordination between the computers.
- The use of algorithms and techniques for fault tolerance, load balancing, and resource allocation to enable the system to handle failures and maintain performance.

Supercomputers:

- Supercomputers are typically designed with specialized hardware and software architectures that allow them to perform their tasks at high speeds. This can include specialized microprocessors, memory systems, and interconnects.
- Supercomputers often use parallel computing techniques to distribute tasks across multiple processors. This can include techniques such as message passing, shared memory, and distributed memory.
- Supercomputers often require specialized data management systems to handle the large amounts of data that they generate. This can include distributed file systems, data warehouses, and data lakes.
- Supercomputers are often used for high performance computing (HPC) applications, which require the processing of large amounts of data in real time. HPC applications can include simulations, data analysis, and other tasks that require fast processing speeds.

Current Limitations with Background Techniques used in these technologies:

(1) Traditional Cloud Computing relies on Centralized Entities to build and manage their Cloud Computing services, resulting in a Single Point of failure and risk of data loss/downtime.
(2) Decentralized Systems do not provide a method for Identifying and Verifying users for KYC (Know Your Customer) regulations required by most government agencies when dealing with finance.
(3) Decentralized Computing through Blockchain does not offer access to HPC, instead relies on users to implement custom code such as Smart Contracts to access compute power offered by Blockchains.
(4) Distributed Computing is dependent on centralized entities to manage and deploy the network.
(5) Supercomputers rely on Centralized entities to deploy and manage infrastructure.
(6) Distributed Computing networks are limited by network interconnection between nodes thus making them unsuitable for certain real-time use cases.

202420313 6 11 Jul 2024

(7) Supercomputers are limited by physical infrastructure constraints, however, provide higher network interconnection between compute nodes compared to Distributed Computing.

(8) There are no publicly commercially available Distributed Computing systems, Folding@Home is an example of an existing Distributed Computing system however access is limited to specific Scientists and Research Organizations within the Protein Folding space.

(9) There are no publicly commercially available Supercomputers that can be rented by anyone, the only available systems are reserved for scientists and researchers such as Summit, Sierra and Frontera supercomputers.

(10) Supercomputers require tremendous CAPEX funds to implement and deploy.

(11) Cloud Computing can provide a limited alternative to supercomputer capabilities, is limited by the amount of hardware that can be deployed and dedicated for HPC tasks.

(12) Blockchain Proof of Work networks such as Bitcoin are inherently extremely wasteful on computing capacity.

(13) Cloud Computing services can be underutilized, thus leading to wasted computing capacity due to the requirements of providers needing additional capacity implemented to deal with surges in demand.

(14) Consumer compute devices such as mobile and PCs computing capacities are often underutilized as devices are not always actively used to their full computing extent.

(15) Blockchain is a decentralized, distributed ledger technology that allows multiple parties to securely record and verify transactions without the need for a central authority. It uses a distributed database that consists of a series of blocks, each containing a list of transactions. These transactions are verified and added to the blockchain through a process known as consensus, which involves multiple nodes (computers) on the network agreeing on the validity of the transaction. While blockchain can be used to process and store data, it is not a type of decentralized computing in the same way that distributed computing or supercomputing are. Distributed computing involves using multiple computers to work on a single task, whereas existing blockchain technologies use a decentralized network of computers to record and verify transactions.

2024203136 11 Jul 2024

SUMMARY OF THE INVENTION

VirtEngine is a decentralized system for identification, authentication, data encryption, distributed computing and cloud computing.

The VirtEngine **(100)** identification system utilizes the Cosmos SDK blockchain platform and TensorFlow machine learning algorithms to score users on a scale from 0 to 100, where 0 is an unknown identity and 100 is a completely verified user.

VirtEngine **(100)** employs a mobile app that enables users to verify their identity and authenticate through document uploads, biometric sensors, facial recognition, and web-based scopes such as single sign-on, email verification, and SMS verification. The mobile app uses the camera to capture images of identity documents and the user's face for biometric and facial recognition. The system then utilizes multiple types of Neural Networks utilizing Tensorflow in order to implement Facial Recognition, Comparisons, and Analysis of data collected to verify the identity of users. This allows VirtEngine **(100)** to recognize a user's identity through an automated, artificially intelligent algorithm.

For authentication, VirtEngine **(100)** offers multiple options including ledger accounts and non-custodial key management, which can be linked to password-less authentication systems such as Google, Facebook, and Microsoft single sign-on. Creating new accounts generates a mnemonic seed, which can be used to login to various wallets and manage their account. Genesis accounts have permissions to access the admin portal and nominate service providers, support staff, and other administrators, as well as disable standard user accounts to prevent fraud and manage the platform.

To ensure sensitive data is not stored on the public ledger, VirtEngine **(100)** uses data encryption based on third party public keys. Only the intended recipients can decrypt the information using their private keys. Sensitive data includes order information, ID documents, support requests, resource details, account settings, and owned organizations. Only authorized user accounts can access this data, and it is encrypted in transit and at rest.

VirtEngine **(100)** also includes a decentralized Cloud Marketplace **(103)** that allows users to rent out their computing resources to the network. The marketplace integrates the SLURM workload manager and deploys it across nodes within Kubernetes clusters. As the marketplace grows, more resources will be available for the decentralized SLURM clusters. Blockchain computing allows for the running of parallel codebases across multiple independent nodes, enabling automated deployment of SLURM workload nodes and adding them to existing clusters to build the largest global decentralized SLURM compute marketplace.

A Golang-based module with support for the SLURM workload manager is used to manage the deployment of workloads across the decentralized nodes. The module includes a library of pre-configured SLURM workloads that can be deployed on demand, as well as support for custom workloads. The module also includes a billing system that tracks the use of

202420313 6 11 Jul 2024

resources and generates invoices for the users who have rented out their computing resources.

VirtEngine **(100)** provides a secure and verifiable way to establish and verify the identity of individuals and entities within a blockchain network. It offers a range of options for identification and authentication, including document uploads, biometric sensors, facial recognition, and web-based scopes. It also uses data encryption to ensure sensitive data is not stored on the public ledger and can only be accessed by authorized user accounts. The decentralized cloud marketplace **(103)** allows users to rent out their computing resources and deploy workloads across a decentralized network of nodes.

Traditional Cloud Computing relies on a Centralized Model of computing, there is a large CAPEX fund expenditure required to build and deploy Data-center Facilities to store and compute data. In today's world, consumer computing devices are quite common and in high supply. Many consumer computing devices utilize an exceedingly small portion of their CPU/Memory/Storage capacity which could be repurposed to deliver Distributed Computing. By repurposing common consumer computing electronics to power a Decentralized Computing cluster – users can achieve an efficient model to utilize High Volume Distributed Computing.

A Decentralized and distributed cloud computing model also provides benefits such as a shared economy, minimal latency use-cases, and no single point of failure by distributing Network, Computing, and Storage services across a decentralized platform.

Advantageous Effects of Invention

1. Removes dependence of human interaction on user sign up process

2. Includes redundancies providing a robust platform for infrastructure services

3. Copes with power supply failures.

4. Copes with network supply failures.

5. Copes with infrastructure supply failures.

6. Provides an infinitely scalable infrastructure platform for compute and physical assets.

7. Provides an identification platform for sensitive transactions.

8. Provides a robust encryption methodology to secure communications

9. Provides a robust identification methodology to authorize transactions

10. Provides users with access to financial institutions such as currency exchange and holdings

2024203136 11 Jul 2024

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1: The overall architecture of VirtEngine **(100)** Protocol, includes how systems interact with each other through API's and Service Mesh.

Figure 2: The overall architecture of VirtEngine Identification System

Figure 3: Includes a screenshot of how the Dashboard looks like.

Figure 4: Includes a screenshot of how Public Offerings overview can be added.

Figure 5: Includes a screenshot of how Public Offerings descriptions can be added.

Figure 6: Includes a screenshot of how Public Offerings management can be added.

Figure 7: Includes a screenshot of how Public Offerings accounting components can be added.

Figure 8: Includes a screenshot of how Public Offerings accounting plans can be added.

Figure 9: Includes a screenshot of the Offering Review.

Figure 10: Includes a screenshot of an offering being edited.

Figure 11: Includes a screenshot of the Global Cloud Marketplace.

Figure 12: Includes a screenshot of an example Offering that has been added to the marketplace.

Figure 13: Includes a screenshot of an offering being purchased.

Figure 14: Includes a screenshot of a deployed Resource.

Figure 15: Includes a screenshot of the Order Details of a resource.

Figure 16: Includes a screenshot of a mobile application designed for VirtEngine Identification.

Figure 17: Includes a screenshot of Multi-Factor Authentication in action.

Figure 18: Includes diagrams of the Waldur Architecture.

Figure 19: Includes diagrams of the Waldur SLURM Diagrams.

DETAILED DESCRIPTION OF EMBODIMENTS

Glossary:

VirtEngine, VE, 100: Refers to the complete VirtEngine **(100)** application.

VirtEngine Blockchain, **VE Blockchain, 101**: Refers to the custom built blockchain that contains the VirtEngine **(100)** system.

Cosmos SDK, **Cosmos, 102**: Refers to the open source blockchain framework known as Cosmos SDK (**102**)

VirtEngine Cloud Marketplace, VE Cloud Marketplace, VECMP, 103: Refers to the VirtEngine Cloud Marketplace module that integrates with the VirtEngine Waldur system.

VirtEngine Identification, VirtEngine ID, VE Identification, VEID, 105: Refers to the VirtEngine Identification system handling User Sign ups, Identification, Verification, and Authentication.

VirtEngine Supercomputer, VE Supercomputer, VESC, 106: Refers to the VirtEngine Supercomputer system that utilizes Distributed Computing to enable High Performance Compute.

VirtEngine Waldur, VE Waldur, Waldur, 107: Refers to the Open Source Waldur Cloud Marketplace system.

Tensorflow, 118: Refers to the open-source machine Learning platform built by Google called Tensorflow.

OpenStack, 127: Refers to the open-source Cloud Management Platform known as OpenStack.

Kubernetes, K8s, 128: Refers to the open-source Container Orchestration, Management and

SLURM, 135, 136: Refers to the open-source High Performance Compute platform known as SLURM (Simple Linux Utility for Resource Management).

The chosen tools to build VirtEngine **(100)** are not a necessity, they are interchangeable and have alternative systems that can be used in place. The tools that aid in building the VirtEngine **(100)** system are as follows: Python, Golang, Cosmos SDK (102), Waldur, Kubernetes, SLURM, OpenStack, Tensorflow. Alternative systems that can be used instead are as follows:

Python/Golang – Any high-level programming language such as C#, NodeJS, C++, Java, and many more.

Cosmos SDK (102) – Any blockchain framework that supports Proof-Of-Stake such as Substrate (Polkadot), EOS, and various open-source blockchains. A blockchain could also be built from scratch.

Waldur, 107 – Building a similar Cloud Marketplace system as well as it's User Interface or implementing the Cloud Marketplace system within the blockchain modules and integrating with systems such as OpenStack directly.

Kubernetes, 128 – Alternatives include simple Docker images or Infrastructure management systems such as Chef and Ansible.

SLURM, 135, 136 – Alternatives include MOAB HPC Suite, The Portable Batch System, Open Grid Scheduler, Kubernetes and other systems that support scheduling computing tasks.

OpenStack, 127 – Alternatives include VMWare, OpenNebula, and CloudStack and any system that supports virtualization such as KVM, OpenVZ, Xen, Hyper-V etc.

Tensorflow – Alternatives include Machine Learning algorithms that support MTCNN and other Neural Networks such as PyTorch, Keras, Caffe, MXNet, and Darknet.

The **VirtEngine (100) system** described includes the following components:

- **VirtEngine Blockchain 101**, powered by Cosmos SDK **(102)**. Provides a proof-of-stake and blockchain base for customization and development of necessary modules.
- **VirtEngine cloud marketplace, 103**, powered by Waldur **(107),** which allows users to purchase cloud services from providers.
- **VirtEngine Identification system, 105**, powered by TensorFlow ML, which allows users to verify their identities in a decentralized system to provide trust and validation when using the marketplace.
- **VirtEngine tokens, 104**, which can be used to purchase services on the marketplace or exchanged for fiat currency through Cryptocurrency Exchanges.
- **Waldur, 107**, a hybrid cloud management system that supports a variety of cloud computing integrations and has a modular design.
- **VirtEngine supercomputer, 106**, a decentralized network of nodes that utilizes the SLURM workload manager to operate as a distributed supercomputer.
- **Authentication** options include ledger accounts and non-custodial key management, which can be linked to password-less authentication systems. Authorization methods include MultiFactor Authentication to achieve additional account security.
- **Data encryption** is based on third-party public keys, used to secure sensitive data such as order information, ID documents, support requests, and more.

VirtEngine Identification System

The VirtEngine ID system is a decentralized identity solution that utilizes blockchain technology to securely store and verify identity information. It allows users to create, manage, and control their own digital identity, which can be used to access various online services and resources. The VirtEngine ID system is built on top of a decentralized, open-

2024203136 11 Jul 2024

source blockchain platform, which ensures that the identity information is stored in a secure and transparent manner.

The VirtEngine ID system consists of several components, including:

1. Identity Wallet: This is a digital wallet that allows users to store and manage their digital identity information, such as personal details, documents, and credentials. The wallet is secured by a private key, which only the user has access to.
2. Identity Verification: This is a process that verifies the authenticity and accuracy of the identity information provided by the user. This can be done through various methods, such as facial recognition, document verification, and biometric authentication.
3. Identity Services: These are online services and resources that use the VirtEngine ID system to verify and authenticate the identity of users. These services can include online banks, e-commerce platforms, and government agencies.
4. Identity Network: This is the decentralized network of nodes that stores and processes the identity information on the blockchain. The network is secured by advanced cryptographic techniques and consensus algorithms, which ensure the integrity and security of the data.
5. Identity Mobile Application: This is a mobile application within a hardware device that captures and verifies the identity of the user through facial recognition and other biometric authentication methods. The mobile app can be integrated with the Identity Wallet to provide an additional layer of security and method to upload and provide identity documentation and data to the system.

The VEID System works by utilizing TensorFlow machine learning algorithms to identify users via various scopes. These scopes include Identity Documents, Biometric Data, Facial Data, Authorized Online Accounts (Single Sign On), Domain Verification, and Email Verification, among other scopes.

A module will be built in Golang for the VEID system, to be integrated into the Cosmos SDK (102) base to be implemented within a decentralized Blockchain. Cosmos SDK (102) Supports Golang codebases, TensorFlow-GO is a Golang Application Programming Interface that will be used to connect TensorFlow with the Cosmos SDK (102) application.

VEID works by scoring users between 0 to 100, 0 being an unknown identity and 100 being a completely verified user. Users will be able to verify their identity with a mobile app utilizing Camera for Document Uploads, Biometric sensors (fingerprint), Camera for Facial Recognition, along web scopes such as Single Sign On (Google, Facebook, Microsoft, etc.), Email Verification, SMS Verification, 2FA Authentication support. Users registering to become providers will need to verify their domain to join the VirtEngine Cloud Marketplace (**103)**.

VEID encodes generated keys by adding salt that are specific to each upload within the metadata of the image/video file, thus when uploaded to the blockchain it can be verified that it indeed gets processed and uploaded directly through the mobile's app camera integration. Preventing users from uploading verification data from saved galleries.

2024203136 11 Jul 2024

VEID also collects the following list of parameters for the AI to process and make better scoring decisions:

1. IP Address: The IP address of the user's device can be used to determine the user's location and to identify patterns of suspicious activity.

2. Device information: Information about the user's device, such as the make, model, and operating system.

3. Location data: The location of the user's device using GPS can be used to identify patterns of suspicious activity and to verify that the user is who they claim to be.

4. Demographic data: Information about the user's age, gender, and other demographic characteristics may be collected and analysed to identify patterns of suspicious activity.

5. Single Sign On metadata, metadata from accounts such as Google, Facebook or Microsoft can assist in identifying customers.

6. Social Media data: VEID can collect and analyse social media profiles via third party services.

7. Government Agencies: VEID can connect and process data from specific government agencies to further secure the system from fraud in select countries with digital systems.

The VirtEngine Identification System will be an extension towards the Authentication module inside Cosmos SDK (102), the authentication module will be extended to support the verification of customer identities utilizing direct API integrations with Tensorflow.

For Tensorflow to verify the identity of its users, it will need to process encrypted data uploaded to the blockchain. Once a user uploads their ID Scopes (in an encrypted manner) to the blockchain, the Identity Verification system running on the Identity Network will decrypt the data via the Identity Network's chosen node's private keys to complete the verification.

1. Detect faces on images.
2. Compute faces' descriptors/embeddings.
3. Compare descriptors.

A Tensorflow algorithm which utilizes the Multi-task Cascaded Convolution Network (MTCNN) can be used for facial recognition and allows comparison of facial data between Uploaded Identification Documents & Uploaded Facial Recognition information.

Other types of Neural Networks that can be implemented in Tensorflow and be used to assist Identification of consumers includes convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and generative adversarial networks (GANs).

Convolutional Neural Networks (CNNs) are often used for image classification tasks and can be trained to recognize specific features in images, such as facial features or text.

2024203136 11 Jul 2024

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network that can process sequential data, such as text or time series data. They can be used to analyze patterns in user behavior or language to identify fraudulent activity.

Generative Adversarial Networks (GANs) are a type of neural network that consists of two networks: a generator and a discriminator. The generator creates synthetic samples, while the discriminator determines whether the samples are real or fake. GANs can be used to generate synthetic images or text that can be used to train other neural networks.

Optical Character Recognition can be used to convert text inside documents to machine-readable information, this data can be analyzed by the other neural networks to decide on the users score. To ensure that the optical character recognition system is not bypassed with faulty data, the system verifies the signatures of the uploaded files to ensure that they have been processed by an approved client and contain both the approved client (interface) signature and the user's signature. The list of approved clients can be controlled in the blockchain's configuration and ensures that only pre-approved interfaces may interact with the blockchain and pass on identity documents during the sign-up process.

Barcode Scanning technology can be used to revalidate the data written on Identification Documents such as Driver's Licenses and Photocards.

Other types of neural networks that may be used for identity verification include autoencoders, which can be used to identify anomalies in data, and self-organizing maps, which can be used to cluster data and identify patterns.

Utilizing a combination of such Networks built on Tensorflow can provide the required automated identification of users on the VirtEngine network, by collecting Meta Data from the user's device (such as IPv4, GPS Data, Device Information, etc.) and Email Verification, SMS Verification and Single-Sign-On Systems such as Google, Facebook, and Microsoft. By combining these scopes, we can recognize a user's identity through an automated Artificially Intelligent algorithm to achieve Decentralized Identification within a blockchain system.

Combining multiple neural networks together can be done through a process known as ensemble learning, where multiple neural network models are trained and their predictions are combined in some way to make a final prediction. This can improve the performance of the overall model by leveraging the strengths of different types of models.

VirtEngine Identification system can be integrated into the Cosmos SDK (**102**) within the authentication module by leveraging the existing framework for handling accounts and account keys, defining a new module to handle the specific functions of the VirtEngine Identification system, and implementing the necessary handler and query functions. This will allow software engineers to easily incorporate the VirtEngine Identification system into their Cosmos SDK (**102**) applications and build secure, decentralized identity management systems.

The module should implement the InitGenesis and ExportGenesis functions to initialize and persist the state of the VirtEngine Identification system within the Cosmos SDK (**102**). This

will involve storing the necessary data, such as encrypted identity documents and verified identities, in the module's state.

The module should also define a set of handler functions to handle the various message types defined in the module. These functions should be responsible for updating the state of the VirtEngine Identification system and performing any necessary validations or checks.

The module should define a set of query functions to allow users to retrieve data from the VirtEngine Identification system, such as identity verification status or encrypted identity documents. These functions should utilize the Cosmos SDK (102)'s query interface to allow for easy integration with other modules and applications.

The VirtEngine Identification module in Cosmos SDK (**102**) can be integrated with Tensorflow AI by utilizing the Tensorflow library in the module's codebase. This can be done by importing the Tensorflow library and utilizing its functions to process the data from the VirtEngine Identification system. For example, the module can use Tensorflow to analyze and classify the data from the identification system, such as by using Tensorflow's machine learning capabilities to identify patterns and trends in the data. The module can also use Tensorflow to perform tasks such as data cleansing, data transformation, and data preprocessing, all of which can be essential for ensuring the accuracy and reliability of the VirtEngine Identification system. Additionally, the module can use Tensorflow to perform real-time analysis of the data, allowing it to provide real-time and automated verification to users of the VirtEngine Identification system.

The CRAFT (Character Region Awareness for Text Detection) method is a state-of-the-art technique for text detection in images. It focuses on detecting individual characters and then grouping them to form words or text regions. The craft-text-detector library is an implementation of the CRAFT method, which can be used to identify regions of interest in an image before applying Optical Character Recognition (OCR) for text extraction. This has been tested and has worked successfully.

Example Algorithim Flow for Identification System

Face Detection: Utilising advanced facial recognition algorithms, the system first scans the input image at varying rotation angles. The objective here is twofold: to detect the presence of a face, indicative of the orientation of the ID, and to record the rotation angle at which this face was detected. This probabilistic approach ensures that the ID is oriented correctly regardless of its initial position.

Image Rotation: Post face detection, the system identifies the rotation angle where the probability of a face being detected is maximised. Using this optimal angle, the original image undergoes a rotation transformation. This ensures that any subsequent processing occurs on an image that is oriented in a standardised fashion.

Perspective Transformation & Cropping: With the image now appropriately oriented, it is paramount to adjust for any inherent skew or tilt. A perspective transformation is applied, ensuring the ID's text is perfectly horizontal, which is a prerequisite for optimal text

detection. Once the image is aligned, the section specifically containing the identification details is isolated and cropped, reducing computational overhead in subsequent steps.

Text Detection with CRAFT: The cropped section of the image is relayed to the CRAFT model. Renowned for its text detection capabilities, CRAFT produces a detailed character density map. This map elucidates the coordinates of various text boxes, demarcating regions of textual data.

UNET Processing for Box Detection: The character density map is not the terminus. It serves as an input to the UNET model, a neural network known for its segmentation capabilities. UNET diligently processes this map to pinpoint masks corresponding to four pivotal data regions: Identity Number, First Name, Last Name, and Date of Birth.

Centre Position Ratio Calculation: Akin to a calibration step, the system calculates the ratio of the centre positions of the boxes detected by CRAFT in relation to the overall image dimensions. This computational step is pivotal as it ensures a standardised and precise alignment of detected text boxes, a foundation for accurate text extraction.

Orientation Adjustment with UNET Output: Harnessing the UNET output, the system discerns the orientation angle of the detected lines. If this angle exceeds a predefined threshold, it suggests an orientation anomaly. The system promptly rotates the RGB image to rectify this, ensuring textual data is horizontally aligned. This corrected image is then looped back to CRAFT for refined text detection.

Box Matching: Precision is the linchpin here. The algorithm matches the four boxes identified on the segmentation mask with the analogous boxes on the ID card. This validation step ensures data fidelity, guaranteeing that each box truly represents its designated section.

Multiple Box Detection: CRAFT's granularity can lead to the detection of multiple boxes, especially for individuals with extended names or surnames. The system is designed to accommodate this. It detects and segregates these additional boxes, ensuring that each name component is accurately represented and extracted.

Box Coordinate Update: Post detection, there's an imperative to ensure the entirety of the text is encapsulated within the detected boxes. To achieve this, the box coordinates are meticulously extended. These refined boxes, now encompassing all textual details, are ushered into the OCR stage.

Text Extraction with Tesseract OCR: The pièce de résistance is the Optical Character Recognition (OCR) stage. But prior to text extraction, the system undergoes a noise removal phase. This phase, employing advanced digital filters, ensures noise-free, clear textual regions. Post this, Tesseract OCR, a state-of-the-art text recognition engine, is summoned to extract pertinent details like name, surname, identification number, and date of birth with unparalleled accuracy.

Authentication

The implemented blockchain authentication system allows users to securely access and manage their accounts on the platform. It offers multiple options for authentication, including the use of ledger accounts and non-custodial key management. Ledger accounts involve the use of a unique combination of a username and password to access an account. Non-custodial key management involves the use of a third-party service, such as Google or Microsoft single sign-on, to authenticate the user. Multifactor authentication is used to verify the identity of clients accessing the platform, multifactor authentication provides users with the ability to verify their identity prior to executing sensitive transactions and can be configured depending on the security level the user wants to achieve – certain sensitive transactions (such as account recovery and processing transactions) can be secured with multiple levels of multifactor authentication (combination of SMS, Facial Recognition, Secret Keys). This level of multifactor authentication protects leaked Secret Keys from processing sensitive transactions and can be used to access locked accounts.

Creating a new account on VirtEngine generates a mnemonic seed, which is a series of words that can be used to log in to various wallets and manage the account. The mnemonic seed is an alternative to a private key and contains the solution to the private key through mathematical equations.

Authentication can be achieved through multiple options including Ledger Accounts, and Non-Custodial Key Management (which can be linked to password-less auth such as Google, and Facebook SSO and Active Directory Systems such as Azure AD, as well as Open-Source Variants).

Genesis accounts on VirtEngine have special permissions to access the admin portal and nominate service providers, support staff, and other administrators. They can also disable standard user accounts to prevent fraud and manage the platform.

A Genesis block is the initial block created with the initial user accounts and can be used as a method to differentiate Administrators that are operating the blockchain from standard users.

To ensure the security and privacy of sensitive data, VirtEngine uses data encryption based on third-party public keys. This means that only the intended recipient can decrypt the information using their private key. Sensitive data on VirtEngine includes order information, ID documents, support requests, resource details, account settings, and owned organizations. Only authorized user accounts can access this data, and it is encrypted in transit and at rest to prevent unauthorized access.

If a user were to lose access to his account through unauthorized access of his private keys, the user can nominate to setup a new private key utilizing multifactor authentication. Modification of multifactor authentication requires the user to also verify his identity with multifactor authentication or go through the identity process again.

List of MultiFactor Authentication Options (Figure 45):

- SMS Code (Text Format)
- Call Code (Audio Format)
- FIDO2 Hardware Key
- FIDO2 Passwordless Authentication (Biometric Fingerprints, FaceID – Device Locked protection) - Only pre-authorized devices
- VEID Facial Bioemetrics
- Trusted Browser
- 2FA Authenticator Protocol (Google Authenticator, Microsoft Authenticator)
- Certificate or Hardware Key

MultiFactor takes 2 Factor Authentication a step further by ensuring that the user has access to multiple scopes – and can limit the user by requiring them to verify an additional 2FA Scope – such as a combination of SMS + FIDO2, or a combination of VEID + FIDO2 in order to authenticate and sign sensitive transactions. A user may choose to trust the browser which would reduce the multifactor requirement by a single scope for further transactions done by that browser.

Hybrid Blockchain System

It is possible to use a single blockchain system with both public and permissioned access. This can be achieved through the use of a hybrid blockchain, which combines elements of both public and permissioned blockchains.

A hybrid blockchain is a type of blockchain that is designed to allow both public and permissioned access to the network and to the data stored on the blockchain. In a hybrid blockchain, access to the network is granted based on predetermined rules or criteria. This can include things like user identity, role, or jurisdiction. Users who meet these criteria are allowed to join the network and participate in the consensus process.

There are several ways to use a hybrid blockchain with both public and permissioned access:

Public access to certain data: A hybrid blockchain can allow public access to certain data stored on the blockchain, while restricting access to other data. For example, a hybrid blockchain could allow anyone to view the transaction history of a particular asset, while only allowing authorized parties to view the details of the transactions.

Public participation in the consensus process: A hybrid blockchain can allow public participation in the consensus process, while also allowing permissioned parties to participate. This can help to ensure that the network is secure and that transactions are validated by a diverse group of users.

Permissioned access to certain data: A hybrid blockchain can allow permissioned access to certain data stored on the blockchain, while restricting access to other data. For example, a hybrid blockchain could allow authorized parties to view sensitive data, such as personal or financial information, while only allowing public access to less sensitive data.

VirtEngine relies on a Hybrid Blockchain to operate, the following main roles are implemented within the system – GenesisAccount, Administrator, Moderator, Staker, ServiceProvider, Customer, SupportAgent.

GenesisAccount: Has uncontrolled access to the whole ledger, however can only decode data on the ledger that they have the private keys to. The majority of the system would be encrypted and cannot be accessed by the GenesisAccount, however the GenesisAccount has the ability to nominate Roles for other user accounts, as well as administrate user accounts by changing their account states to suspended, terminated, active, as well as controlling cloud marketplace **(103)** listings by changing their states.

Administrator: Has access to a large portion of the blockchain ledger to administrate the system – is also limited in that they cannot read private encrypted data from customers data, can nominate other accounts for different roles such as Moderators and ServiceProviders. Can control user account states to manage them as well as controlling cloud marketplace **(103)** listing states.

Moderators: Has access to moderating the system, has access to Support Requests submitted to the maintainers of the platform – can raise requests directly to administrators for further review.

ServiceProvider: Has access to list Cloud Computing services to the marketplace, can approve/reject customer orders, can suspend customer services, can leave public reviews on customer accounts, can report fraudulent activity to moderators/administrators of the platform.

Customer: Has access to purchase Cloud Computing services from ServiceProvider listings, has access to purchase High Performance Compute services from the VE Supercomputer can leave public review on purchased services, can cancel or delete existing orders under their account, can raise support requests directly to ServiceProvider related to purchased services. Can raise support requests to Moderators & Administrators of the platform.

SupportAgents: Has access to support requests on services listed by a ServiceProvider who has added them to their organization.

Staker: Staker accounts can be used to join the Proof-Of-Stake network for handling transactions on the VirtEngine Blockchain, Staker accounts use the following method to operate the network.

1. Users who want to participate in the consensus process and earn rewards must first hold a certain amount of the VE Tokens that is used on the network. This is known as the staking requirement – the staking requirements is only required for users looking to run their own staking node.
2. Users can then "stake" their VE Tokens by committing it to the network as collateral. This involves locking up their cryptocurrency in a special wallet or account that is designated for staking.

3. Users can either stake their VE Tokens in a staking node run by an organization with shared wallet, or to setup their own node and run their own staking server with their own wallet and collateral.
4. The network then selects the users who will participate in the consensus process based on the amount of cryptocurrency that they have staked. In general, users with larger stakes have a higher probability of being selected to validate transactions and earn rewards.
5. Once a user is selected to participate in the consensus process, they are responsible for validating transactions on the network and helping to reach consensus on the state of the blockchain.
6. If the user successfully participates in the consensus process and helps to secure the network, they will earn rewards in the form of cryptocurrency

The staking network also operates the Identity Network, which means that the nodes that handle transaction consensus are also used for verifying and identifying customer data such as ID Documents, etc. via the same Tensorflow algorithms and trained dataset – they are chosen the same way transactions are generally chosen via the consensus process.

For Data Encryption to work in a public blockchain, ensuring sensitive data is not stored in a public ledger – but instead can only be read by its intended participants. Messages between Providers and Users for example are encrypted based on the Third Parties Public Keys, one way encryption occurs in which only the third party can decrypt the information using their Private Keys. This is in conjunction with the Hybrid Blockchain, for example only certain roles can access certain data that is already encrypted. Meaning even if somehow a user where to bypass the role permissions and access such sensitive data, they will also need to have access to the Secret Keys that the data was encrypted for.

Examples of Data Encryption:

The examples provided below are not a full list of all the different scenarios that encryption occurs when sending data to the ledger, when building the system, you would need to identify whether the data being sent to the ledger is considered sensitive or not. Sensitive data then needs to be encrypted in a way that only the authorized user may access and read such data.

User uploading Identification Documentation: Encrypted to the Identity_Network chosen node via Staking System, which can then be decrypted and processed by Identity System nodes using their own private key to process the Identification Verification methods.

User sending an order to a ServiceProvider: Encrypted to the ServiceProvider's Public Key, which can then be decrypted and processed by the ServiceProvider.

Order approved by ServiceProvider and being deployed on OpenStack: The Cloud Marketplace **(103)** system will handle the deployment of the Order, however can only access the necessary API Secrets for the ServiceProvider's listing using the ServiceProvider's

own Encryption keys to decode and process the required API calls to VirtEngine Waldur. ServiceProvider's can run their own staking nodes which will have access to their SecretKey's, when a transaction that requires the ServiceProvider's key is to be processed – the validator will first process the transaction via Consensus regularly – then flow the data to the ServiceProvider's node to process the transaction through the ServiceProvider's Staking Node.

Order Deployed and Delivered to Customer: The Order details, such as the VM IPv4, State, and other relevant/secret information will be encrypted towards the Customer's Public Key, thus only the customer is able to decrypt and retrieve the deployed order.

Sensitive Data include Order Information, ID Documents, Support Requests, Resource Details, Account Settings, Owned Organizations, Team Members, Audit Logs, Account Details, Settings, Verified Identity details, etc.

Only user accounts or authorized systems that are intended to read this data will be able to decrypt the information using their private keys.

Blockchain can be used as a secure database for sensitive data using encryption. In a blockchain system, data is encrypted using a public key and can only be decrypted using the corresponding private key. This means that only the intended recipient, who has the private key, can decrypt and read the data.

Another way to use blockchain as a secure database for sensitive data is to use a permissioned blockchain, where only authorized users are allowed to access the data stored on the blockchain. This can be achieved using access control lists or by requiring users to present a valid digital certificate to access the data.

Overall, the use of encryption and access controls can help ensure that sensitive data stored on a blockchain is secure and can only be accessed by the intended recipient.

VirtEngine Cloud Marketplace **103**

The VirtEngine cloud marketplace **(103)** consists of the following components:

1. VirtEngine Waldur: a modular open-source hybrid cloud management system that allows multi-cloud integration, including private cloud, public cloud, HPC, and service desk.
2. VirtEngine Tokens: a cryptocurrency used to pay for services on the marketplace, which can be prefunded or purchased on the spot through a partner exchange.
3. Provider and user accounts: providers can offer services on the marketplace, while users can purchase these services and access sensitive data such as order information, ID documents, and support requests.
4. Data encryption: sensitive data is encrypted based on third party public keys, and only authorized user accounts can access and decrypt this data.

2024203136 11 Jul 2024

5. Integration with various cloud platforms: VirtEngine Waldur supports integrations with various private and public cloud platforms, HPC systems, and identity and service desk systems.
6. User interface: VirtEngine Waldur provides a user interface for users to interact with the marketplace.
7. Custom modules: Waldur allows the development of custom modules to carry out specific tasks, including communication with the VirtEngine blockchain.
8. Public ledger: VirtEngine Waldur supports storing public information on a public ledger, including provider and user accounts, offerings, and support requests.
9. Sensitive data storage: only authorized user accounts can access sensitive data such as provider offerings and support requests, which is encrypted in transit and at rest.
10. Genesis accounts: these special accounts have access to the admin portal and can nominate service providers, support staff, and other administrators, as well as disable standard user accounts to prevent fraud and manage the platform.

The VirtEngine cloud marketplace **(103)** utilizes VirtEngine Waldur (OpenNode Waldur Collaboration) codebase to facilitate cloud services between customers and providers. Waldur is a modular open-source hybrid cloud marketplace system that provides multi-cloud integration. Waldur currently supports a wide variety of integrations via its Plugin System.

Providers can offer services based on integrations such as Private Cloud, Public Cloud, HPC, and Service Desk utilizing the Marketplace system. Currently, Waldur supports PayPal for billing. Within VirtEngine Cloud Marketplace **103**, currencies such as USD/AUD/EUR would automatically be converted to VirtEngine Tokens when services are ordered through the VE Marketplace through a partner exchange or a decentralized exchange such as Uniswap. This automatic conversion of VE Tokens enables users to purchase services outside the VirtEngine marketplace utilizing VE Tokens via a payment processing system such as Visa or Mastercard.

VirtEngine Tokens can either be pre-funded into user accounts or purchased on the spot via an Exchange System and live price guides.

Providers, Validators (Stakers), and Network Participants will receive rewards in VirtEngine Tokens, these tokens can then be exchanged for a FIAT Currency such as USD through a partner exchange or spent to purchase other services from the marketplace.

VirtEngine's Waldur platform supports the following integrations, providing direct implementations for any Infrastructure, Platform, HPC, and other Services rendered through the Decentralized Cloud Marketplace **103**.

The current integrations supported by VirtEngine Waldur include:

Private Cloud: OpenStack, Kubernetes (Rancher), VMWare, OpenNebula (in development)

Public Cloud: AWS, DigitalOcean, MS Azure

HPC: MOAB, SLURM, Open OnDemand

Identity: Keycloak, EduGAIN, LDAP, SAML, Waldur Database

Service Desk: Atlassian Jira Service Desk, Zammad Ticketing System

Marketplace: Waldur In-Built Marketplace

Billing: PayPal, Waldur Billing

Automation: Ansible

Waldur is an advanced system that can be used to build complex Cloud Computing marketplaces, the goal of VirtEngine is to integrate the Waldur codebase into the Cosmos SDK (**102**) to run a decentralized version of Waldur within blockchain technology. Waldur will be integrated into Cosmos SDK (**102**) by porting the Cloud Marketplace codebase to within the blockchain project via API Interactions between the VirtEngine Blockchain and a localized Waldur API running within a Docker container.

Waldur offers a User Interface system that allows users to interact with the decentralized Cloud Marketplace system. This User Interface will be a port for users to register and utilize the system from an easy-to-use User Interface.

Waldur is a modular system and allows the development of custom modules to carry out specific tasks. A new module within Cosmos SDK (**102**) will be built to enable communication between Waldur and the VirtEngine Blockchain through API calls. Waldur will continue to run with its own database however will support public information downloaded from a public ledger, which includes Providers, User Accounts, Offerings, Support Requests, and more.

Sensitive data that should only be accessed by Administrators will be encrypted via Genesis accounts which run the staking network, only such nodes will be able to access sensitive encrypted data such as Provider Offerings details, Support Requests, Uploaded Identity scopes, to automate the platform via the Waldur Mastermind codebase.

VirtEngine Supercomputer

The VirtEngine Supercomputer consists of the following components:

- A decentralized network of nodes
- SLURM workload manager
- Kubernetes clusters
- Golang module within Cosmos SDK (102)

- Multiple computers connected over WAN or LAN
- Clustering of closely connected computing systems based on location & latency.
- Blockchain technology
- Rewards system in VirtEngine tokens
- Integration with VirtEngine Cloud Marketplace codebase to automate deployments of HPC Clusters via Kubernetes
- Accessible via VirtEngine Cloud Marketplace to schedule and pass on computing tasks.
- Custom module for communication with Cosmos SDK (**102**)
- Encryption of sensitive data
- Data encrypted in transit and at rest

The VirtEngine Supercomputer is a decentralized network of nodes that integrate the SLURM workload manager to operate a decentralized Supercomputer, SLURM will be deployed across nodes within Kubernetes clusters from the VirtEngine marketplace. As the VirtEngine marketplace grows there will be more available resources towards the Decentralized SLURM Clusters.

Blockchain computing allows running parallel codebases across multiple independent nodes, this allows us to facilitate automated deployments of SLURM workload nodes and add them to existing clusters to build the largest global Decentralized SLURM Compute marketplace.

A golang module will be implemented within the Cosmos SDK (**102**) to facilitate automated SLURM deployment and integration within the node that joins the mining network. This golang module will also facilitate rewards for the node that is offering its compute to the network.

It is generally accepted that a supercomputer is faster than a distributed computing network of the same size. This is because a supercomputer has all its components (e.g., processors, memory, etc.) located in a single location and connected by a high-speed local network. This allows for faster communication and data transfer between components, which results in faster overall performance.

On the other hand, a distributed computing network is made up of multiple computers located in different locations that are connected over a wide area network (WAN). The WAN connection may not be as fast as the local network used in a supercomputer, which can result in slower communication and data transfer between the computers in the network.

However, a distributed computing network has much higher scalability due to the fact it is not constrained by a single geographical environment – meaning that the overall computing power can easily exceed any existing supercomputer with ease.

Due to the geographical limitation of supercomputers, we believe that a Distributed Computing network built on blockchain can easily exceed the computing power available within the current supercomputing architecture.

VirtEngine supercomputer would also support the clustering of closely connected computing systems to build mini supercomputers within the overall distributed computing network – this can be used to schedule compute-intensive tasks within clusters that can communicate through a higher-speed network more effectively.

Patent Request:

Technical Innovation, the VirtEngine technical model of combining a Decentralized Identification System, a Decentralized Cloud Marketplace and a Decentralized Supercomputer is a novel and unique system that have been submitted into the patent system for protection.

This technical innovation allows us to deploy an autonomous business model using such an enclosed technical system that requires minimal intervention from the DET-IO Pty. Ltd. Team once deployed if VirtEngine & dependent codebases are maintained and further improved over time through the open-source model. The VirtEngine codebase will be released as Open Source under an Australian Creative Commons license with no commercialization allowed.

VirtEngine involves a physical aspect in that the software runs on multiple independent physical computing systems to achieve the desired outcome, for example, Cloud Computing services run on specialized Physical Servers to virtualize the environment and provide Virtual Computing to users.

The distributed computing network relies on Virtual & Physical computing to provide access to High-Performance-Computing to its users. The Identification system relies on mobile computing to verify the identities of users through cameras and fingerprint sensors.

Reproducing VirtEngine

Develop the VE Authentication & Encryption Modules:

1. Implement a secure login system using ledger accounts or non-custodial key management, linked to password-less authentication systems such as Google, Facebook, and Microsoft single sign-on.

2. Generate a mnemonic seed for new accounts, which can be used to log in to various wallets and manage their account.

3. Implement a system for granting permissions to Genesis accounts to access the admin portal, nominate service providers and support staff, and disable standard user accounts.

4. Implement data encryption using third-party public keys, allowing only the intended recipients to decrypt the information using their private keys.

5. Implement a system for restricting access to sensitive data, including order information, ID documents, support requests, resource details, account settings, and owned organizations, to an authorized user, accounts only using public encryption keys from the desired recipient.
6. Implement security measures to ensure that sensitive data is encrypted in transit and at rest.

Develop the VEID Module:

Train & Implement the Machine Learning algorithms:

1. Data collection: The first step in training a machine learning model is to gather a large dataset of examples. For an identification system, this might include a variety of documents such as driver's licenses, passport, or other forms of identification. It is important to have a diverse and representative dataset in order to ensure that the model is able to accurately identify individuals from a wide range of backgrounds and locations.
2. Data pre-processing: Once the data has been collected, it must be cleaned and pre-processed in order to make it usable for training the model. This may involve tasks such as removing duplicate or corrupted records, standardizing the format of the data, or filling in missing values.
3. Splitting the data into training and test sets: In order to evaluate the performance of the model, it is necessary to split the dataset into a training set and a test set. The training set is used to train the model, while the test set is used to evaluate the model's performance on unseen data.
4. Feature engineering: In order to help the model learn patterns in the data, it may be necessary to extract features from the raw data. This might involve extracting information such as the age, gender, or nationality of an individual from their identification documents.
5. Model training: Once the data has been prepared, it can be used to train the machine learning model. This typically involves feeding the data to the model and adjusting the model's internal parameters in order to minimize the error between the model's predictions and the true labels of the data.
6. Model evaluation: Once the model has been trained, it is important to evaluate its performance on the test set in order to determine how well it will generalize to new data. This might involve calculating metrics such as accuracy, precision, or recall.
7. Model fine-tuning: Based on the results of the model evaluation, it may be necessary to fine-tune the model by adjusting its parameters or adding additional layers to the model. This process may be repeated until the model achieves the desired level of performance.

8. Model deployment: Once the model has been trained and fine-tuned, it can be deployed in a production environment where it can be used to automate the verification of user identities.

Implement the VEID Module with the Machine Learning algorithm:

1. Develop a machine learning algorithm using TensorFlow that can accurately identify users based on various scopes, such as identity documents, biometric data, and facial recognition.
2. Create a mobile app that allows users to verify their identity using the camera on their device to scan documents, access biometric sensors, and perform facial recognition.
3. Integrate the VEID system with web scopes, such as Single Sign On (SSO) and email/SMS verification, to further verify the user's identity.
4. Integrate the VEID system with other web scopes such as Government Agency integrations, and Social Media data collection systems.
5. Implement an SMS/Email verification process that allows users to verify their email's and SMS number's.
6. Implement a domain verification process for users who want to register as providers on the VirtEngine marketplace.
7. Build a golang module for the VEID system that can be integrated into the Cosmos SDK (102) blockchain application. This module will use TensorFlow-GO to connect the TensorFlow trained machine learning algorithms with Cosmos SDK (102).
8. Extend the authentication module within Cosmos SDK (102) to support the verification of customer identities using the VEID system's API integrations with TensorFlow.
9. Allow users to upload encrypted versions of their identification scopes to the blockchain. The TensorFlow module will then decrypt this data using the Genesis Private Key.
10. Score users on a scale from 0 to 100, with 0 being an unknown identity and 100 being a fully verified user.
11. Use the TensorFlow algorithm to combine all of these scopes and recognize a user's identity through an automated, artificial intelligence-based process, achieving decentralized identification within the blockchain system.

Develop the VE Provider Daemon Module:

1. Developing a distributed computing network using blockchain technology to facilitate the deployment and management of computing resources from multiple providers.

2. Implementing a bid engine that queries for existing orders on the blockchain and places bids on behalf of the configured provider based on configured selling prices for resources.
3. Implementing code for interacting with clusters of servers offered by the provider, such as support for Kubernetes as a backend cluster management solution.
4. Developing a command line utility using the Cobra library to wrap the rest of the code and facilitate buildability.
5. Declaring pubsub events and implementing necessary code for the provider to take action on won leases and received manifests.
6. Implementing handler code for the REST server exposed by the provider to allow communication with the VirtEngine platform and other providers.
7. Developing code for parsing and handling manifests, including support for different manifest formats and the ability to deploy orders on behalf of users.
8. Implementing key management code for provider transaction signing, including support for Ledger accounts and non-custodial key management systems.
9. Implementing code for tracking and reporting provider metrics and usage data to the VirtEngine platform.
10. Integrating the VirtEngine API and other relevant external APIs to allow for automation and integration with third-party tools.

Develop the VE Cloud Marketplace **(103)** System:

1. Set up a distributed computing network using blockchain technology to facilitate communication between providers, validators, and network participants. This network would be used to store and transmit data related to provider offerings, user accounts, and support requests.
2. Integrate the VirtEngine Waldur codebase into the Cosmos SDK (102) to run a decentralized version of Waldur within the blockchain. This would allow for the creation of a decentralized cloud marketplace and enable communication between Waldur and the VirtEngine blockchain via API interactions.
3. Develop a custom module to communicate with the Cosmos SDK (102) and enable communication between Waldur and the VirtEngine blockchain. This module would allow Waldur to access public information from the public ledger, such as provider offerings and user accounts, and to automate the platform using the Waldur Mastermind codebase.
4. Use Genesis accounts to encrypt sensitive data such as provider offerings and support requests. Only nodes with access to these Genesis accounts would be able to decrypt and access this sensitive data, ensuring that it is only accessible to authorized administrators. Encrypt provider's sensitive data based on the provider's

encryption keys (API Secrets), Encrypt Customer sensitive data (resource, organization data) using the Customer's keys.

5. Set up a user interface using the Waldur codebase to allow users to easily register and interact with the decentralized cloud marketplace. This interface would provide a user-friendly way for users to access and utilize the system.
6. Implement various integrations, such as private cloud, public cloud, HPC, and service desk offerings, to allow providers to offer a wide range of services through the VirtEngine marketplace.
7. Implement a billing system using Waldur or a partner system such as PayPal to facilitate payments for services rendered through the marketplace.
8. Use VirtEngine tokens as a medium of exchange within the marketplace, allowing users to purchase services using these tokens or to exchange them for fiat currency through a partner exchange.

Develop the VirtEngine Supercomputer system

1. Implement a decentralized network of nodes that can communicate with each other over a WAN. This can be achieved by using a blockchain platform such as Cosmos SDK (102), which allows for decentralized communication between nodes.
2. Integrate the SLURM workload manager into the decentralized network of nodes. SLURM is a popular open-source workload manager that can be used to manage the allocation of resources and job scheduling within a cluster.
3. Implement a system that can deploy SLURM across nodes within Kubernetes clusters. Kubernetes is a popular container orchestration platform that can be used to manage the deployment and scaling of containerized applications. By deploying SLURM within Kubernetes clusters, we can ensure that the workload manager has access to the resources it needs to operate effectively.
4. Implement a golang module within the Cosmos SDK (102) to facilitate automated SLURM deployment and integration. This module would be responsible for automating the process of deploying SLURM workload nodes and adding them to existing clusters within the VirtEngine marketplace.
5. Use the golang module to facilitate rewards for nodes that are offering their compute resources to the network. This can be achieved by implementing a reward system within the golang module that distributes VirtEngine tokens to nodes based on their contribution to the network.
6. Implement a system for clustering closely connected computing systems to build mini supercomputers within the overall distributed computing network. This can be achieved by using tools such as Kubernetes to manage the deployment and scaling of containerized applications within the mini supercomputers.

7. Use the VirtEngine supercomputer to schedule compute-intensive tasks within clusters that can communicate through a higher speed network more effectively. This can be achieved by using the SLURM workload manager to allocate resources and manage job scheduling within the clusters.

Integrations:

For VirtEngine to function with all three components, they need to be integrated in various ways. The VirtEngine Cloud Marketplace is dependent on the Identification System as providers and users will need a way to verify the other party, whom they are dealing with to avoid fraudulent use. The identification system allows parties to leave feedback for each other based on their experience, this means users can leave public reviews and feedback for providers while providers can add fraudulent users to blacklists and leave public feedback on their accounts for other providers.

The VirtEngine Supercomputer is dependent on the VirtEngine Cloud Marketplace as it needs Infrastructure-as-a-Service to deploy & manage SLURM clusters via its Platform-as-a-Service system, SLURM clusters are deployed across providers within the VirtEngine Cloud Marketplace while mining nodes are added as compute nodes daemon to existing SLURM Workload clusters.

The VirtEngine Cloud Marketplace is also dependent on a benchmarking daemon that runs within added cloud computing clusters to collect metric data and transparently store it within the VirtEngine Blockchain to help users decide between providers and transparently compares public metrics available.

Blockchain Transactions:

VirtEngine utilizes the Proof-of-Stake model to handle transactions within its blockchain, in comparison to Proof-of-Work networks the power required to run the network is minimal. Proof-of-Stake allows users with the largest shares in stake to handle transactions within the network. The more a user has staked in the network the larger his investment is, which provides a secure way to run a blockchain network as these users need to ensure the validity of the blockchain for other users to trust the system. Generally, a user with a large investment will want to ensure his investment is secure and will be a good actor, other users with large stakes can also ban bad actors from participating within the staking model.

Cosmos SDK (**102**) (Blockchain Framework):

Cosmos SDK (**102**) comes with a module system that allows developers to implement custom application logic. VESC, VEID, and VECMP can all be built under custom modules.

Cosmos SDK (**102**) Architecture (Figure 40) describes how an application routes transactions into different modules. Auth Module for account management, Bank Module for monetary transactions, Staking module for network validation, Gov module for network governance and voting. Custom Modules such as the VirtEngine Provider Daemon can be built into Cosmos SDK (**102**) to provide additional functionality such as connecting to an external Third-Party API, in this scenario the open-source cloud marketplace system known as Waldur.

SLURM

SLURM (Simple Linux Utility for Resource Management) is an open-source workload manager and job scheduler for high-performance computing (HPC) systems. It is designed to manage the allocation of compute resources and the execution of jobs on a compute cluster. SLURM is widely used in the HPC community and is considered a complete solution for managing HPC resources and executing jobs. It provides features such as resource allocation, job scheduling, resource monitoring, and job accounting. It is a flexible and scalable tool that can be customized to meet the needs of various HPC environments.

VirtEngine Cloud Marketplace Daemon

The bid engine queries for any existing orders on chain, and based on the on-chain provider configuration, places bids on behalf of the configured provider based on configured selling prices for resources. The daemon listens for changes in the configuration so users can use automation tooling to dynamically change the prices they are charging w/o restarting the daemon. You can see the key management code for `provider` tx signing in `cmd/run.go`.

`cluster`

The cluster package contains the necessary code for interacting with clusters of compute that a provider is offering on the open marketplace to deploy orders on behalf of users creating `deployments` based on `manifests`. `Providers` could easily implement cluster management solutions such as OpenStack, VMWare, OpenShift, Azure, AWS, OpenNebula, CloudStack, Kubernetes etc through VirtEngine-Waldur's API which supports a wide array of Cloud Computing systems.

`cmd`

The `cobra` command line utility wraps the rest of the code here and is buildable.

`event`

Declares the pubsub events that the `provider` needs to take action on won leases and received manifests.

`gateway`

Contains handler code for the rest server exposed by the `provider`

`manifest`

Contains code to parse manifests presented by users and interact with the underlying cloud platform APIs orderbook.

Waldur Architecture

Waldur is a service for sharing resources across projects. It is based on the delegation model where a customer can allocate certain users to perform technical or non-technical actions in the projects.

Waldur is composed of a several components. (Figure 44)

- Homeport (web client, graphical interface) - React application
- Mastermind API server - Django/Django REST Framework application implementing the core business logic
- Celery workers - Background processing of tasks
- Celery beat - Scheduling of periodic tasks for background processing
- PostgreSQL database - Storing persistent data, also serves as Celery result store in Kubernetes deployment
- Redis - Tasks queue and result store for Celery (Docker Compose deployment only)
- RabbitMQ - Tasks queue and result store for Celery (Kubernetes deployment only)

Natural Language Interaction

The VirtEngine system can be enhanced by implementing LLM (Language and Learning Models) that allow users to interact with the system using natural language. This AI-based approach will make the interaction between users and the system more intuitive and user-friendly. Users would be able to submit commands to an AI Chat Agent that is then able to process the request within the VirtEngine System (by interacting on behalf of the user) as well as answer questions with models trained on VirtEngine related queries, as well as connect users with human support agents.

The algorithm will utilize Natural Language Processing (NLP) techniques to understand user input and identify keywords and phrases that indicate their requirements.

The LLM would be able to automatically generate API calls that are then processed by the system, for example a user may submit the following request: “Destroy all Virtual Machines that have the tag of DELETEME” – the LLM would then generate an equivalent API Call that destroys all Virtual Machines that have the tag DELETEME. Prior to processing the command, the LLM would attempt to identify the result of the action – this can be done by the LLM initiating a command that pulls all VM’s with the tag DELETEME and providing the user with a response such as “I have found 3 Virtual Machines with the Tag DELETEME, the details of these Virtual Machines is as follows:
#1: Database-Test, 192.168.1.1, Created: 04/05/2023
#2: Web-Test, 192.168.1.2, Created: 05/06/2023
#3: DemoDeployment: 192.168.1.3, Created: 01/24/2023

Are you sure you would like to proceed with deleting these machines? “

This response will act as a summary of the users request and also as a confirmation prior to any action being conducted.

DETAILED DESCRIPTIONS OF THE FIGURES

Figure 1:

100. "VirtEngine" **100** is a system that provides a cloud-based platform for users to access virtual machines, containers, and supercomputing resources. It connects to various other components in the system, including "VE Blockchain" (1**01**), "VE Cloud Marketplace" (1**03**), "Cosmos SDK" (1**02**), "VE Supercomputer" (1**06**), "VE Cloud Integrations" (**108**), "VE API" (**109**), and "VE User Interface" (1**10**).

101. "VE Blockchain" (**101**) is a decentralized ledger that allows for secure and transparent record-keeping of transactions within the VirtEngine system. It is connected to the "Public Ledger" (**114**) and "Genesis Account" (**131**) for storing and accessing transaction records, as well as the "Cosmos SDK" (**102**) for implementing blockchain functionality.

102. "Cosmos SDK" (**102**) is a software development kit for building blockchain applications. It is connected to the "VE Blockchain" (**101**) to provide the necessary functionality for the decentralized ledger.

103. VE Cloud Marketplace (103) is connected to and communicates with the VE API (component **109**) and the VE User Interface (component **110**). The VE API allows for programmatic access to the marketplace and enables users to automate the process of purchasing and selling cloud services. The VE User Interface provides a graphical interface for users to interact with the marketplace and make purchases and sales.

VE Cloud Marketplace (**103**) is also connected to the VE Waldur (component **107**) marketplace system which subsequently connects to VE Cloud Integrations (component **108**), which enables the marketplace to interface with various cloud service providers such as OpenStack (component **127**) and Kubernetes (component 128). This allows users to access and purchase a variety of cloud services from different providers through the VE Cloud Marketplace 103.

VE Cloud Marketplace (**103**) is further connected to the VE Supercomputer (component **106**) and the VE SLURM Clusters (component **113**). The VE Supercomputer is a high-performance computing system that provides powerful computational resources for users. The VE SLURM Clusters are groups of compute nodes (component **114**) that are managed by the Simple Linux Utility for Resource Management (SLURM) scheduling system. The connection between the VE Cloud Marketplace (103) and these components enables users to purchase and access the computational resources provided by the VE Supercomputer and VE SLURM Clusters through the marketplace.

VE Cloud Marketplace (**103**) also interfaces with the VE Tokens (component **105**), which are digital assets that can be used to purchase and sell cloud services on the marketplace. The VE Tokens are connected to the VE Blockchain (component **101**) and the Public Ledger (component **114**), which are used to track and verify

transactions made using the VE Tokens. VE Identification **104,** is a component that manages user identities within the VirtEngine system. It is connected to the VE Account **112** and the VE Custom Modules **111**, allowing it to verify user identities and grant access to resources within the system. It is connected to Tensorflow **117** in order to process data and provide an overall score for the identity of a user.

104. VirtEngine (VE) Tokens (**104**) are digital assets that represent the underlying value within the VirtEngine system. These tokens may be used for various purposes within the VirtEngine ecosystem, such as to purchase resources or services from the VE Cloud Marketplace (**103**), to access certain features or functionality within the system, or to stake in order to participate in the VE Staking Network (**150**).

VE Tokens (**104**) are typically stored within the Identity Wallet (**112**), which is a digital wallet that is used to manage and securely store digital assets. The Identity Wallet (**122**) is connected to the Identity Network (**111**), which is a decentralized network that is used to verify and validate the authenticity of the VE Tokens (**104**). This network may be connected to various identity services (**110**), such as those provided by government agencies (**151**) or social media platforms (**152**), in order to verify the identity of the user associated with the Identity Wallet (**112**).

VE Tokens (**104**) may also be used to access certain features or functionality within the VirtEngine system, such as the VE Supercomputer (**106**) or the VE Cloud Integrations (**108**). These components are connected to the VE Tokens (**104**) through the VirtEngine (**100**) and the Cosmos SDK (**102**), which is a software development kit that is used to build and deploy decentralized applications within the VirtEngine ecosystem. VE Tokens (**104**) may also be used to purchase resources or services from the VE Cloud Marketplace (**103**), such as virtual machines (**137**), containers (**138**), and VE Supercomputer (**106**) which may be used to run applications or workloads within the VirtEngine system.

105. VE Identification **105** is a component of the VirtEngine system as shown in **Figure 1**. It is responsible for verifying the identity of users who want to access the system and its various services. To do this, it employs a number of different methods including identity verification, identity services, authentication and identity network.

VE Identification **105** is connected with VE Authentication **112**, providing a method for users to access their user accounts via Ledger Accounts, and Non-Custodial Key Management.

The identity verification component **119** is responsible for verifying the identity of users through various methods such as email verification **147**, SMS verification 1**48**, and two factor authentication **150**. These methods are used to confirm the identity of users and ensure that only authorized individuals are able to access the system.

The identity services component **111** is responsible for providing services related to identity verification such as identity wallet 1**13**, biometric sensors 1**42**, and facial

2024203136 11 Jul 2024

recognition 1**42**. These services help to further confirm the identity of users and enhance security within the system.

The identity network component 1**12** is responsible for connecting the various identity-related components within the VirtEngine system and enabling communication between them. It ensures that the various methods and services used for identity verification are able to work together effectively to confirm the identity of users.

VE Identification **105** is connected to VE Tokens **104** through the VE API 1**09**. This connection allows the system to provide rewards to compute owners and charge customers using the system. By verifying the identity of users through the various methods and services described above, VE Identification 1**04** is able to ensure that only authorized individuals are able to access the system and its services, and that they are charged appropriately for their use.

106. VE Supercomputer 1**06** is a component within the VirtEngine system (**100**) that is responsible for providing high-performance computing (HPC) capabilities to users. It is connected to a number of other components within the system, including VE API **109**, VE User Interface **110**, VE Accounts **112**, VE Tokens **104**, VE Cloud Marketplace **103**, and VE SLURM Clusters **113**.

VE Supercomputer **106** is connected to VE API **109**, which allows it to interface with other components within the system and expose its capabilities to external applications and users. VE API **109** also enables users to access and interact with the supercomputer via a web-based interface, such as through a web browser.

VE Supercomputer **106** is also connected to VE User Interface **110** via VE API **109**, which provides a user-friendly interface through which users can access and utilize the supercomputer's capabilities. VE User Interface **110** may include features such as a graphical user interface (GUI), command-line interface (CLI), and various tools and resources for configuring and managing the supercomputer.

VE Supercomputer **106** is connected to VE Accounts **112**, which stores and manages user accounts within the VirtEngine system. VE Accounts **112** allows users to securely log in to the supercomputer and access their resources and configurations.

VE Supercomputer **106** is connected to VE Cloud Marketplace **103**, which provides Platform-as-a-Service functionality through VE Waldur **107**

VE Supercomputer **106** is connected to VE Tokens **104** through a connection that allows for the exchange of tokens as a form of payment or reward. When customers use the VE Supercomputer 1**06** to perform computational tasks, they may be required to pay in VE Tokens 1**04** as a form of payment. Similarly, the owners of the VE Supercomputer 1**06** may be rewarded in VE Tokens **104** for providing access to their computational resources. This connection between VE Supercomputer **105** and VE Tokens **104** allows for a flexible and secure means of exchange that can be easily

tracked and managed through the use of the VE Blockchain **101**. The VE Blockchain **101** serves as a decentralized ledger that records all transactions involving VE Tokens **104**, ensuring transparency and security for all parties involved.

VE Supercomputer **106** is also connected to VE SLURM Clusters **113**, which provides support for managing and scheduling workloads within the supercomputer. VE SLURM Clusters **113** may include features such as job scheduling, resource allocation, and monitoring tools for optimizing the utilization of the supercomputer.

107. VE Waldur **107** is a component of the VirtEngine system, as shown in Figure 2. It is connected to various other components of the system, including VE Cloud Integrations **108**, VE API **109**, VE Supercomputer **106**, and VE User Interface **110**.

VE Waldur **107** is responsible for providing a platform for managing cloud resources within the VirtEngine system. It allows users to create, provision, and manage virtual machines, containers, and other cloud resources through a user-friendly interface.

VE Waldur **107** is connected to VE Cloud Integrations **108**, which is responsible for integrating with various cloud platforms and services. This allows VirtEngine users to access and manage resources on a variety of cloud platforms, such as OpenStack and Kubernetes.

VE Waldur **107** is also connected to the VE Supercomputer **106** component, which allows it to manage the allocation of resources within the VirtEngine system's supercomputer **106**. This includes allocating compute nodes, managing the workload of the supercomputer, and ensuring that resources are used efficiently.

VE Waldur **107** is also connected to the VE API **109**, which provides an interface for interacting with the VirtEngine system programmatically. This allows developers to build custom applications and integrations that interact with the VirtEngine system and manage cloud resources.

Finally, VE Waldur **107** is connected to the VE User Interface **110**, which provides a user-friendly interface for interacting with the VirtEngine system and managing cloud resources. This allows users to easily access and manage their resources within the VirtEngine system.

108. VE Cloud Integrations **108** is a component of the VirtEngine system that enables the integration of the VirtEngine system with various cloud platforms. This component is responsible for providing an interface for connecting the VirtEngine system to various cloud platforms, such as OpenStack **127** and Kubernetes **128**, and for enabling the VirtEngine system to access and utilize the resources and services provided by these cloud platforms. The VE Cloud Integrations component is connected to the VE Supercomputer **106** and the VE Cloud Marketplace **103**, and it enables these components to access and utilize the resources and services provided by the various cloud platforms that are integrated with the VirtEngine system. The VE Cloud Integrations component is also connected to the VE API **109**, and it enables the VE API to access and utilize the resources and services provided by the various

cloud platforms that are integrated with the VirtEngine system. This allows users of the VirtEngine system to access and utilize the resources and services provided by the integrated cloud platforms through the VE API **109** and the VE User Interface **110**.

109. The VirtEngine API (**109**) is a software interface that allows different components within the VirtEngine system to communicate with each other and exchange data. It acts as a bridge between the various components of the system, allowing them to interact and collaborate to perform various tasks. The API allows the VirtEngine system to access and manipulate data stored in the VE Blockchain (**101**), VE Cloud Marketplace (**103**), VE Supercomputer (**106**), and other components of the system. It also enables external applications and systems to access and use the capabilities of the VirtEngine system through a set of defined interfaces and protocols. The API may be implemented using a variety of technologies, such as RESTful web services or GraphQL, and may be exposed to external clients through various methods, such as HTTP or WebSockets. The API may be secured using various authentication and authorization mechanisms, such as OAuth or JWT, as well as Multifactor Authentication for sensitive transactions to ensure that only authorized users and systems can access and use the VirtEngine system as its intended.

110. VE User Interface **110** is a component of the VirtEngine system that allows users to interact with and access various features and functionality of the platform. It is connected to VE Identification **104**, which is responsible for verifying the identity of users and providing secure access to the system. This connection allows the user interface to present personalized options and features to users based on their verified identity.

VE User Interface **110** is also connected to VE Tokens **105**, which is responsible for managing and issuing virtual tokens that can be used as a form of payment or reward within the VirtEngine system. This connection allows the user interface to present options for purchasing and using tokens, as well as displaying the user's current token balance.

VE User Interface **110** is connected to VE API **109**, which provides a programmatic interface for accessing and interacting with the VirtEngine platform. This connection allows the user interface to make requests to the API and display the resulting data and functionality to the user.

VE User Interface **110** is connected to VE Cloud Marketplace **103**, which is a platform for buying and selling computing resources within the VirtEngine system. This connection allows the user interface to present options for purchasing and selling resources, as well as displaying the current state of the marketplace.

VE User Interface **110** is connected to VE Supercomputer **105**, which is a platform for running high-performance computing tasks on the VirtEngine system. This connection allows the user interface to present options for running tasks on the

supercomputer, as well as displaying the status and progress of tasks that are currently running.

111. VE Custom Modules (**111**) component is used to extend the functionality of the Cosmos SDK blockchain (**102**) by providing a framework for the development of additional modules. The VE Custom Modules component is connected to the Cosmos SDK blockchain through the use of APIs, which allow the custom modules to interact with and make use of the features and functionality provided by the blockchain. In the VirtEngine system, the VE Custom Modules component is used to support the development of the **VEID** (VE Identification **104**), **VESC** (VE Supercomputer **106**), and **VECMP** (VE Cloud Marketplace **103**) modules, which provide additional features and functionality within the VirtEngine system.

112. VE Authentication **112** component is used to authenticate users into VirtEngine **100**, VE Authentication **112** provides access for users secured using various authentication and authorization mechanisms, such as OAuth or JWT, as well as Multifactor Authentication for sensitive transactions to ensure that only authorized users and systems can access and use the VirtEngine system as its intended. VE Authentication **112** also enables access to the platform via SSO Providers such as Google, Facebook, Microsoft, and Active Directory implementations.

113. VE Accounts **113** is a component that manages user accounts and their associated permissions within the VirtEngine system. This component allows users to create and manage their own accounts, as well as assign permissions to other users, enabling them to access and manage various resources within the system.

114. VE SLURM Clusters **114** is a component that manages the creation and deployment of SLURM clusters within the VirtEngine system. SLURM is a popular open-source cluster management and job scheduling system used in high-performance computing environments. This component allows users to create and manage SLURM clusters, as well as submit jobs to these clusters, through the VirtEngine User Interface **110**.

115. VE Compute Nodes **115** is a component that manages the allocation and scheduling of compute nodes within the VirtEngine system. Compute nodes are physical or virtual machines that are used to run jobs submitted to the VE Supercomputer **106**

116. VE Benchmarking **116**: This component is responsible for measuring the performance of the compute nodes within the VirtEngine system. This can be used to assess the capabilities of the system, optimize resource allocation, and identify potential bottlenecks.

117. VE Metric Data **117**: This component stores data related to the performance of the compute nodes in the VirtEngine system. This data can be used to track the

performance of the system over time, and to identify trends or patterns that may be relevant to optimizing resource utilization.

118. TensorFlow **118**, TensorFlow is a software library for machine learning and deep learning that allows users to train and deploy machine learning models. It is connected to the VirtEngine **100** and the VE Identification **104**, indicating that it is used for machine learning tasks within the system.

119. VE Mobile Application **119**, is a mobile application that allows users to access and interact with the VirtEngine system from their mobile devices. It is connected to the VirtEngine **100** and the VE User Interface **110**, indicating that it is used as a means for users to access and interact with the system.

120. Identity Verification **110**, is a function of the system that is used to verify the identity of users. It is connected to the VE Identification **104** and the Identity Services **110**, indicating that it is used to verify the identity of users through the use of various identity services.

121. Identity Services **111** is a system that is responsible for managing the identities of users within the VirtEngine system. It is connected to the Identity Wallet **113**, which stores the identity information and allows users to manage their identity information. It is also connected to the Identity Network **122**, which is a decentralized network of identity-verifying nodes that help to ensure the integrity and security of the identity information. The Identity Services **111** system is also connected to the Identity System **132**, which is a centralized system that is responsible for managing the overall identity infrastructure within the VirtEngine system. It is also connected to the Email Verification **148** and SMS Verification **149** systems, which are used to verify the identity of users through email and SMS messages. It is connected to the Two Factor Authentication **150** system, which adds an additional layer of security to the identity verification process by requiring users to provide a second form of authentication in order to access their identity information.

Identity Services provides the VE Identification system with a direct implementation to Tensorflow **118**

122. The Identity Network **122** is a decentralized network that is used to authenticate users and devices within the VirtEngine system. It uses blockchain technology to securely store and manage identity information, including personal data and authentication credentials. The Identity Network is connected to the Identity System **132**, which is responsible for managing the overall identity management process within the VirtEngine system. The Identity Network is also connected to the Identity Wallet **113**, which is used to securely store and manage digital assets, such as VE Tokens **105** and other digital currencies. Additionally, the Identity Network is connected to the Identity Services **111**, which provides a range of

services related to identity management, including identity verification, authentication, and authorization. The Identity Network is designed to provide a secure and reliable means of identifying and authenticating users within the VirtEngine system, ensuring that only authorized users have access to the resources and services provided by the system.

123. Identity Wallet **113**, is a digital wallet that is used to store and manage identity information for users. It is connected to the Identity Network **122** and the VE Accounts **113**, indicating that it is used to store and manage identity information for users within the VirtEngine system.

124. Service Offerings **124**: This component refers to the various services that are offered by the VirtEngine system. These services may include things like compute resources, storage, networking, and other infrastructure components that are necessary for running workloads on the system.

125. Public Ledger **125**: This component refers to the publicly-available ledger that is used to track the transactions and interactions that occur within the VirtEngine system. This ledger can be used to verify the authenticity of transactions, and to ensure that the system is operating in a transparent and trustworthy manner.

126. Sensitive Data Storage **126**: This component refers to the secure storage system that is used to store sensitive data within the VirtEngine system. This can include things like user credentials, financial data, and other information that needs to be protected from unauthorized access.

127. OpenStack **127**, is a software platform that provides cloud computing services for building and managing public, private, and hybrid clouds. OpenStack is connected to component **115**, Sensitive Data Storage, which is a system designed to securely store and manage sensitive data within the VirtEngine system.

128. Kubernetes **128**, is an open-source container orchestration system that is used to manage containerized applications in a clustered environment. Kubernetes is connected to component **137**, Virtual Machine, which is a software emulation of a physical computer that allows the VirtEngine system to run multiple operating systems and applications on a single physical machine.

129. Orders **129**, is a system used to manage orders placed within the VirtEngine system, including the tracking of orders and the provisioning of resources to fulfill those orders. Orders is connected to component **126**, Sensitive Data Storage, which stores and manages sensitive data related to the orders placed within the VirtEngine system.

130. HPC Plugins **130**, are plugins designed to enhance the performance and capabilities of the VirtEngine system for high-performance computing (HPC) applications. HPC Plugins are connected to component **136**, SLURM Workload Cluster, which is a system used to manage and schedule workloads on a cluster of compute nodes within the VirtEngine system.

131. The Identification System **131** is a component of the VirtEngine system that is responsible for verifying the identity of users and devices. It includes a range of tools and technologies that are used to confirm the authenticity of user identity claims and establish trust between parties.

The Identification System is connected to a number of other components within the VirtEngine system, including the Sensitive Data Storage **115**, the Genesis Accounts **126**, the Identity Services **110**, and the Identity Network **211**. It also has connections to external systems and services, including the Web Scopes **142**, Google **143**, Facebook **144**, and Microsoft **145**, which are used for identity verification and authentication.

Identity System is connected to component **139**, Camera, which is used to capture images of users for the purposes of identity verification. Identity System is also connected to component **140**, Biometric Sensors, which are sensors used to collect biometric data from users for identity verification purposes.

The Identification System is designed to be highly secure and reliable, using a combination of biometric data, facial recognition, and two-factor authentication to ensure that only authorized users and devices are granted access to the system. It also includes mechanisms for storing and managing sensitive data in a secure manner, including support for encrypted data storage and secure key management.

132. Genesis Account **132**, is the first account created within the VirtEngine system and is used to manage the initial distribution of tokens and resources within the system. Genesis Account is connected to component **115**, Sensitive Data Storage, which stores and manages sensitive data related to the User Accounts such as Order Information, Provider Connections, and overall administration data.

133. Support Requests **133**, is a system used to manage and track support requests made by users within the VirtEngine system. Support Requests is connected to component **133**, Service Desk System, which is a system used to manage and resolve support requests made by users.

134. Resources **134**, is a system used to manage and track the allocation and usage of resources within the VirtEngine system, including compute nodes, storage, and networking resources. Resources is connected to component **135**, SLURM Workload, which is a system used to manage and schedule workloads on the resources managed by the Resources system.

2024203136 11 Jul 2024

CLAIMS

1. A decentralized system for identification, authentication, data encryption, and cloud computing, comprising: a method that enables users to verify their identity and authenticate through document uploads, biometric sensors, facial recognition, and web-based scopes such as single sign-on, email verification, and SMS verification; a blockchain platform; a AI based facial recognition algorithm; a module supporting a cluster compute workload manager for deploying workloads across decentralized nodes; a module that integrates Cloud Computing services such as Virtual Machines, Containers; and a billing system for tracking the use of resources and generating invoices for users who purchased computing resources.
2. The system of claim 1, wherein the mobile app uses the camera to capture images of identity documents and the user's face for biometric and facial recognition.
3. The system of claim 1, wherein an algorithm compares the uploaded data with meta data collected from email, SMS, and single sign-on systems to recognize a user's identity through an automated, artificially intelligent algorithm.
4. The system of claim 1, wherein the system offers Decentralized Exchange services to convert Cryptocurrency into FIAT Currencies such as USD/AUD/EUR.
5. The system of claim 1, wherein the system offers multiple options for authentication, including ledger accounts, multifactor and non-custodial key management, which can be linked to password-less authentication systems such as Google, Facebook, Microsoft or Active Directory single sign-on.
6. The system of claim 1, wherein the system uses data encryption based on third party public keys to ensure sensitive data is not stored on the public ledger and can only be accessed by authorized user accounts.
7. The system of claim 1, wherein the system uses a Proof-Of-Stake consensus process to handle transactions and identification verification.
8. The system of claim 1, wherein the decentralized cloud marketplace allows users to rent out their computing resources and deploy workloads across a decentralized network of nodes.
9. The system of claim 1, wherein the HPC module includes a library of pre-configured HPC workloads that can be deployed on demand, as well as support for custom workloads.
10. A system that combines two of the following independent systems: Decentralized Identification, Decentralized Cloud Computing, and Decentralized Distributed Computing. Such as a system that uses Decentralized Identification & Decentralized Cloud Computing together or a system that uses Decentralized Cloud Computing & Decentralized Distributed Computing.

2024203136 11 Jul 2024

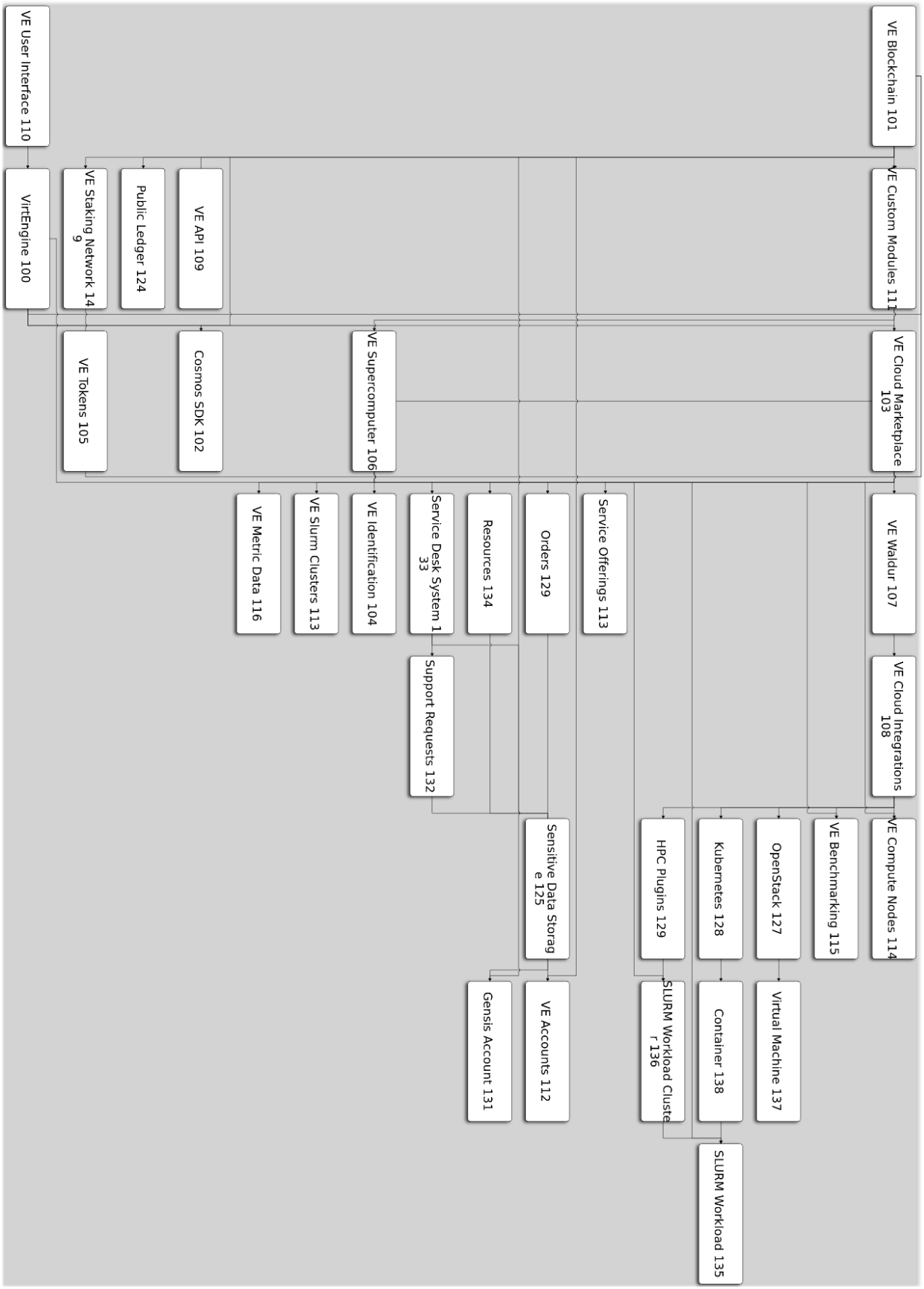
1. Decentralized Identification consists of a method used to verify user identities in a distributed system such as Blockchain.
2. Decentralized Cloud Computing consists of a method used to provide Cloud Computing services in a distributed system such as Blockchain.
3. Decentralized Distributed Computing consists of a method used to provide High-Performance-Compute or Cluster Computing in a distributed system such as Blockchain.

11. A method for providing decentralized cloud computing services using a blockchain network, comprising:
    1. Allowing providers to integrate their cloud computing services through a marketplace system.
    2. Receiving a request for a cloud computing service from a user;
    3. Storing certain data on the blockchain network in an encrypted form;
    4. Determining available resources within the decentralized network that can fulfill the request;
    5. Allocating the requested resources to the user;
    6. Providing the cloud computing service to the user using the allocated resources;
    7. Providing the user with access to manage the cloud computing service.
    8. Tracking the usage of the allocated resources and providing a record of the usage on the blockchain network.
    9. A user interface for receiving requests for cloud computing services from users, and allowing providers to enter their cloud computing services;
12. A method for providing decentralized identification services using a blockchain network, comprising:
    1. Receiving a request for identification verification from a user;
    2. Verifying the user's identity using a combination of biometric data and identifying documents;
    3. Storing certain identity data on the blockchain network in an encrypted form;
    4. Processing the Identity Data using an ML/AI algorithm to provide a scoring system for the user's identity.
    5. An access control module for providing access to the encrypted identity data to authorized parties upon request.
    6. A biometric data capture module for collecting biometric data from the user;

13. A method for providing a decentralized supercomputer comprising a network of nodes, the method comprising:
    1. deploying the supercomputer across multiple nodes in a decentralized manner;
    2. integrating a workload manager to operate the supercomputer;
    3. offering the supercomputer for use from a cloud marketplace system;
    4. providing rewards to nodes offering their compute resources to the supercomputer.
14. The method of claim 11, wherein a network of nodes that are located within close proximity are clustered to act as mini supercomputers for HPC tasks that are submitted by users.

2024203136 11 Jul 2024

Figure 1A:



11 Jul 2024
2024203136

Figure 1B:

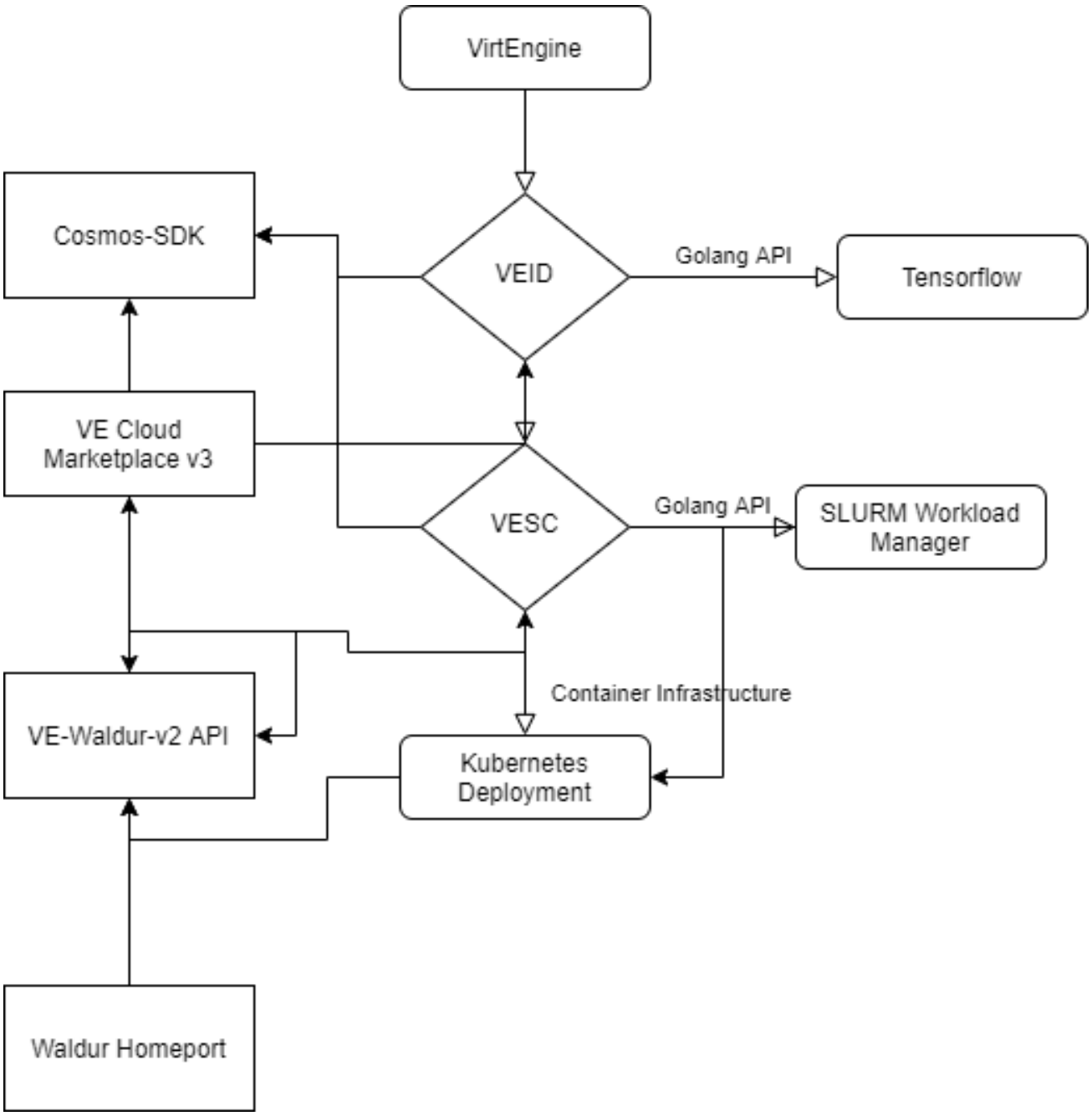
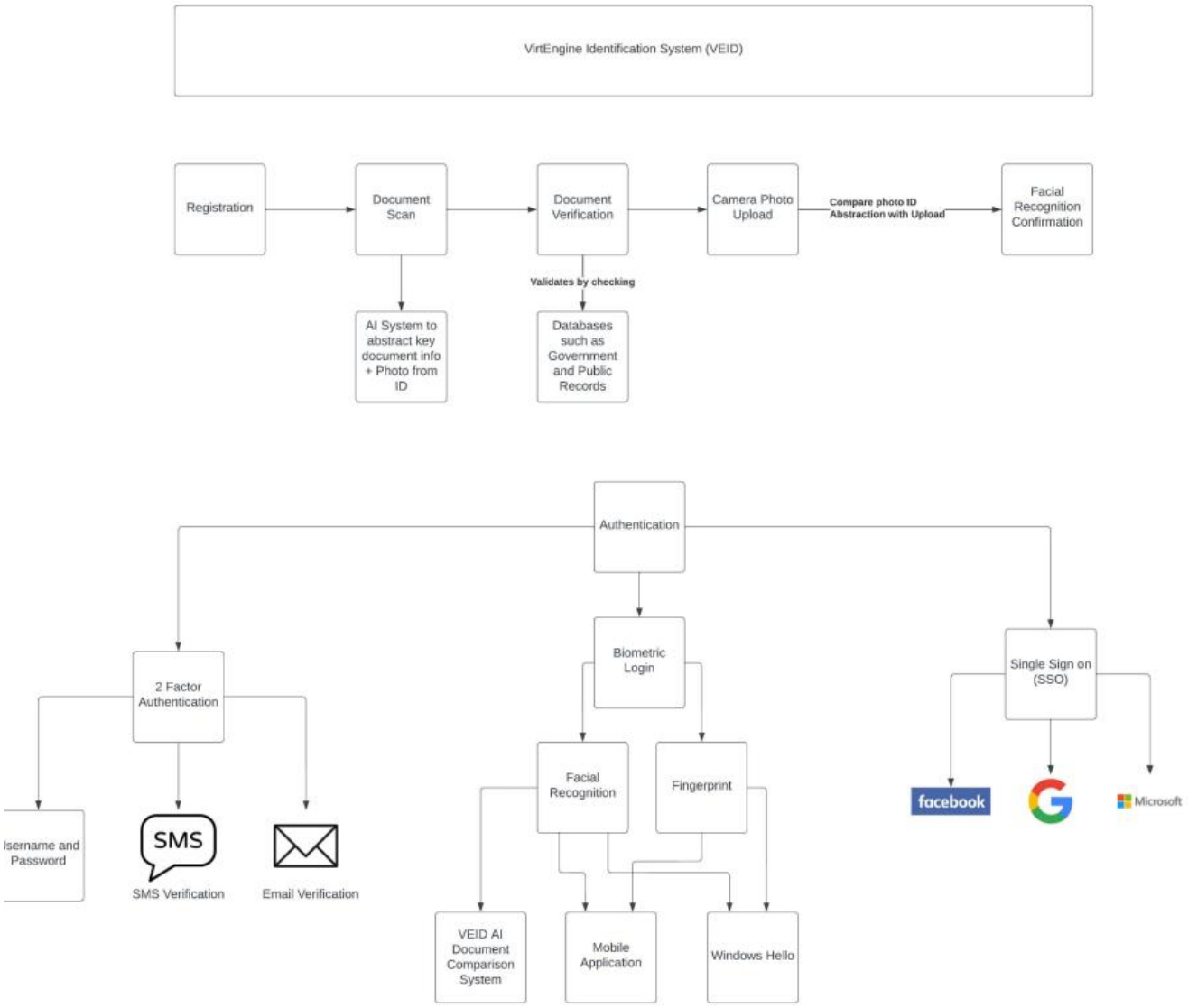


Figure 2A:



2024203136 11 Jul 2024

Figure 2B:

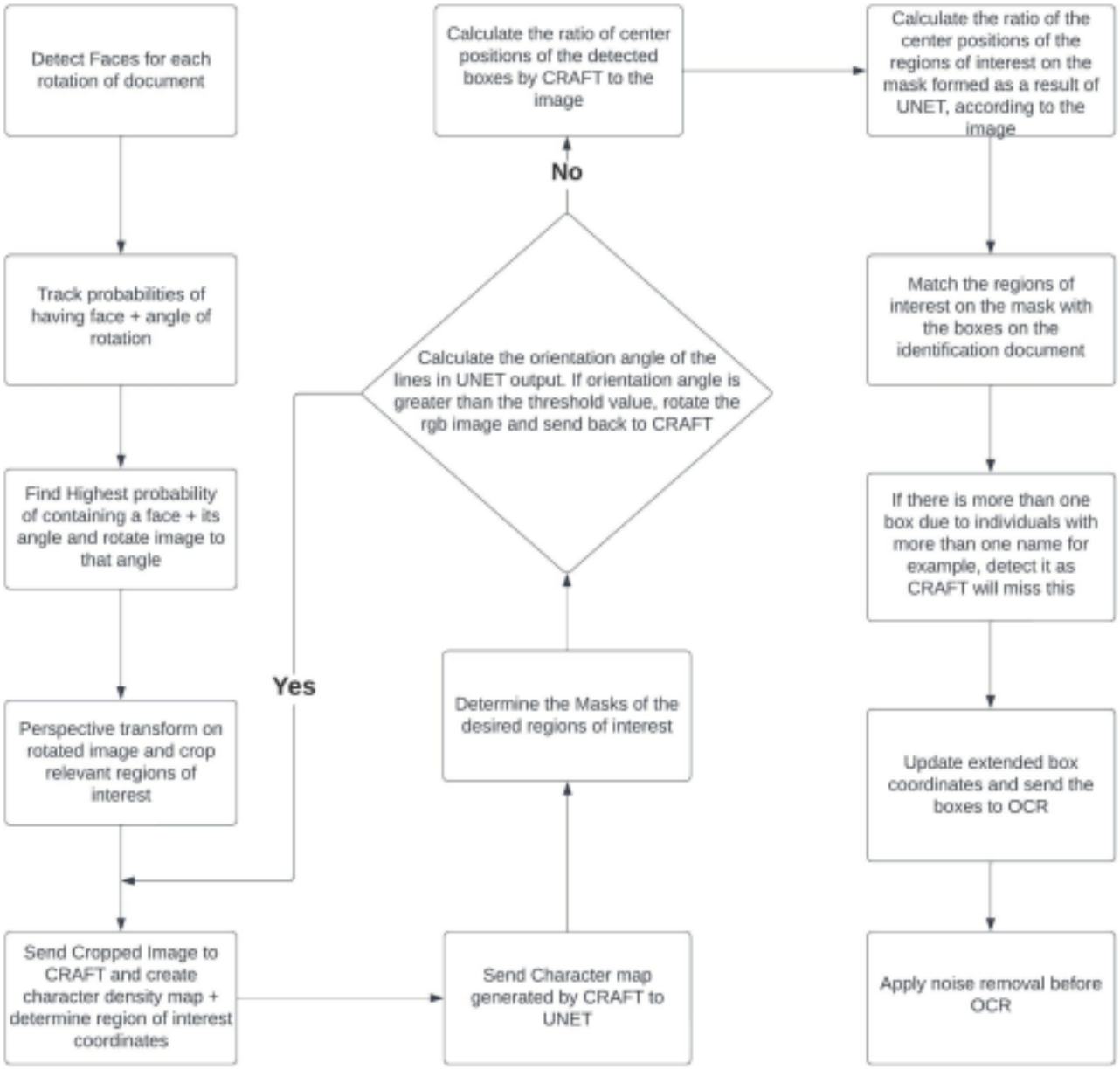


Figure 3 (Dashboard):

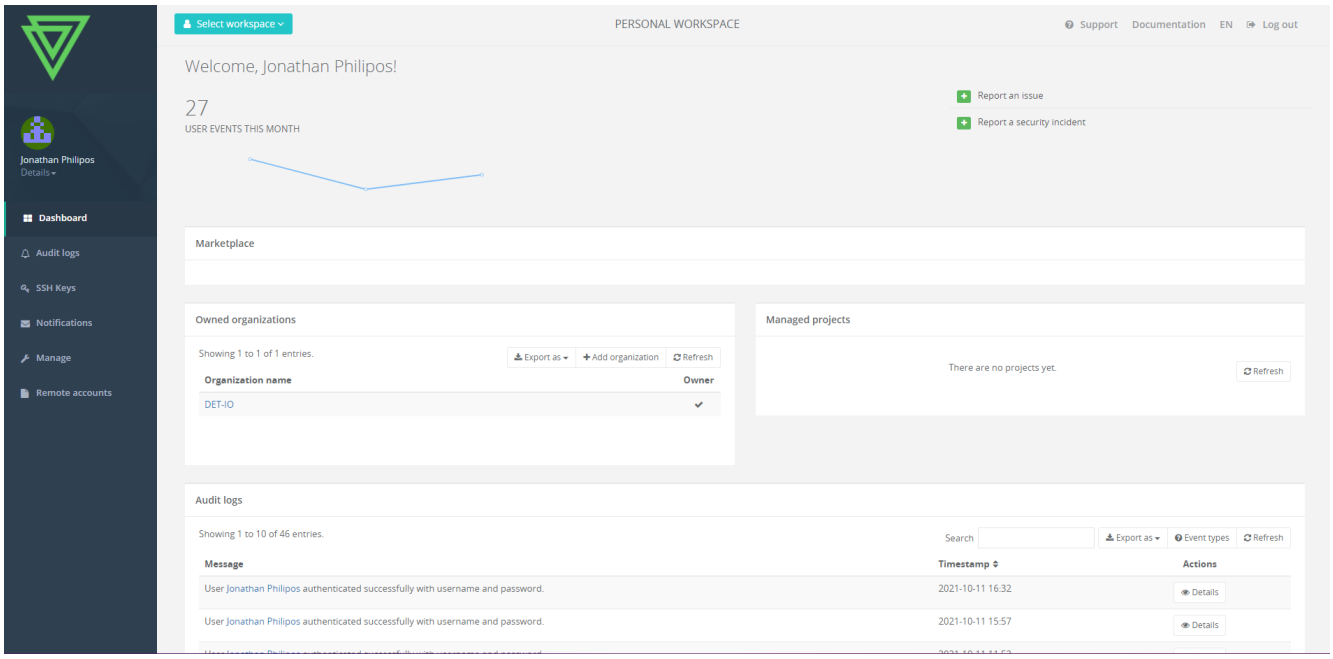


Figure 4 (Add Public Offering: Overview):

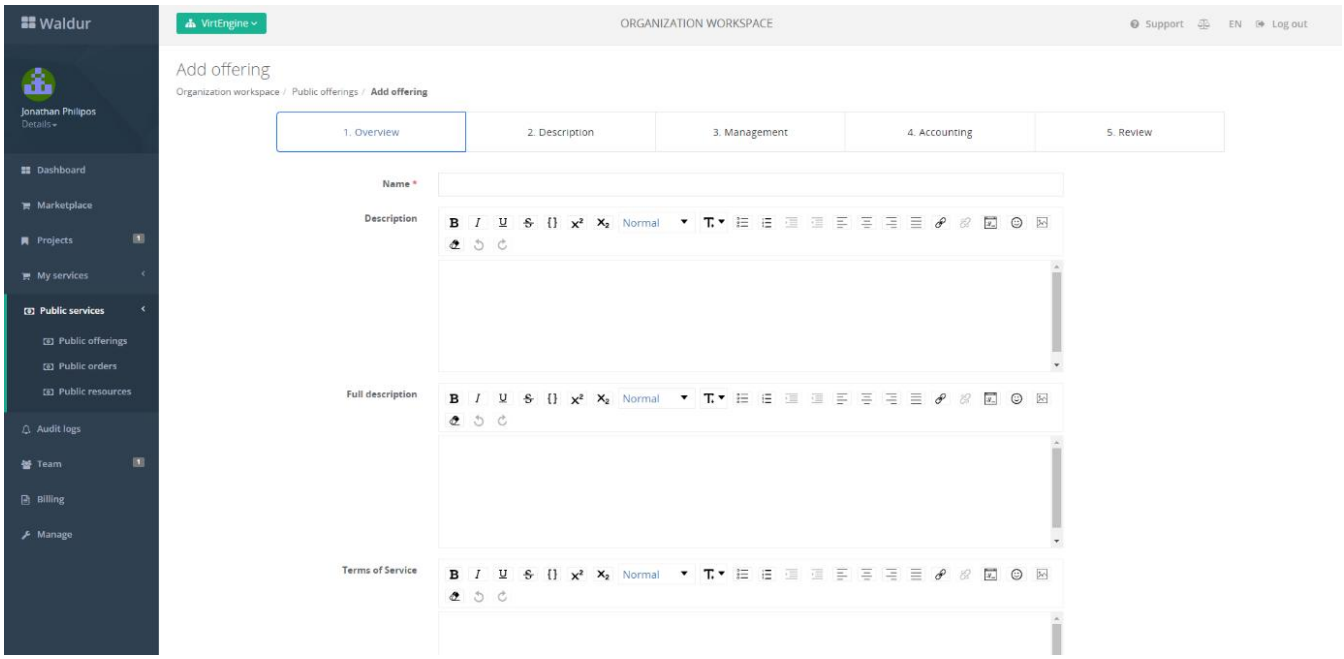


Figure 5 (Add Public Offering: Description):

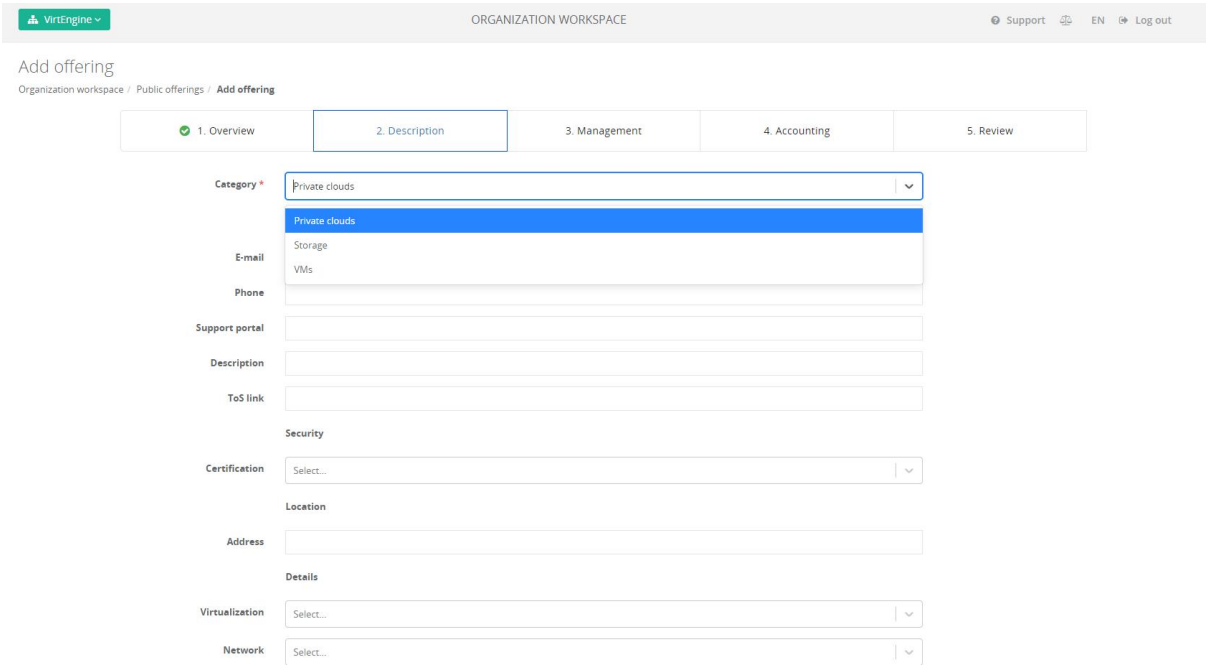


Figure 6 (Add Public Offering: Management)

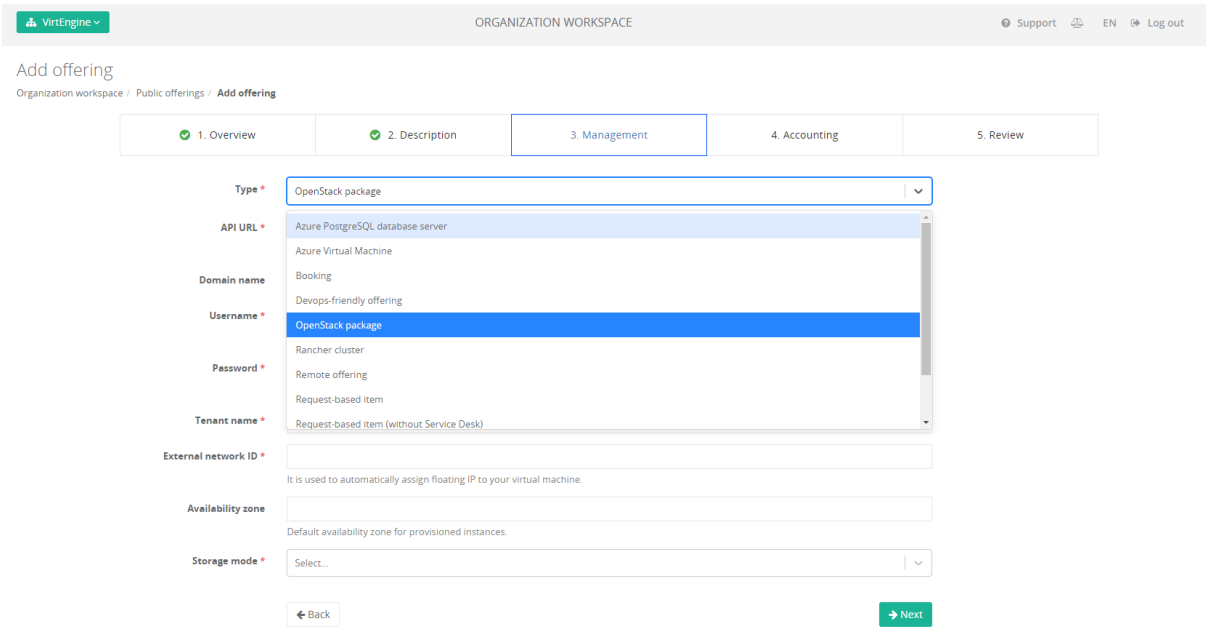


Figure 7 (Add Public Offering: Accounting Components)

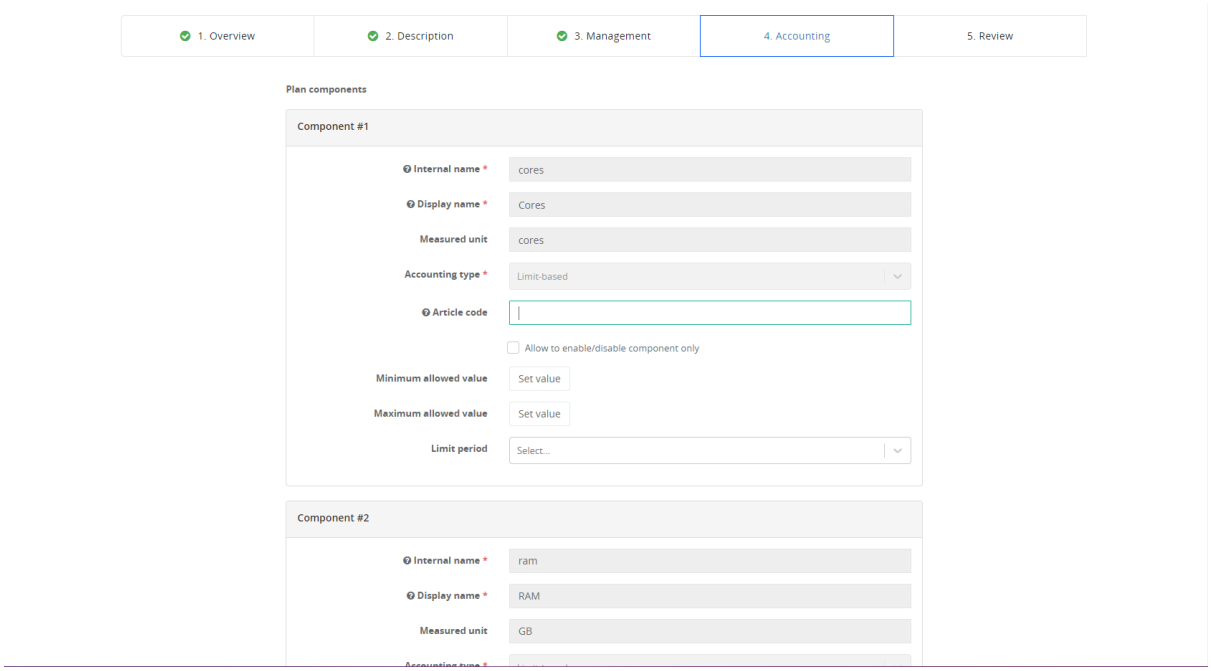


Figure 8 (Add Public Offering: Accounting Plans)

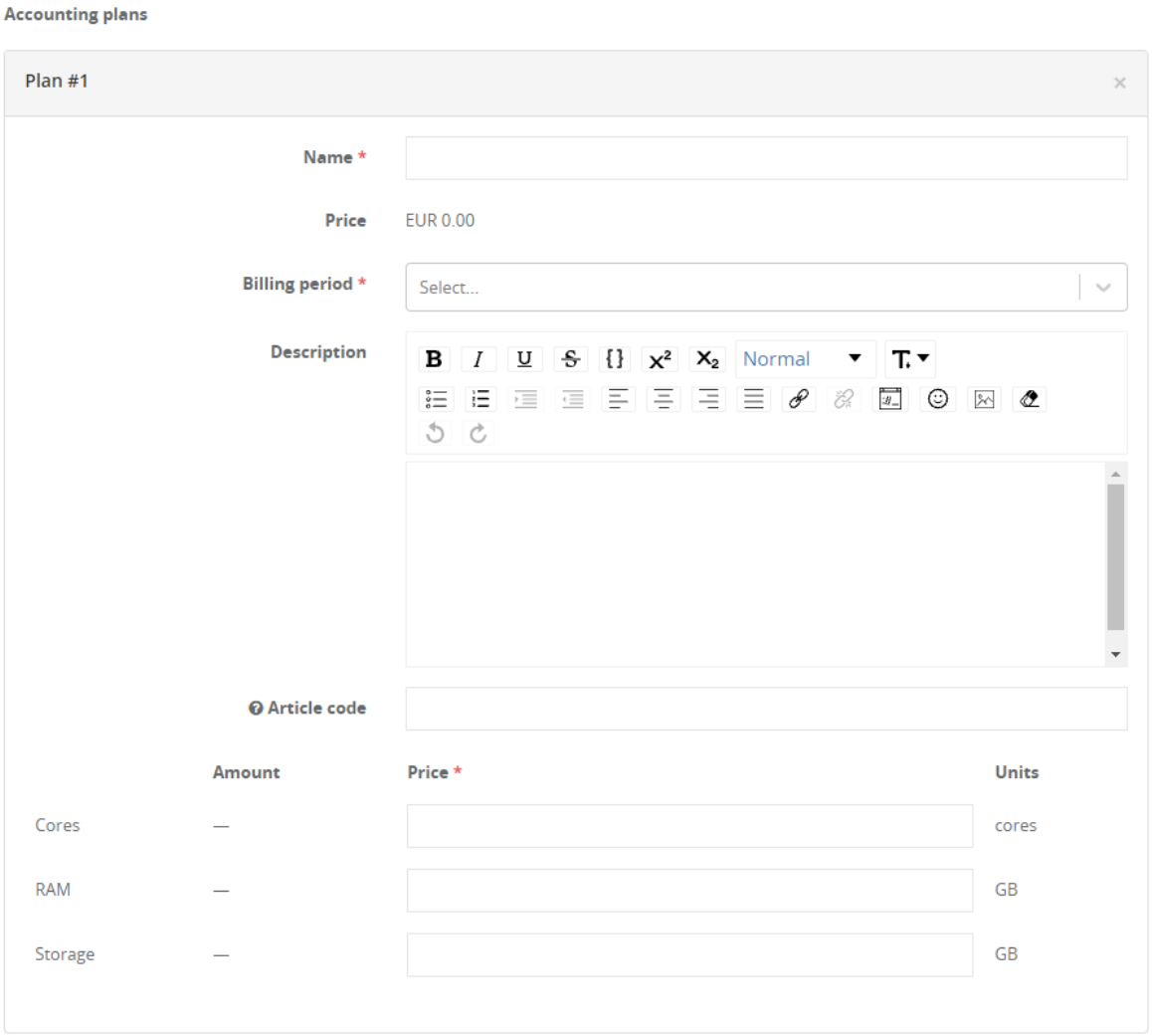


Figure 9 (Review Offerings)

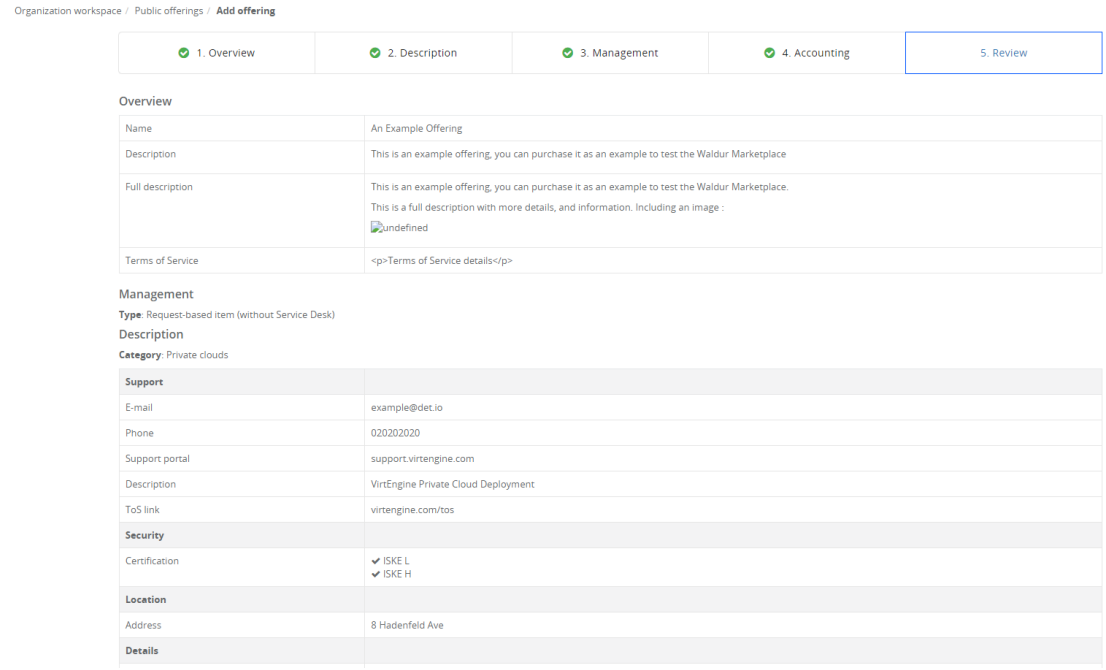


Figure 10 (Public Offering: Edit Offering)

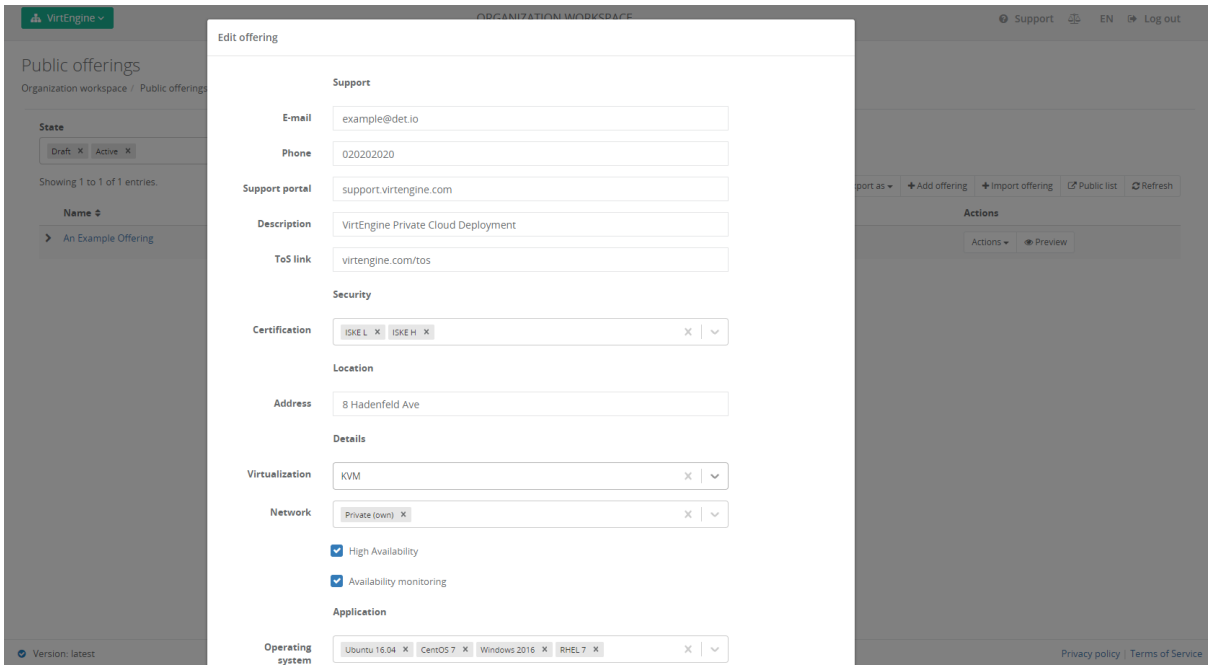


Figure 11 (Global Cloud Marketplace)

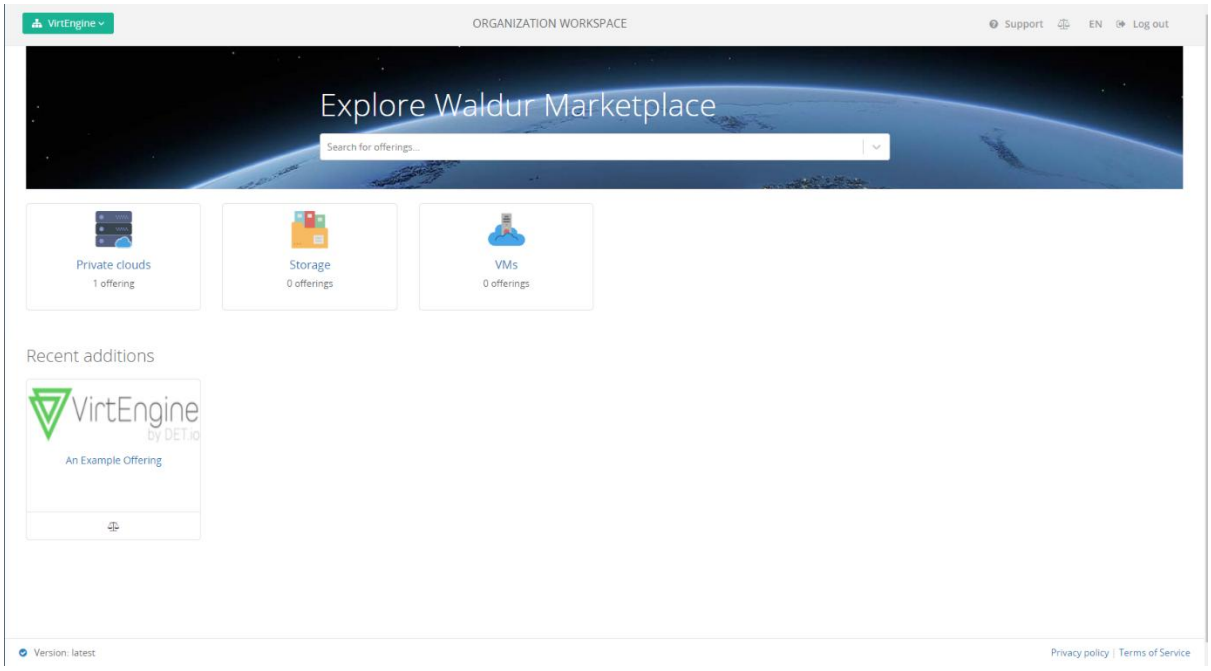


Figure 12 (Example Offering)

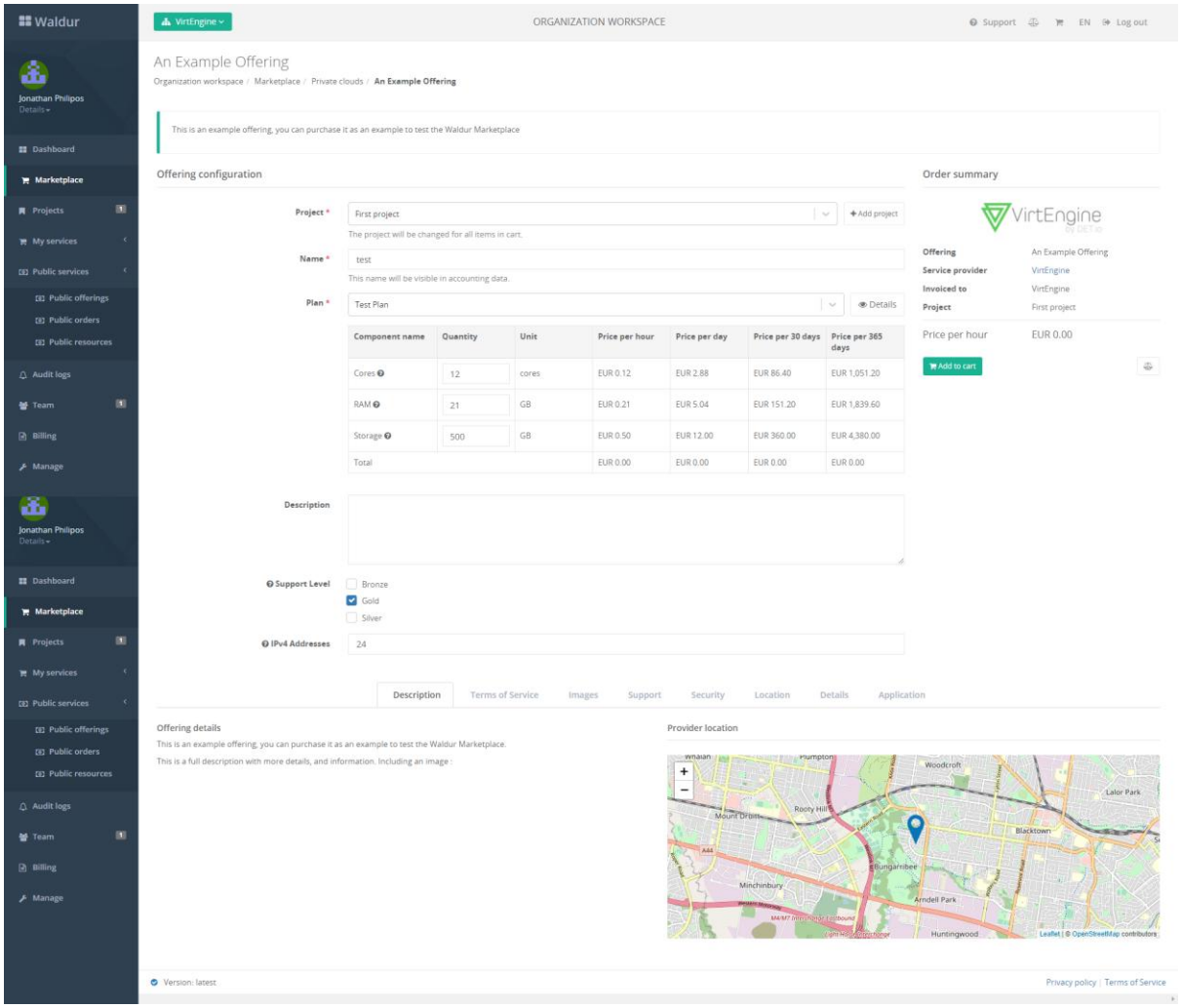


Figure 13 (Check Out: Configure)

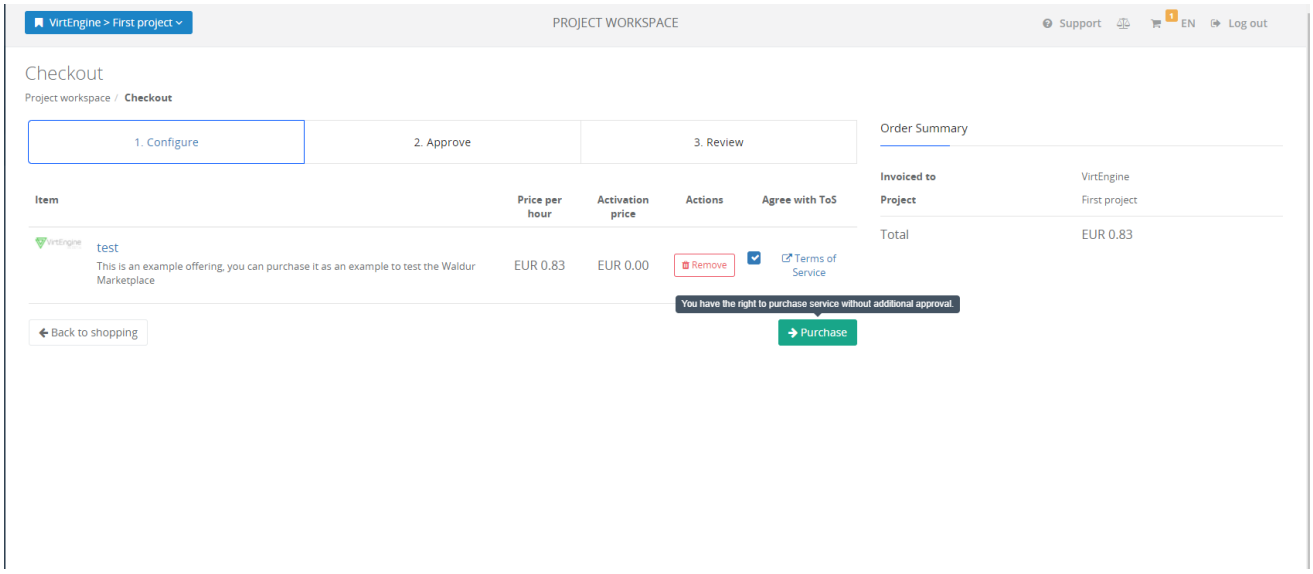


Figure 14 (Resource Example)

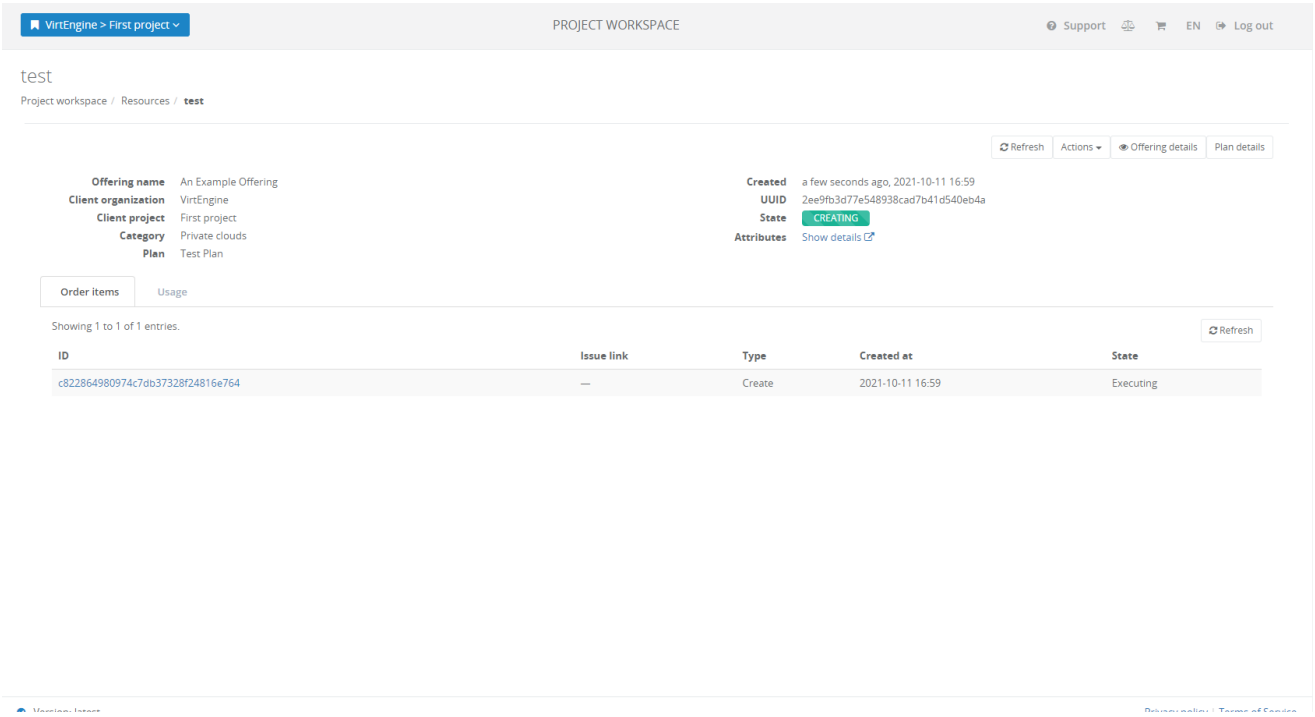


Figure 15 (Order Details)

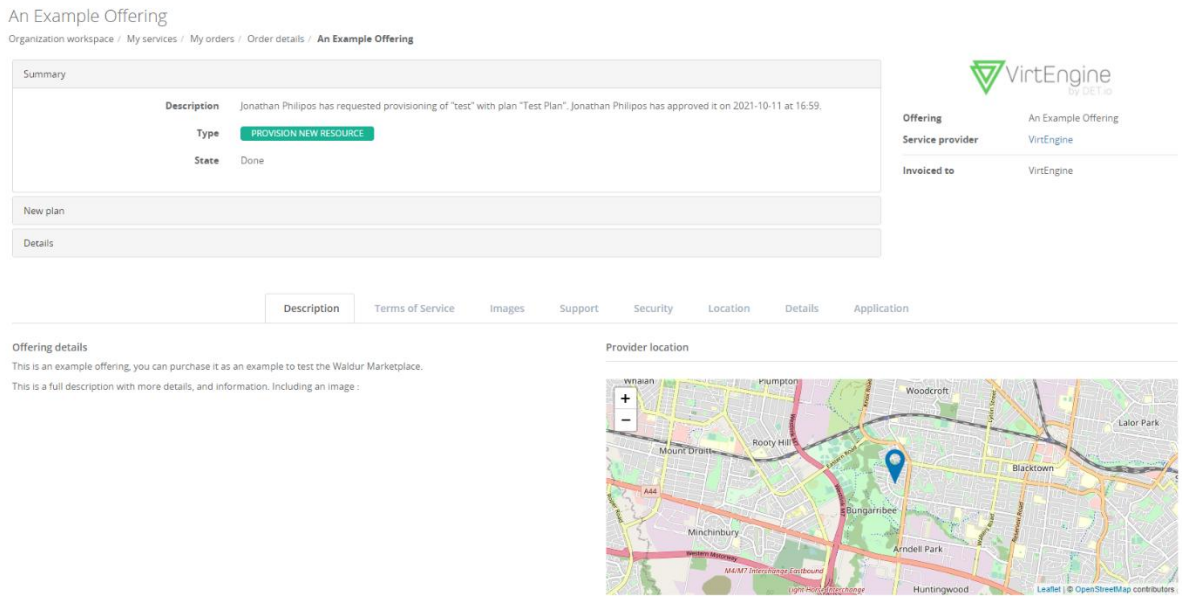


Figure 16 (VirtEngine Identification Mobile Application):

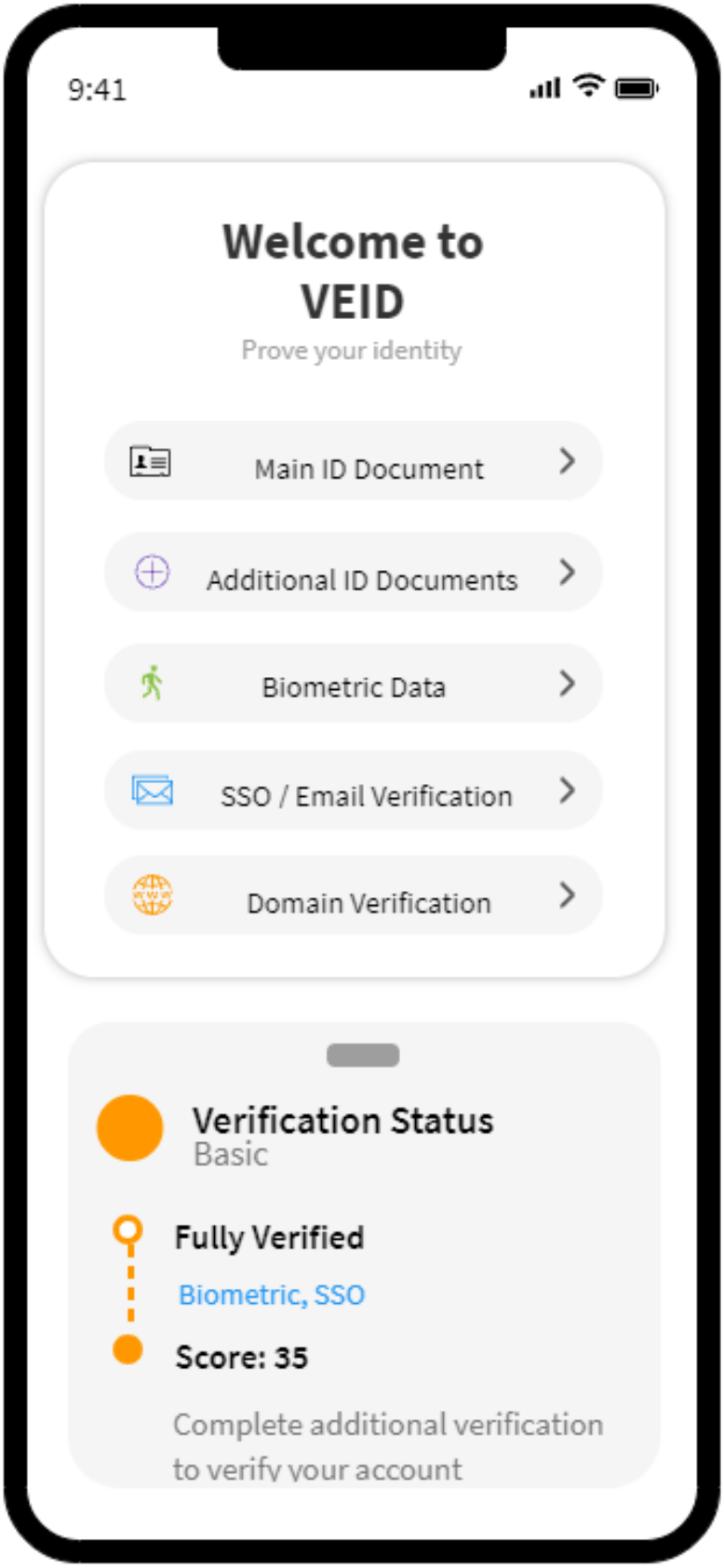


Figure 17 (VirtEngine MultiFactor Authentication)

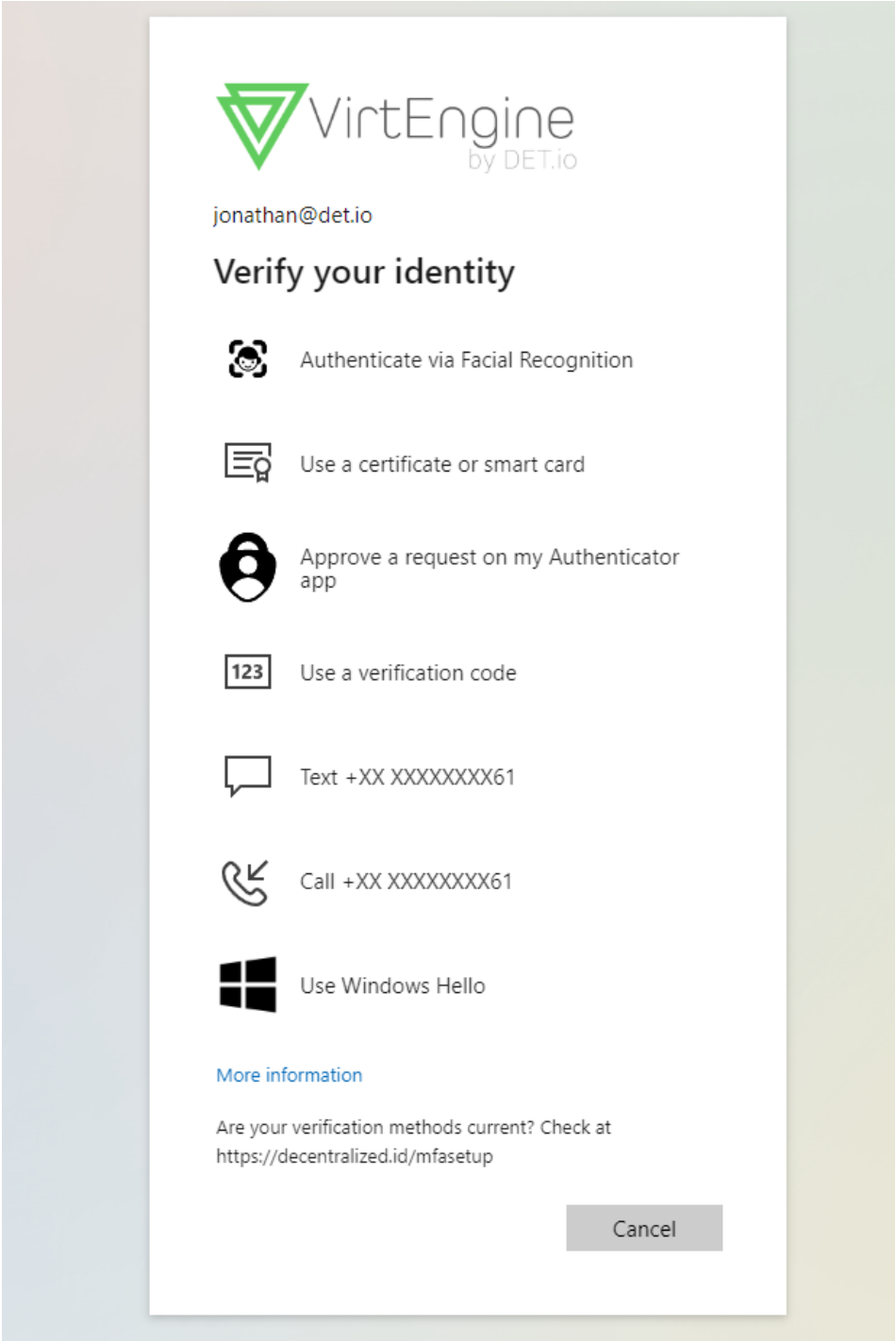


Figure 18A (Waldur Architecture)

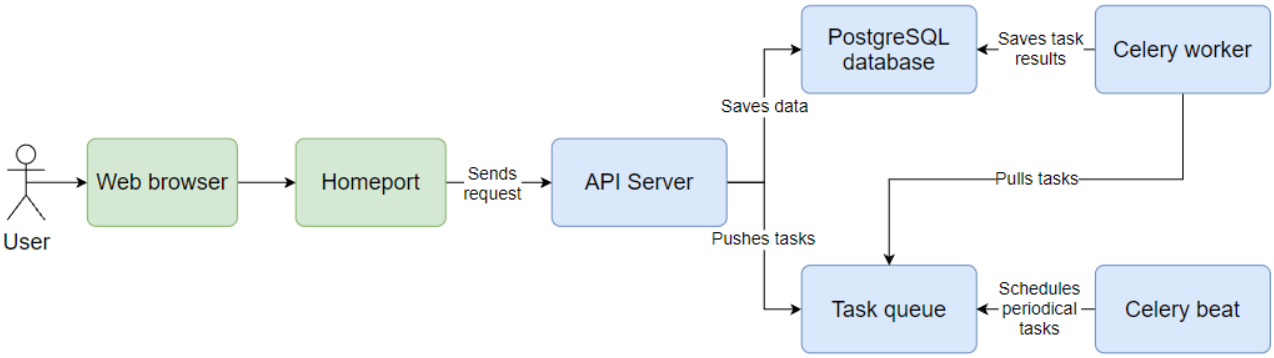


Figure 18B (Waldur Architecture)

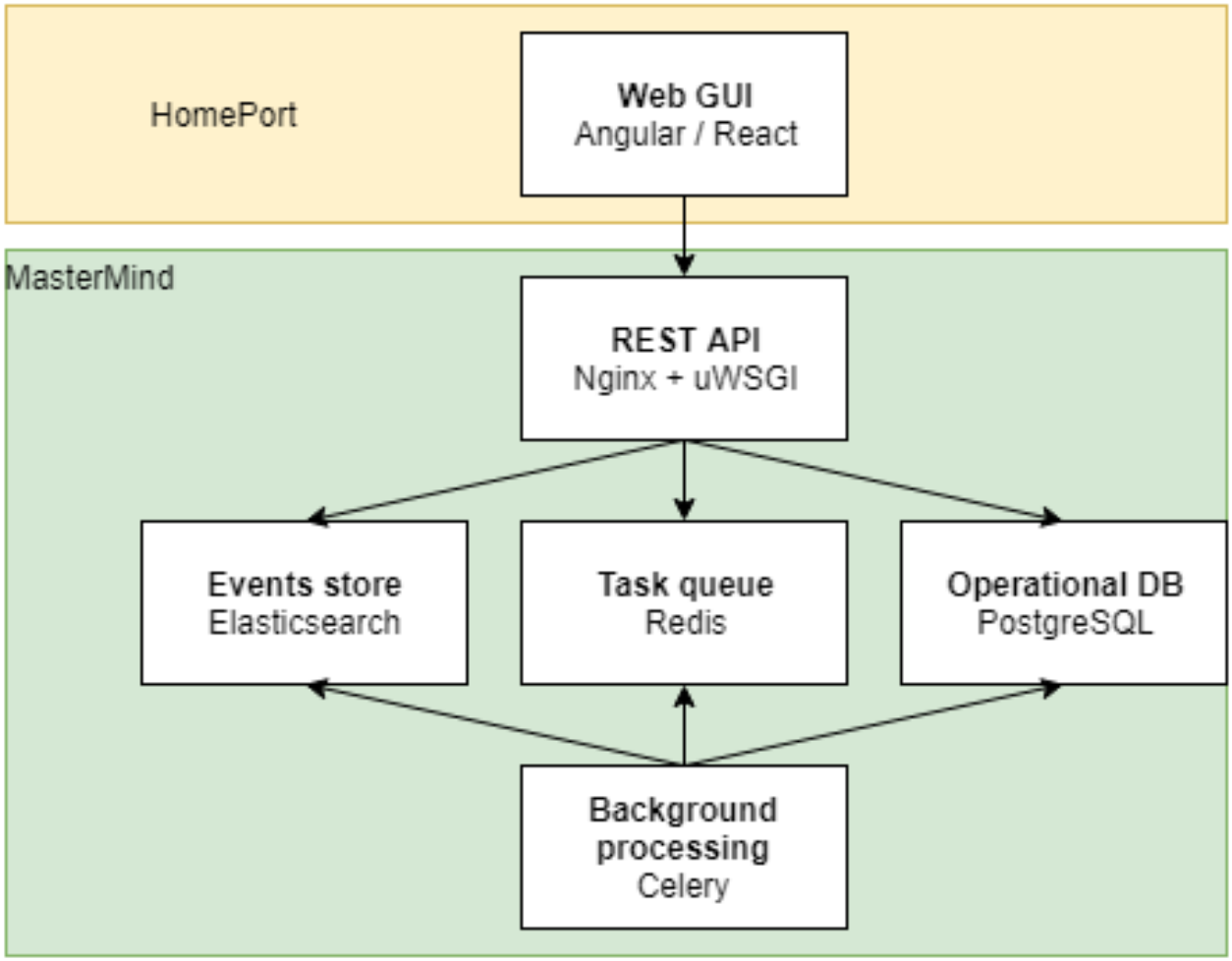
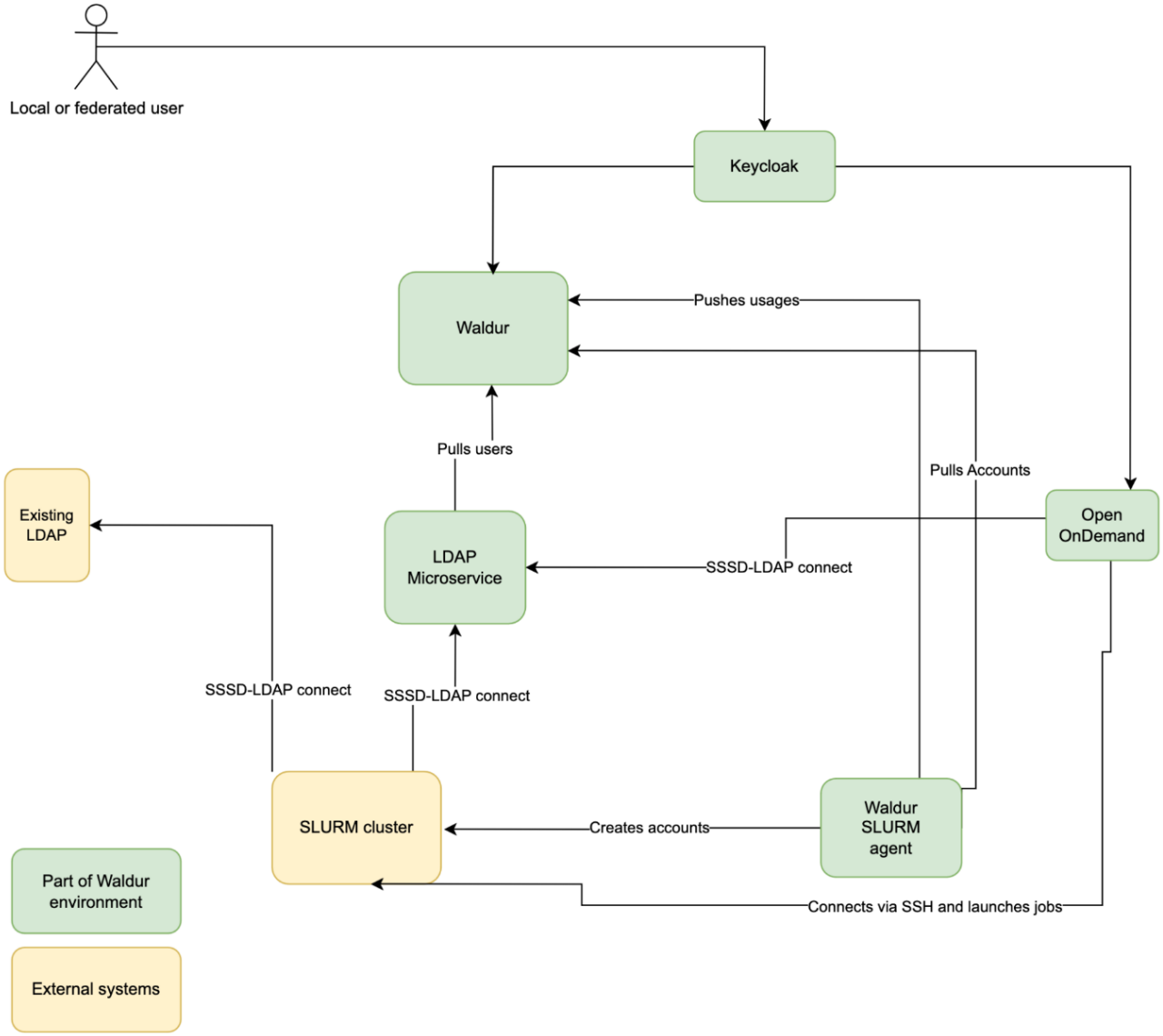


Figure 19 (Waldur SLURM Diagram)



Referenced Codebases:

VE-waldur-v2 (107):
https://github.com/virtengine/ve-waldur-v2

VirtEngine (100):
https://github.com/virtengine/virtengine

VE-portal-v2:
https://github.com/virtengine/ve-portal-v2

Cosmos SDK (102):
https://github.com/virtengine/Cosmos SDK (102)

Tendermint:
https://github.com/virtengine/tendermint

Tensorflow:
https://github.com/tensorflow/tensorflow

SLURM:
https://github.com/SchedMD/slurm

Kubernetes:

https://github.com/kubernetes/kubernetes

OpenStack:
https://github.com/openstack/openstack

References Cited:
https://docs.cosmos.network/master/building-modules/intro.html

VirtEngine Waldur documentation:
http://docs.waldur.com/

Waldur+DET.io Partnership:
https://waldur.com/solutions/hosting-provider/

OpenStack
https://www.openstack.org/

Cosmos SDK (102)
https://v1.cosmos.network/sdk

SLURM:

https://slurm.schedmd.com/

Inventors:
Jonathan Philipos – Designed VirtEngine architecture, contributed towards Waldur feature implementations (Marketplace System, Kubernetes) –

DET-IO PTY LIMITED, Australia

Ilja Livenson – Developed and implemented Open Source Waldur Cloud computing system.

OpenNode LLC, Estonia

Contributors:

Zhaokun Chen
Yifei Chen
Liam Dao
Rohan Poorun

Royal Melbourne Institute of Technology School of Engineering, Australia