**MODULES:**

* user
* cloud
* admin
* REST API.

**User**

It defines the access rights of the cloud users. A volume can be created, if the it has not exceeded its quota of the permitted volumes and a user Authorization is an important security concern in cloud computing environments. a POST request from the authorized user on the volumes resource would create a new volume. a DELETE request on the volume resource by an authorized user would delete the volume . if the user of the service is authorized to do so, and the volume is not attached to any instance .It aims at regulating an access of the users to system resources.

**Cloud**

.The cloud monitors contain contracts used to automatically verify the implementation . A cloud developer uses IaaS to develop a private cloud for her/his organization that would be used by different cloud users within the organization. In some cases, this private cloud may be implemented by a group of developers working collaboratively on different machines. We use Django web framework to implement cloud monitor and OpenStack to validate our implementation.

**Admin**

the cloud administrator using Keystone and users or user groups are assigned the roles in these projects. It deﬁnes the access rights of the cloud users in the project. A volume can be created, if the project has not exceeded its quota of the permitted volumes and a user is authorized to create a volume in the project. Similarly, a volume can be deleted, if the user of the service is authorized to do so, and the volume is not attached to any instance, i.e., its status is not in-use.

**REST API:**

REST is an acronym for **RE**presentational **S**tate **T**ransfer. It is an architectural style for **distributed hypermedia systems**.By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components. Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client. Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests. By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components. REST is defined by four interface constraints: identification of resources; manipulation of resource through representations; self-descriptive messages; and, hypermedia as the engine of application state.The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting. REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented.