Assignment no 3

Title

Programming of project functions as per UML diagram mention in earlier submission.

UML Sequence Diagram

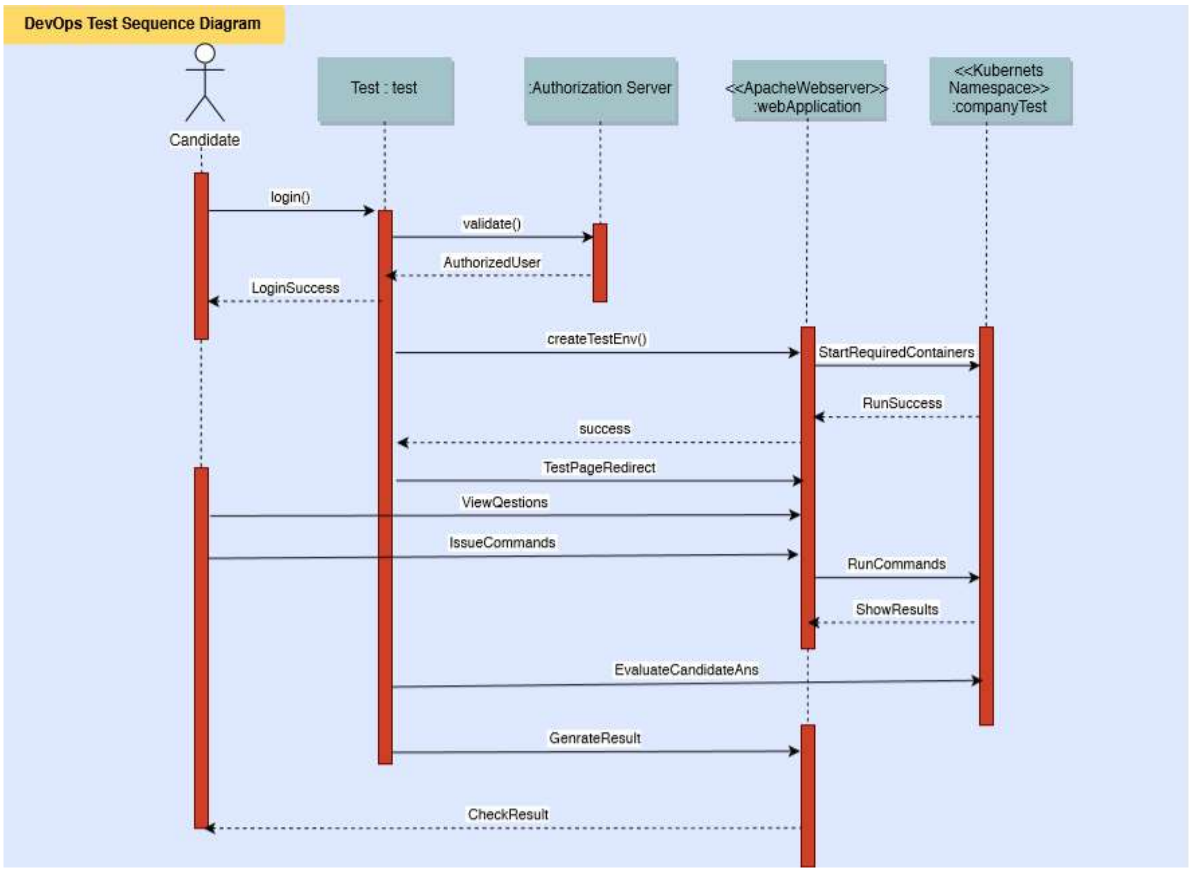

Fig:1. Sequence Diagram

Lifeline represents typical instances of the components or classes in our system. Messages are shown as arrows. They can be complete, lost or found; synchronous or asynchronous; call or signal. Activate is used to denote participant activation. Once a participant is activated, its lifeline appears. Objects are model elements that represent instances of a class or of classes. Classes in UML show architecture and features of the designed system. Actor specifies a role played by a user or any other system that interacts with the subject.

Candidate logins to the system and lifeline gets activated. Candidate can give exam using the test instance. The Authorization server issues commands given by the candidate to actual running container. The candidate and the actual running container are isolated from each other. They communicate using a webserver. After each completion of task as per the question, the page is redirected to the next question. Once the test is completed. A request for test generation is passed to Company test portal to generate the result.

UML State Diagram

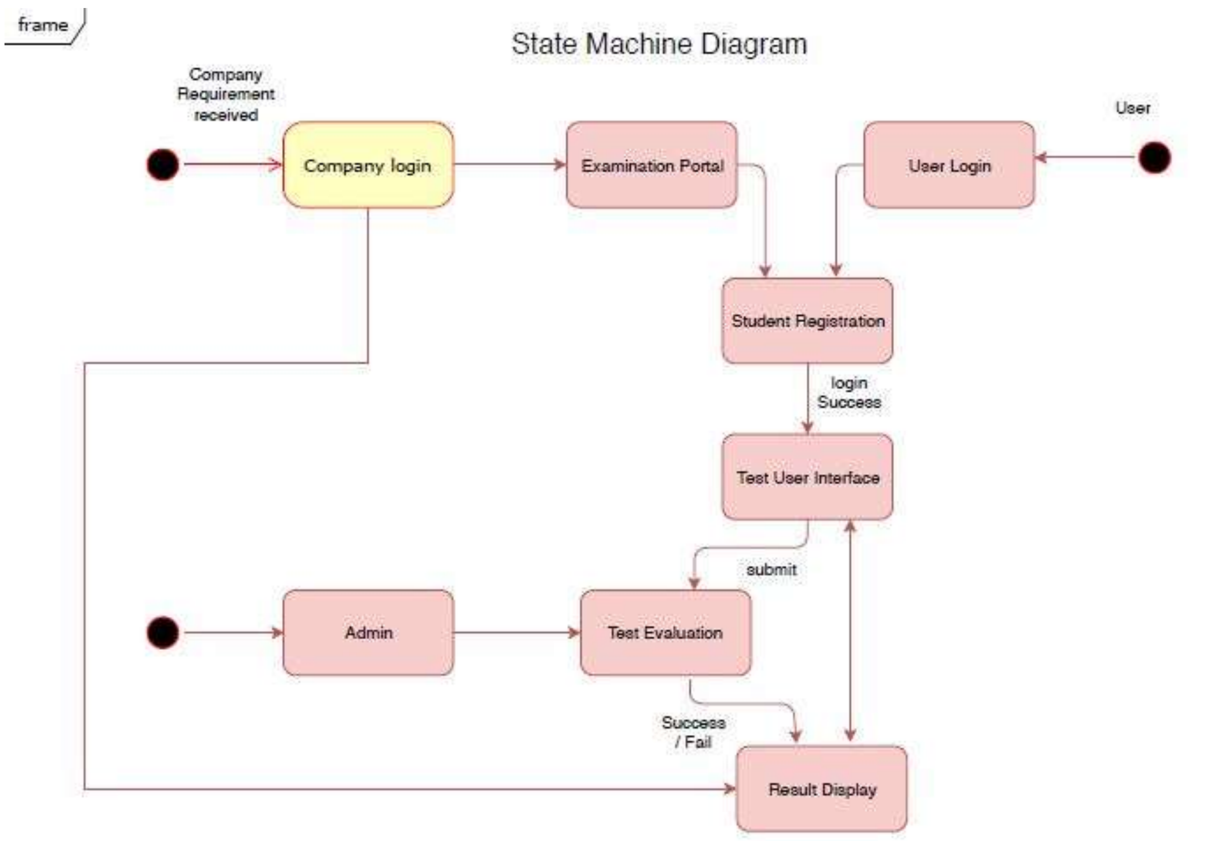

Fig:2 State Diagram

A state is a condition during the life of an object during which it satisfies some condition, performs some activity, or waits for some external event. A start state is the state that a new object will be in immediately following its creation. An end state is a state that represents the object going out of existence. A transition is a relationship between two states indicating that an object in the first state will perform certain actions and enter the second state, when a specified set of events and conditions are satisfied.

Candidate logins to the framework and lifeline of object gets initiated. Candidate can give test utilizing the test occurrence. The Authorization server issues directions given by the candidate to real running compartment. The candidate and the genuine running compartment are detached from one another. They convey utilizing a webserver. After every fulfillment of undertaking according to the inquiry, the page is diverted to the following inquiry. When the test is finished. A solicitation for test age is breezed through to Company test entryway to produce the outcome.

UML Deployment Diagram

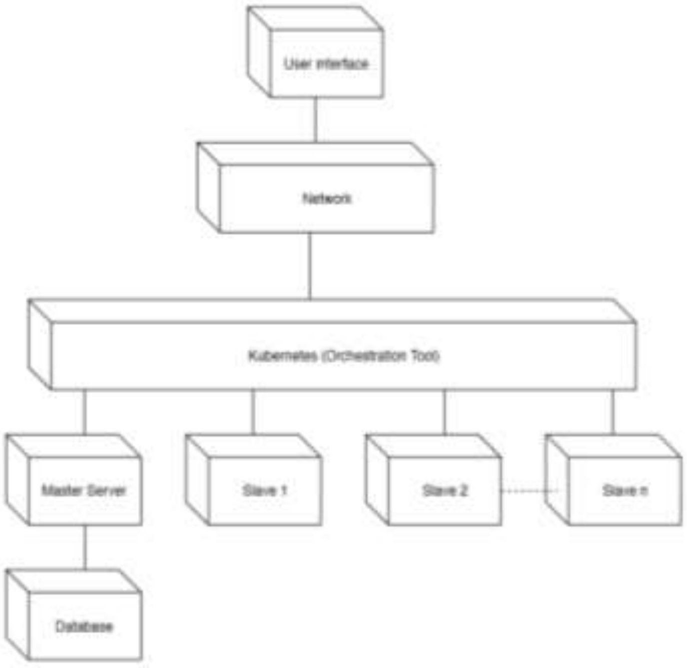

Fig:3. Deployment Diagram

Nodes represent either hardware devices or software execution environments. They could be connected through communication paths to create network systems of arbitrary complexity. Component represents a modular part of a system. A component defines its behavior in terms of provided and required interfaces. Dependency relationship is a relationship in which one element, the client, uses or depends on another element, the supplier.

Master node: Runs multiple controllers that are responsible for the health of the cluster, replication, scheduling, endpoints (linking *Services* and *Pods*), Kubernetes API, interacting with the underlying cloud providers etc. Generally it makes sure everything is running as it should be and looks after *worker nodes*.

Worker node (minion): Runs the Kubernetes agent that is responsible for running *Pod* containers via Docker or rkt, requests secrets or configurations, mounts required *Pod* volumes, does health checks and reports the status of *Pods* and the node to the rest of the system.

Pod: The smallest and simplest unit in the Kubernetes object model that you can create or deploy. It represents a running process in the cluster. Can contain one or multiple containers.

Deployment: Provides declarative updates for *Pods* (like the template for the *Pods*), for example the Docker image(s) to use, environment variables, how many *Pod* replicas to run, labels, node selectors, volumes etc.

DaemonSet: It's like a *Deployment* but instead runs a copy of a *Pod* (or multiple) on all (or some) nodes. Useful for things like log collection daemons (sumologic, fluentd), node monitoring daemons (datadog) and cluster storage daemons (glusterd).

ReplicaSet: Controller that ensures a specified number of *Pod* replicas (defined in the *Deployment*) is running at any given time.

Service: An abstraction which defines a logical set of *Pods* and a policy by which to access them (determined by a label selector). Generally it's used to expose Pods to other services within the cluster (using targetPort) or externally (using NodePort or LoadBalancer objects).