**MULTI-TENANT APPROACH DOCUMENT**

**Multi**-**tenancy** means that a single instance of the software and its supporting infrastructure serves **multiple** clients.

Each client’s shares the software application with different **database.**

**Advantages:**

* Multi-tenant database provides is that the application is shared.
* Through centralizing resources, all customers are maintained at the same version, reducing the overhead an application provider would otherwise encounter when maintaining multiple versions of an application across deployments.

Attendance Portal

**Client 1**: GSS **Client 2:** KITE **Client 3:** CAS

Created separated database for all 3 clients i.e for GSS, KITE and for CAS.

When a client tries to login to portal, tenant-id will be identified to find database. Using that database authorization will happen. And their punch entry will be inserted into respective database.

* A tenant-id will be assigned to all clients.
* All http requests to the system should have tenant id, else it will be marked as a bad request in the tenant filter.
* Tenant-id will be stored in the client session.
* Before session or transaction layer a tenant routing data source will be invoked to determine database for the client request.

Database request will be triggered on logging into the portal and on putting punch entry. For each action data will be fetched and updated to the respective clients (KITE, CAS or GSS).

* A

Flow Diagram for the Attendance Portal:

