**Big Data And Cloud Computing**

**(SOFT7011)**

Detailed Design and implementation of a Big Data and Cloud Application

*Name*: Ravali Billakanti

*Student ID*: 19132818

*Subsystem Taken*: Staff of Disease Controlled Area

**Detailed design of API:**

**AREA**

1. Endpoint: /area/add-area

Method: POST

Payload: {

areaName: areaName,

latitude: latitude,

longitude: longitude,

population: population,

area: area

}

Response: {

message: "Area add Success."

areaData: {

\_id: ObjectId("62666631235f5afe8a4d025a")

}

}

1. Endpoint: /area/add-sub-area

Method: POST

Payload: {

areaId: areaId,

name: name,

lat: lat,

long: long,

population: population,

areaSqKm: areaSqKm,

infected: infected,

}

Response: {

message: "Sub Area add Success."

areaData: {

\_id: ObjectId("62666631235f5afe8a4d0351")

}

}

1. Endpoint: /area/remove-area

Method: POST

Payload: {

areaId: areaId

}

Response: {

message: "Area delete Success."

}

1. Endpoint: /area/getSubArea

Method: POST

Payload: {

\_id: subareaId

}

Response: {

message: "list Fetched."

result: {

\_id: ObjectId("62666631235f4afe8aa95d23"),

areaId: ObjectId("62666631235f5afe8a4d025a"),

name: "subarea1",

lat: .12,

long: .54,

population: 100000,

areaSqKm: 200,

infected: 1050,

}

}

1. Endpoint: /area/markInfected

Method: POST

Payload: {

subareaId: subareaId

}

Response: {

message: "Marked as infected."

}

1. Endpoint: /area/list

Method: GET

Payload: {

\_id: subareaId

}

Response: {

message: "list Fetched."

result: [

{

\_id: ObjectId("62666631235f4afe8aa95d23"),

name : “area1”,

lat : .41,

long: .54,

population: 200000,

areaSqKm: 300,

},

{

\_id: ObjectId("62666631235f4afe8aa95d54"),

name : “area2”,

lat : .40,

long: .54,

population: 205000,

areaSqKm: 400,

}

]

}

**USERS**

1. Endpoint: /auth/login

Method: POST

Payload: {

email : “user@email.com”,

password : password

}

Response: {

message: "Login Success."

result: {

\_id: ObjectId("62666631235f5afe8a448e68a"),

firstName : “User1”,

}

}

1. Endpoint: /auth/add-staff

Method: POST

Payload: {

staffName : “User1”,

staffType : 1/2,

Email : “user@email.com”,

password : passwordHash

}

Response: {

message: "add staff success."

result: {

\_id: ObjectId("62666631235f5afe8a448e68a")

}

}

1. Endpoint: /auth/remove-staff

Method: POST

Payload: {

\_id: ObjectId("62666631235f5afe8a448e68a")

}

Response: {

message: "remove staff success."

}

1. Endpoint: /auth/staff-list

Method: GET

Response: {

message: "list Fetched."

result: [

{

\_id: ObjectId("62666631235f4afe8aa95d23"),

name : “staffname”,

type:2

},

{

\_id: ObjectId("62666631235f4afe8aa95d54"),

name : “staffname”,

type:2

}

]

}

**INFECTION DATA**

1. Endpoint: /data/saveInfData

Method: POST

Payload: {

area\_id :area\_id,

sub\_area\_id:sub\_area\_id,

positive:positives,

negative:negatives,

suspected\_infection:suspected\_infection,

in\_treatment:in\_treatment,

first\_dose:first\_dose,

second\_dose:second\_dose,

inconclusive\_results: inconclusive\_results

}

Response: {

message: "add data success."

result: {

\_id: ObjectId("62666631235f5afe8a433e64b")

}

}

Endpoint: /data/getInfData

Method: POST

Payload: {

area\_id :area\_id,

sub\_area\_id:sub\_area\_id,

positive:positives,

negative:negatives,

suspected\_infection:suspected\_infection,

in\_treatment:in\_treatment,

first\_dose:first\_dose,

second\_dose:second\_dose,

inconclusive\_results: inconclusive\_results

}

Response: {

message: "add data success."

result: [{

\_id:ObjectId("62666631235f5afe8a433e64b")

area\_id :ObjectId("62666631235f5afe8a433356"),

sub\_area\_id:ObjectId("62666631235f5afe8a43456eb"),

positive: 1000,

negative:200,

suspected\_infection:400,

in\_treatment:400,

first\_dose:100,

second\_dose:300,

inconclusive\_results: 300

}, {

\_id:ObjectId("62666631235f5afe8a433e32f")

area\_id :ObjectId("62666631235f5afe8a433356"),

sub\_area\_id:ObjectId("62666631235f5afe8a43456ed"),

positive: 1030,

negative:200,

suspected\_infection:300,

in\_treatment:400,

first\_dose:300,

second\_dose:100,

inconclusive\_results: 500

}]

}

1. Endpoint: /data/dashboard

Method: GET

Response: {

message: "list Fetched."

result:

{totalCases :'648',

vaccinated: '658',

recovered: '455',

inTreatment: '110',

areasAffected: '33',

centers: '18'

} }

1. **Quality Requirement of COVID-19 disease control Centre Staff:**
   1. **Performance:**

Disease control center staff has only four functions that are efficient in performance.

* + 1. **Setup new disease control center staff:**

The disease control center admin need to create new staff member who have limited access to view covid-19 test results and vaccination details for managing the data and setting the area as containment zone for local restriction.

* + 1. **Setup new regions and sub-regions:**

This function allows the admin to create new region and sub-region for identifying the places with community spread and setting those areas as containment zone. System should allow create region without any delay between 5-10 seconds.

* + 1. **View covid-19 and vaccination statistics:**

The admin and staff can see the statistics and graph of the covid positive, covid negative, first dose vaccination, second dose vaccination. The staff should take decision the basis of the given data, whether to make the region as containment or not. The data should load efficiently within 10 seconds.

* + 1. **Setup regions as containment zone:**

By evaluating the data staff should set a region as containment zone. This will bring local restriction in the area. The process should not take more than 10 seconds to complete setting area as containment zone.

* 1. **Scalability**

As the Disease control center should evaluate more than twenty thousand data a day, the system should the scalable as per demand. Highly scalable cloud servers will use for to meeting the requirement. For handling this much data around thousands of staff accounts will be created by the admin.

* 1. **Reliability:**

The computer systems used by a Disease control center staff are able to work 24x7. The mean time failure is less than 30 minutes and failure time a year is also less than 150 hours.

* 1. **Security and privacy:**

The Disease control center staff will have access to sensitive data of users with covid-19 infection. The recruitment of staff should be strict for avoiding leakage of the data.

* Https will be used to for improving data transfer security between backend and frontend.
* For accessing backend access token will be required.
* Sensitive data like password will be encrypted using SHA256 algorithm.
* The staffs must be very well qualified and trained.
  1. **Usability:**

The application must meet all the usability aspects like effectiveness, efficiency and the overall satisfaction of the user.

**Architecture Design of Disease control Centre**

**Staff application of disease control Area**:

The microservice architecture of the disease control center has 4 services as shown in figure 1:

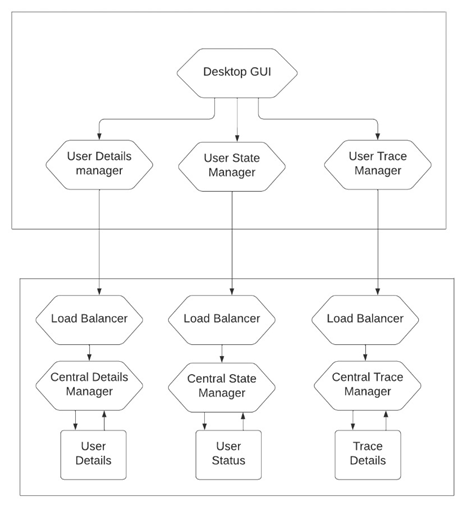
* Microservice for covid-19 statistics view and setting up containment zones.
* Microservice for the Staff Authentication.
* Microservice for creating staff account.
* Microservices for creating region and sub region.

And for the storage there are 3 databases:

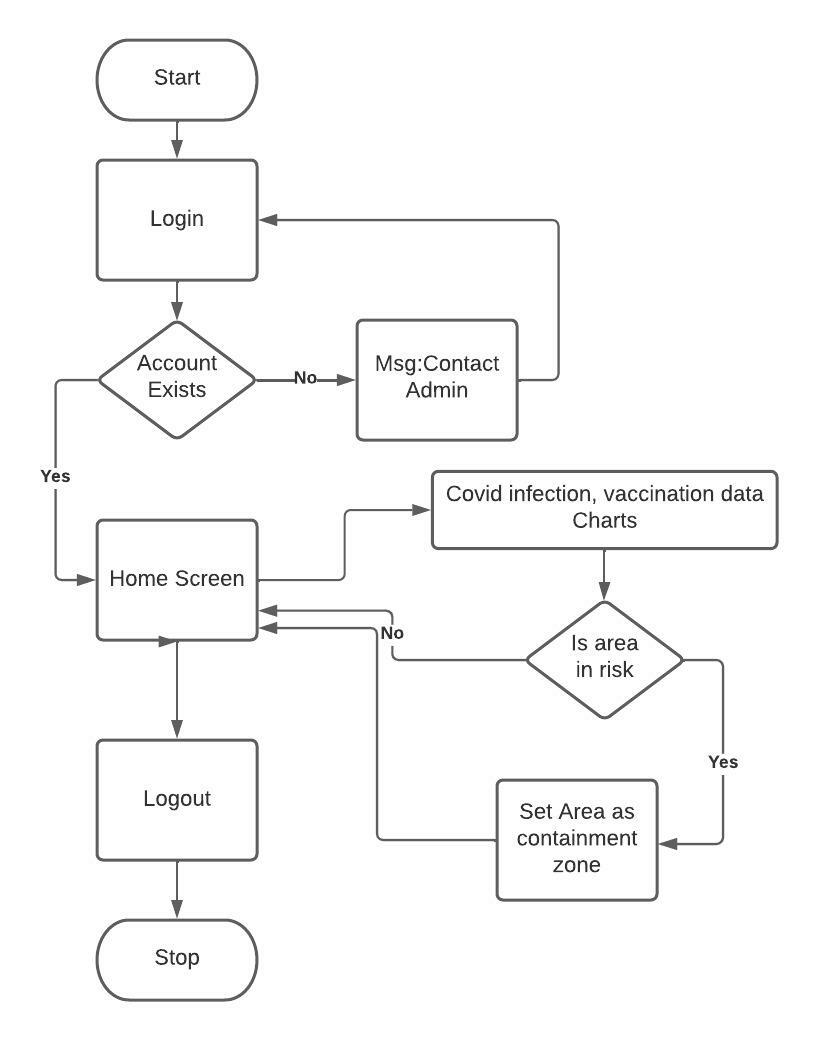
* Covid-19 infection database.
* Staff account database.
* Area details database.

The system need to handle huge amount of data. So the design is account to the quality and performance required. The micro service architecture can ensure the load balancing and reduce down time.

**Figure 1: Architecture design of Staff desktop application**



The flow chart of this staff desktop application is as follows

****

**Specification of Micro-services**

1. Disease control center Staff Authentication:

|  |  |
| --- | --- |
| Service Name | Staff authentication |
| Description | The service allow staffs to login to the system |
| Provided Service | Login to the disease control center staff system. |

1. Covid-19 statistic view:

|  |  |
| --- | --- |
| Service Name | Covid-19 statistic view |
| Description | The service allow User to analyse data though graphs and set areas as containment zone |
| Provided Service | Set areas according to data |

1. Create Staff accounts:

|  |  |
| --- | --- |
| Service Name | Create new staff account |
| Description | The admin account should have the ability to create new staffs |
| Provided service | Create new staffs using email |

1. Create new Areas and sub areas:

|  |  |
| --- | --- |
| Service Name | Create new Areas and sub areas. |
| Description | The Admin have access to create to area and sub area. |
| Provided service | Create new areas and sub areas with area location details. |

**Specification of database**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Database name** | | **Covid 19 infection Data** | |
|  | Description | | The database stores the data of covid19 positive and negative result. | |
|  | Type | | non-relational database | |
|  | Design | | Results, Isolation Data, Crown-pass users, User Vaccination Details | |
|  | Table-Result | | Fields: Crown-pass User Id, date, result | |
|  | Crown-pass Users | | Fields: id, name, address, phone, age, gender, national Id | |
|  | Table-isolation Data | | Fields: Crown-pass User Id, start Date, end Date | |
|  | Vaccination Details | | Crown pass User Id, Dose Type, Vaccine, Date, Vaccine Staff Id | |
| **Database name** | | **Staff Accounts** | |
| Description | | The database stores Staff account details. | |
| Type | | non-relational database | |
| Design | | DCC Users | |
| Table-DCC Users | | Fields: id, Staff Name, email, password, staff Type | |
| **Database name** | | **Area database** | |
| Description | | The database stores details of area and sub area | |
| Type | | non-relational database | |
| Design | | areas, sub Areas | |
| Table-areas | | Fields: id, name, lat, long, area in SqKm, population | |
| Table-Sub Areas | | Fields: id, name, area Id, lat, long, area in SqKm, population | |