1. **Proposed architecture:**

**High level Design:**

Which ui, Service layer and database with components of master-slave Load balancer.

The same load balancer can also act as API gateway.

Two instances of API can be running. we can have health check for API every 3o mins and check if there are up and running.

We can Selenium Test running on ui and mail can triggered to development team if critical workflows are down.

Logging can implemented can be pulled using Logstash or Kibana to monitor any errors showing up.

Analytics can run on ui to track which component user are using more.

Low level Design and Technologies

Ui –implemented in React. since application is simple and we can configure the third-party components so that vary independently React is selected. Bootstrap was taken for styling components.

Service layer was implemented using Repository patterns and dapper.

Since database is very light micro orm are very good of performance

**Assumption**

assumptions Made

1)Since there is no specific clarification whether reads will be more or write will be more

Assumption is made both are Equal

2)Document will be stored on server and URL will be mapped in Document table

Document table consists of three columns identity column name and URL. Since we have to search by name mapping

indexing is applied on Name columns.

since assumption is made for 50 times lazy loading is not done

if Documents are more than 1000 lazy loading can be considered with type ahead search

3)keyword Table consist of two columns identity, keywordname

maximum size and approx. upper limit of keywords is not given. assumed to be max size 10

since it can be alpla numeric we can store (10 to power (26 aplha + 10 digits)) perumations and combination of keyword

if still more combinations are required, we can go for hashing.

4)Document Keyword mapper to columns keyword foreign key, document foreign key

foreign key constraints are implemented and index on keyword.

Database Sql server

Scripts for table were placed in embed file.

**Only simple positive scenario was handled**

**features left out due to time**

1)pagenation was not handled .

2)Validation Not handled

3)negative workflow not handled .

4)Spinner are not implemented.

<https://youtu.be/I8IPATvWpAw>

**Steps to run the App**

In DellAdminApp folder in cmd run the following commands

Assumption nodes is already installed

1) npm install

2)npm install react

3)npm install react-create-app

4)npm install react-scripts

5)Run Npm start

When you run npm start window launches in browser.

In Api Modify coneection string in apppsettings

"DefaultConnection": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=Admin;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"

Run the scripts in database