Spring BootApplication AggregatorApp\_Site

[Problem statement](#_32pv4mu3nalm)

[Application Architecture](#_hosq0si31t0u)

[DBDesign](#_fldsouc1zte)

[Class details](#_dyw2ncpqrj7)

[Functionalities](#_aowfel9yyuhw)

[● EPIC 1](#_w5luar5rpkc2)

[Note:- Use any Relational DB for the User details](#_6ne84fih9bc5)

[● EPIC 2](#_s4pcbclq0rgt)

[Note:- Use any NoSQL DB for the GloQuora\_Post](#_ovnh5a4ccqm2)

[● EPIC 3](#_gi4rxtxeabml)

[Instructions](#_3dm46okec2xm)

[Sample Data](#_4youtne6n3fy)

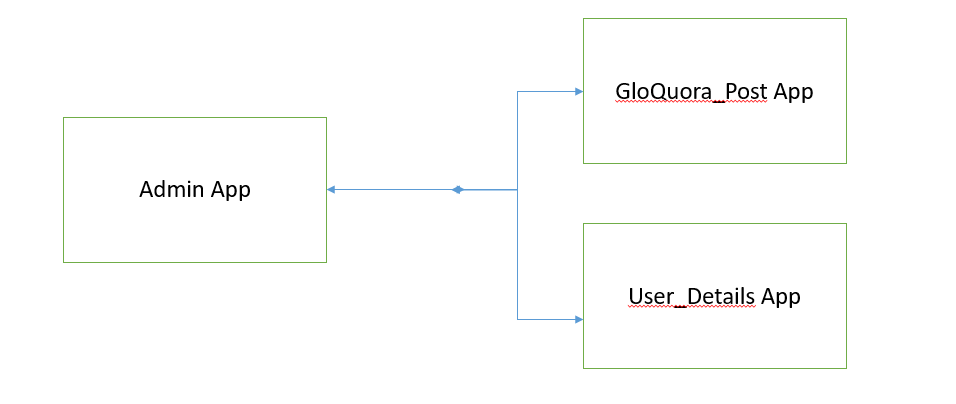
## Problem statement

GloQuora is a community-based questions and answers website. GloQuora app has many features such as registering as a user and posting queries in the GloQuora forums.

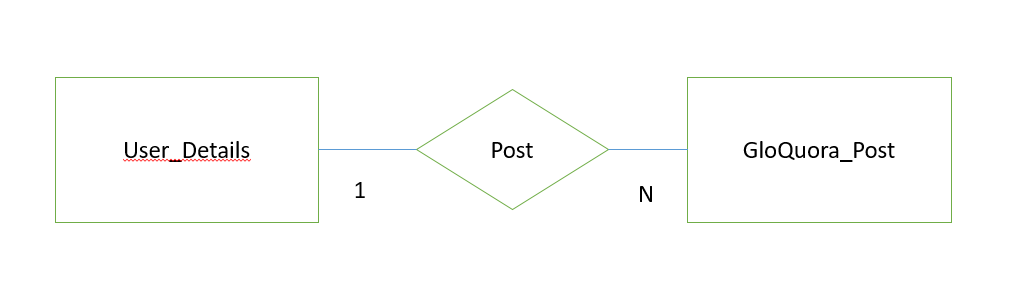
Sam has to create an admin module that generates a report to analyze the GloQuora site based on the users and posts.

Help Sam to design an aggregator app for admin users that consumes the data from Userdetails App and GloQuoraPost App which are two separate services.

## Application Architecture



## DBDesign



## Class details

**Class - User\_Details**

UserId

Name

Username

Email

Address

Phone

Geo

Company

**Class-Company**

Name

Location

**Class- Address**

Street

City

**Class- Geo**

Lat

Long

**Class - GloQuora\_Post**

Userid

Post\_id

Title

Body

## Functionalities

## EPIC 1 User details App

CRUD service for the User details

* + GetAll User details
  + GetSpecificUserDetails
  + AddUser
  + DeleteUser
  + UpdateUser

## Note:- Use any Relational DB for the User details

## EPIC 2 GloQuora\_PostApp

CRUD for the GloQuora\_Post details

* + GetAllGloQuora\_Post
  + Add\_GloQuora\_Post
  + Delete\_GloQuora\_Post
  + Update\_GloQuora\_Post

## Note:- Use any NoSQL DB for the GloQuora\_Post

## EPIC 3 AdminApp

| User Story | User Story | API | Response |
| --- | --- | --- | --- |
| US1 | As an admin, one should be able to get all the details of the GloQuora\_Post sent by a specific user | http://localhost: port/userspost/{UserId} | All posts by specific user 1 must be displayed at the browser console/Postman console. |
| US2 | As an admin, one should be able to get all the posts posted by the users with details of users and their posts. | http://localhost: port/userspost | All his/her posts must be displayed at the browser console/Postman console. |
| US3 | As an admin, one should be able to get all the user names who have posted more than k post.(k can be an integer like 5 or 10) | http://localhost: port/userspost/k | all the user name who has posted more than k posts has to be displayed in the browser console/Postman console. |
| US4 | As an admin, one should be able to get all the company name of the users who have posted at least one post | http://localhost: port/userspost/company | all the company names has to be displayed in the browser console/Postman console which has posted at least one comment |

Epic 4 Security Features

Add the below given requirements to the GloQuora app and provide the required Spring security configuration.

| US5 | Allow the access to http://localhost: port/userspost/\*\* for only the users who have logged in by providing username and password explicitly  (ref Epic 3 ) | use in-memory authentication, with a role as ADMIN. | username = “sam@gloquora.com”, password = “sam@123” |
| --- | --- | --- | --- |
| US5 | While storing the username and email address int the UserApp ( ref Epic1 ) The data need to be encrypted | Use Spring Security's BCryptPasswordEncoder. | Store the Username and Email in an encoded format using Spring Security's BCryptPasswordEncoder. |
| US6 | Only Admin with username John should be able to delete the users. | Secure user delete functionality of the application to be accessible only to the ADMIN with username as "John" | Implement the Method Level Security for the functionality. |

## Instructions

1. Use Spring Boot Framework and Microservice Architecture
2. Use the Controller /Service /Repository/Entity /DTO Pattern
3. Use a centralized registry to register the Services.
4. Use RestTemplate/ Any technique other to Consume the services.
5. Use APIGateway to route the request to GloQuora\_PostApp and UserApp
6. Use appropriate validations of data at the controller
7. Use appropriate exception management.
8. Use Logger appropriately
9. Use Create appropriate testing modules at Controllers
10. Use Swagger API to create a documentation

## 

## Sample Data

User\_Details Json Data format

{

"userId": 1,

"name": "Graham",

"username": "Bret",

"email": "Sincere@april.biz",

"address": {

"street": "Kulas Light",

"city": "Gwenborough",

"geo": {

"lat": "-37.3159",

"lng": "81.1496"

}

},

"phone": "6546546456",

"company": {

"name": "Romaguera-Crona",

"location": "Multi-layered client-server neural-net",

}

}

GloQuora\_Post Json Data format

{

"userId": 1,

"post\_id": 1,

"title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",

"body": "quia et suscipit\nsuscipit recusandae consequuntur expedit"

}