Software Design and Architecture Document

Microservice Architecture:

- There are 2 microservices :

i)Profile

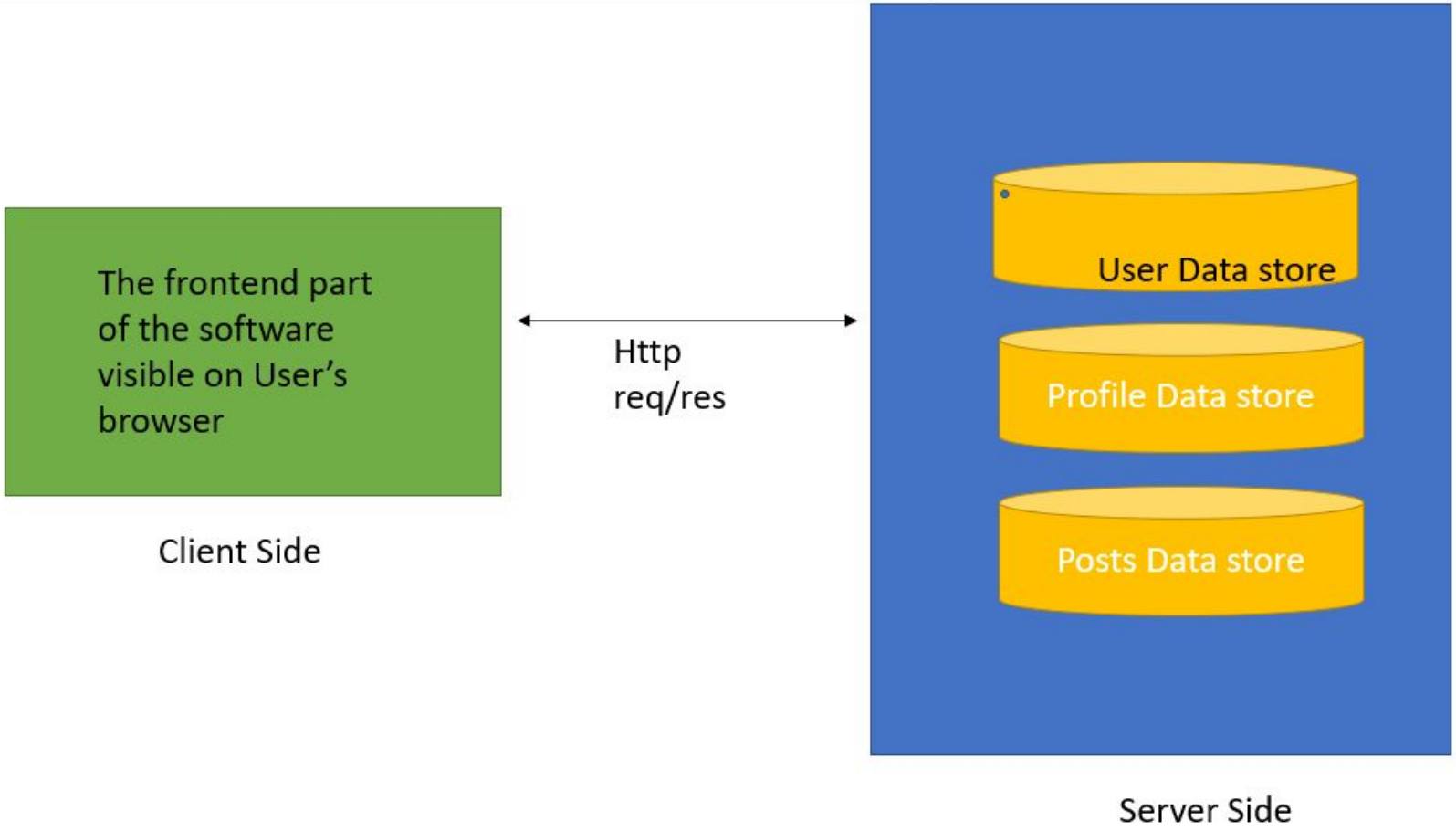
ii)Posts

Quality Attributes:

i) Usability

ii) Security

Physical Or Deployment View

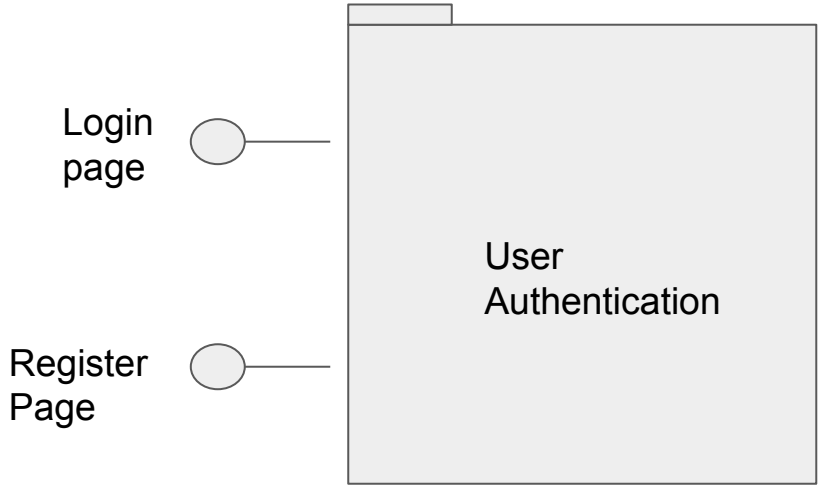


Description

- The client side will be able to open the URL of the web app after its deployment on their respective browser and see the frontend part.
- They will interact with the server side with clicking on buttons present on the respective pages and will be able to make get ,post,delete,put request to server where the database is present.
- Accordingly as they register themselves a post request will be send with all their details ,on the server side , in the database they will be added . Then server side will send them the url page to their dashboard along with the token which will be make their route private.
- Like this interaction will keep on happening between server and client.
- Justification based on User requirement :
- Hence user will get an interactive software where they will be able to interact with the software with GUI and keyboard (as mentioned in the interface requirement of SRS).The response will be given by the server as soon as the request is made making the software rich on performance (satisfying the performance requirement of SRS).

User Authentication (there in both microservices)

- User database contain the data of all the users registered so far on the app.
- So basically it will perform 2 functionality Login , Register.



For the inside view of microservice I have used 4+1 view model.Logical View and Process view.

Logical View

Class

User
Class

*Name : String,
*Email : String ,
Unique
*Password : String
Date :
Default(Date.Now)

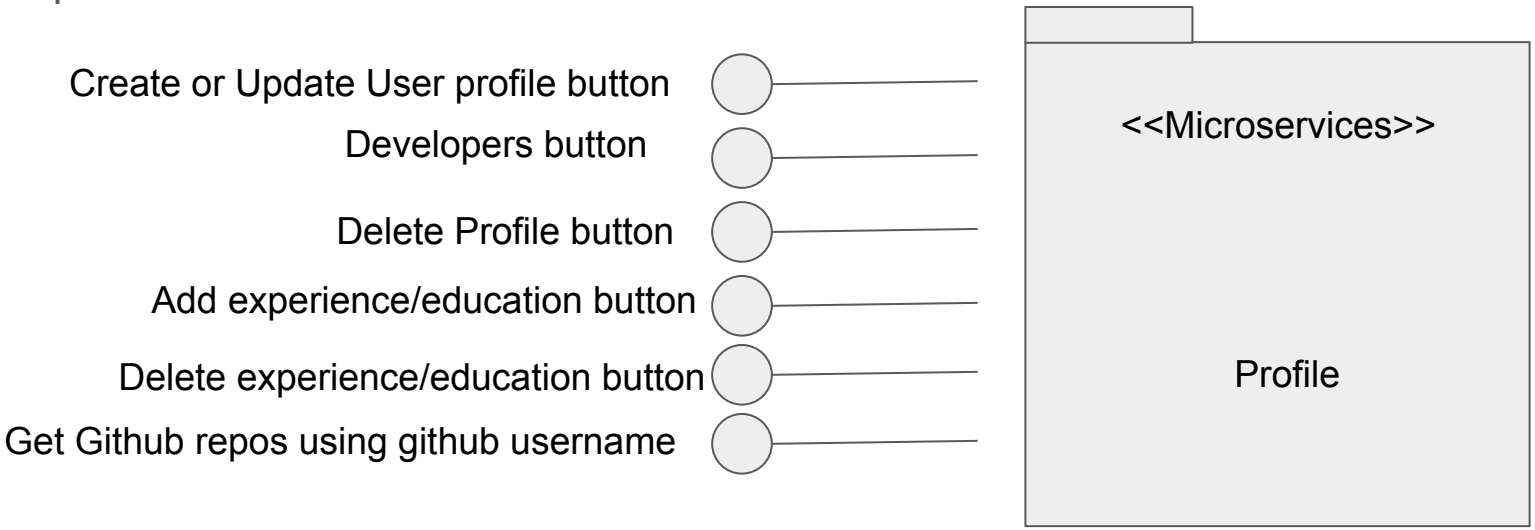
•Here “*” denotes that field is compulsorily required. Mongodb will give every user an object_id attribute as well.

Justification based on SRS of Logical view

Logical view includes the database models which is one of the most important part of software.All the user login details will be there in it and user will be able to open its account whenever hi likes providing the login details. Hence login functionality will be satisfied(as mentioned in SRS). If password provided does not match the password in database , no login will happen hence giving security to the user (non functional requirement in SRS).He will be able to register himself in the database as well. (Register functionality of SRS)

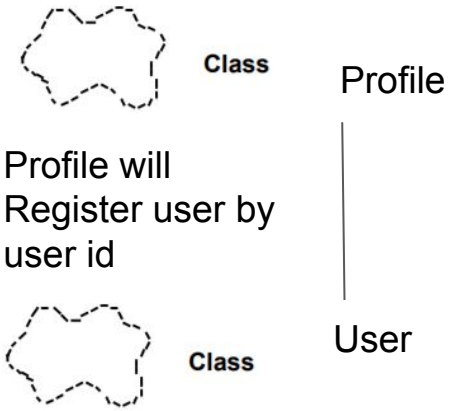
Profile Microservice Design

Profile database contains all the data about the profiles created.First the dashboard page of a user opens on login/register. It has to perform multiple functionality.Like create,update,delete profile. The profile has fields like bio,skills,job status company.Then there are buttons to add and delete education and job experience. Also on providing github username their repos display on their profile. Developers button is to see all the developers at once.



For the inside view of microservice I have used 4+1 view model.Logical View and Process view.

Logical View



User : user will be identified by the user id which will be created by User db itself for every user registered.
Company : String
Website : String
Location: String
*Skills : String
Bio : String
GitHub Username : String
Date : Default(Date.Now)
Experience :
Education :

*Title : String
*Company : String
Location: String
*From : Date
To : Date
Current: Boolean
Description : String

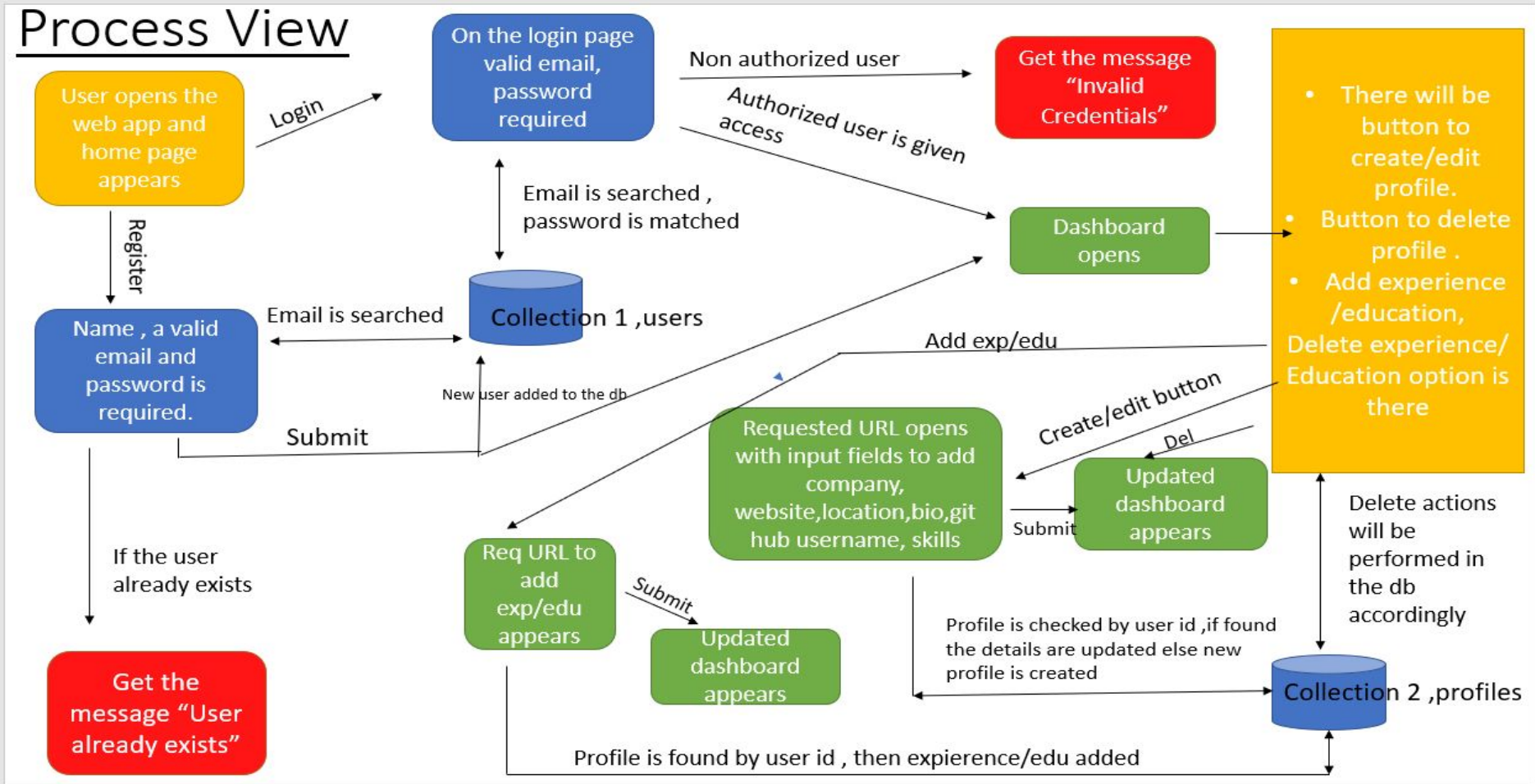
*School : String
*Degree : String
*Subject : String
*From : Date
To : Date
Current: Boolean

Justification based on SRS of Logical view(focus on quality)

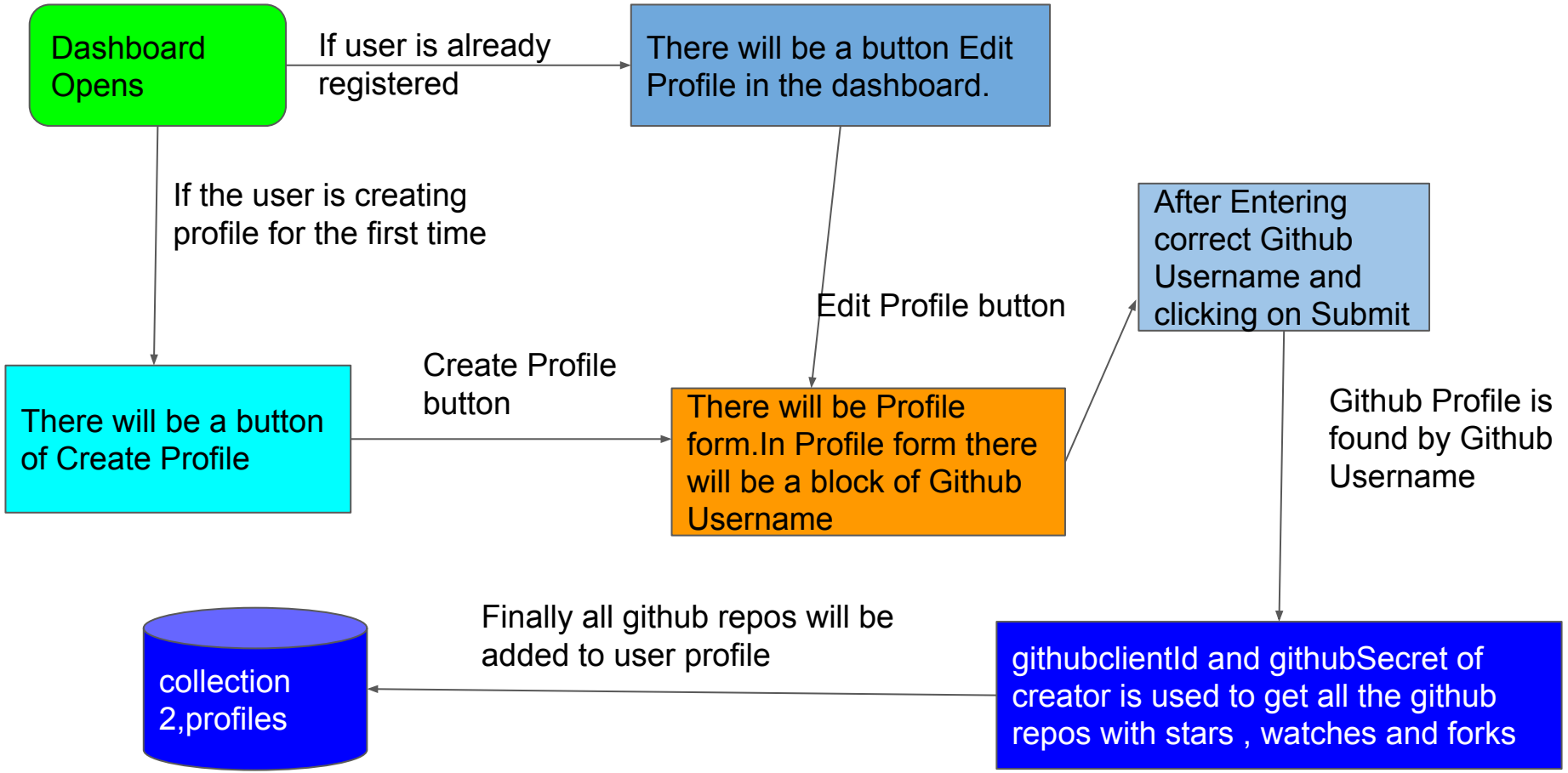
There will be user’s profile saved in the database model profile and he will be able to update in whenever he wants or delete the existing one from database and create new one. Hence functional requirement of profile,making and updating satisfied.

As we can see only profession/career related fields are there this will increase appropriate recognizability (usability quality)aspect of the software as mentioned int SRS.
There will be current option for edu/exp. And its made boolean data type. Hence clicking on it in frontend will make people know its person’s current job and user won’t have to mention it in his job description . Hence increasing the operability.

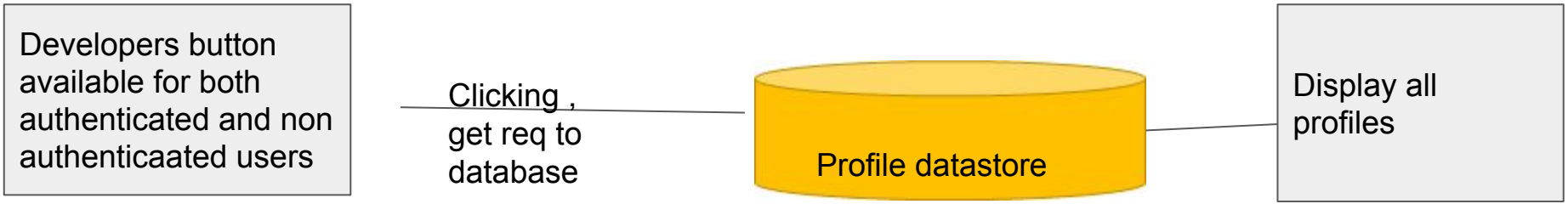
Process View of User authentication and profile microservice design



To see github repos on profile page



To view all the profiles



Description

- First the user opens the home page on their browser. If already registered they login , else register themselves first.
- On registering with valid email , name and password , user is given authenticated access to their dashboard and they are added to the user collection of the database.
- Once registered, the dashboard opens with option to create profile . User clicks on the create profile button .
- Required URL is opened and the user is asked to give the details . On clicking submit , new profile is created in the profile collection and updated dashboard appears on the screen.
- When user logins with correct email id and password , dashboard opens along with options to edit profile , add experience , add education .
- On clicking on these buttons user is given input fields to fill the form . On submitting it the profile is updated on the profile database and user gets the updated dashboard.
- User also has delete profile , delete experience , delete education options available . On clicking these options database gets changed accordingly and updated dashboard appears.
- All the profiles can be viewed by both authorized and non authorized users.

- After entering the correct github username in the profile form github profile of the user is found using the githubClientId and githubSecret of the creator.
- After finding the github profile of the users all the repositories present in the github profile are directly added to profile collections with their stars , watchers and forks.

Justification based on SRS and focus on quality attribute

•As we can see in the process view maximum number of functional requirements mentioned in the SRS are included over here. Login, register ,profile making updating,viewing each other's profile.

•The first quality attribute usability involves appropriatenes recognizability.Here adding functionalities like github repos on the profile page of the user shows how dedicated this app is to developers.Apart from this only professional profile can be created as we can see.

•All the functionalities being performed are having alerts generated and all the options like edit profile, delete profile , add edu/exp these appear together hence it increases the operability of the software.

•Alerts will help in user error protection as well which helps in increasing the usability.On deleting file user will be asked for confirmation.

•Second quality we have focused upon is the security.

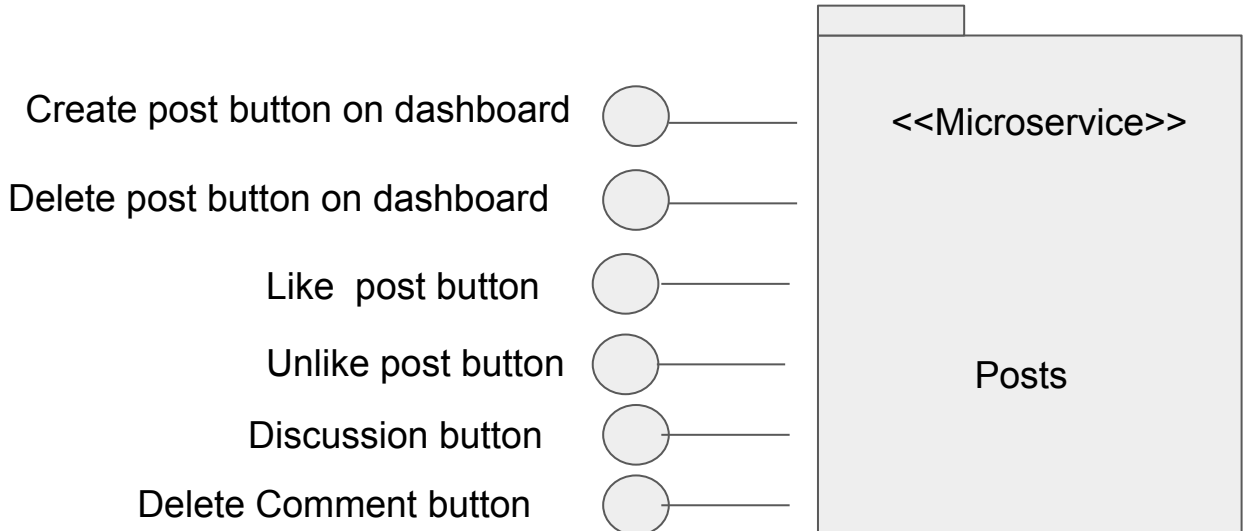
We have only provided user with correct email id password to get access.Hence preventing unauthorized action.So here confidentiality and integrity aspect of the software is maintained.

•All the warning given upon wrong email/password would be “invalid credential” not mentioning whats wrong hence increasing security again.Apart from this bcryptjs, jwt will be used for security as mentioned in SRS.

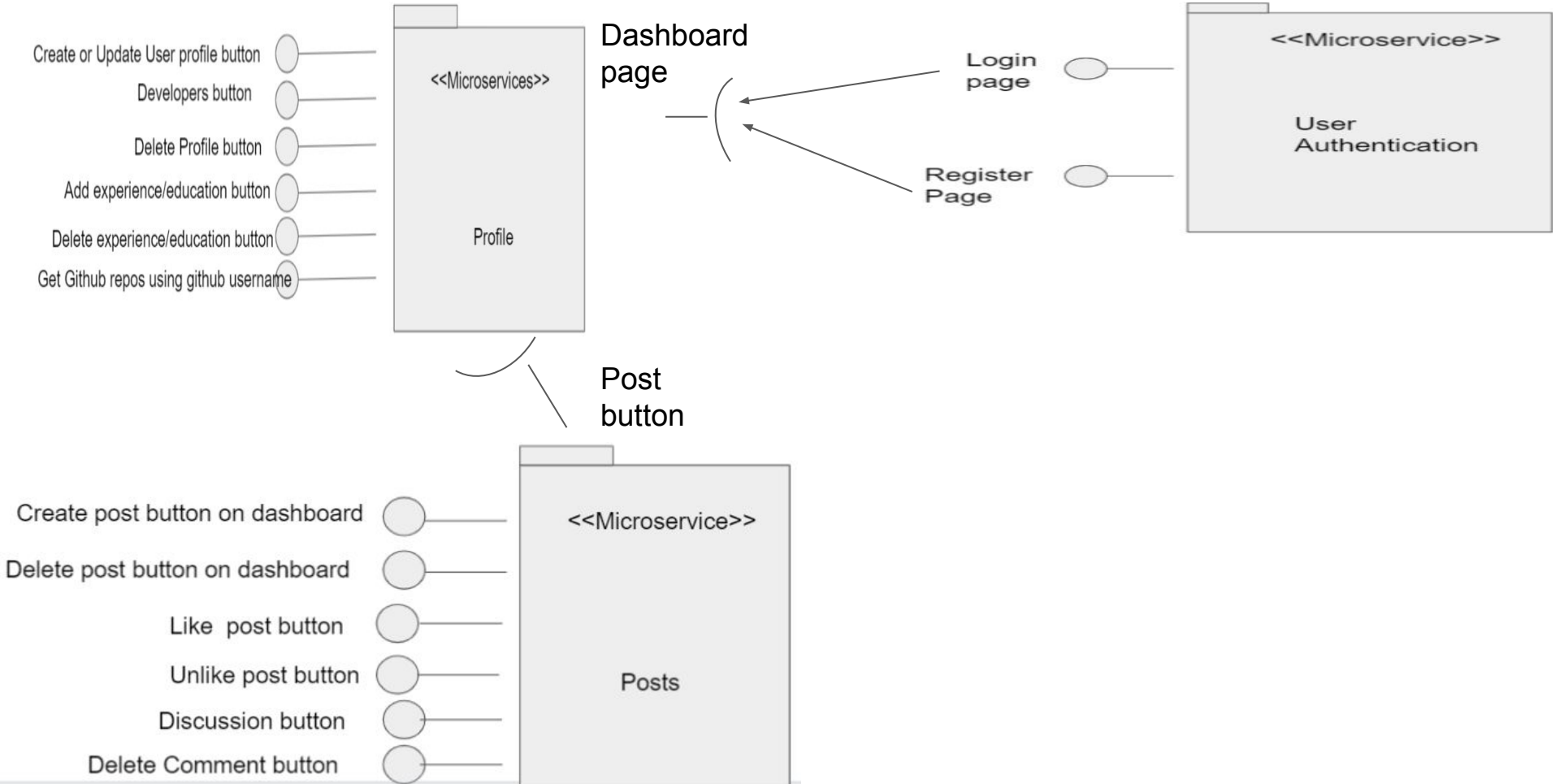
Posts microservice design

•Posts database contain all the information related to post made by users.

•It has functionality creating a post , deleting posts and like/replying on other people's posts.



Design Model



For the inside view of microservice I have used 4+1 view model.Logical View and Process view.

Logical View

Post Class

Class

Posts get saved by user id

Class

User class

User can create post and do comment on others posts. User can also like or dislike others profile.
user:Schema.Types.ObjectId
*text:String
name:String
avatar:String
likes:
comments:
date:Date(default Date.now)

user:Schema.Types.ObjectId

user:Schema.Types.ObjectId
*text:String
name:String
avatar:String
date:Date(default Date.now)

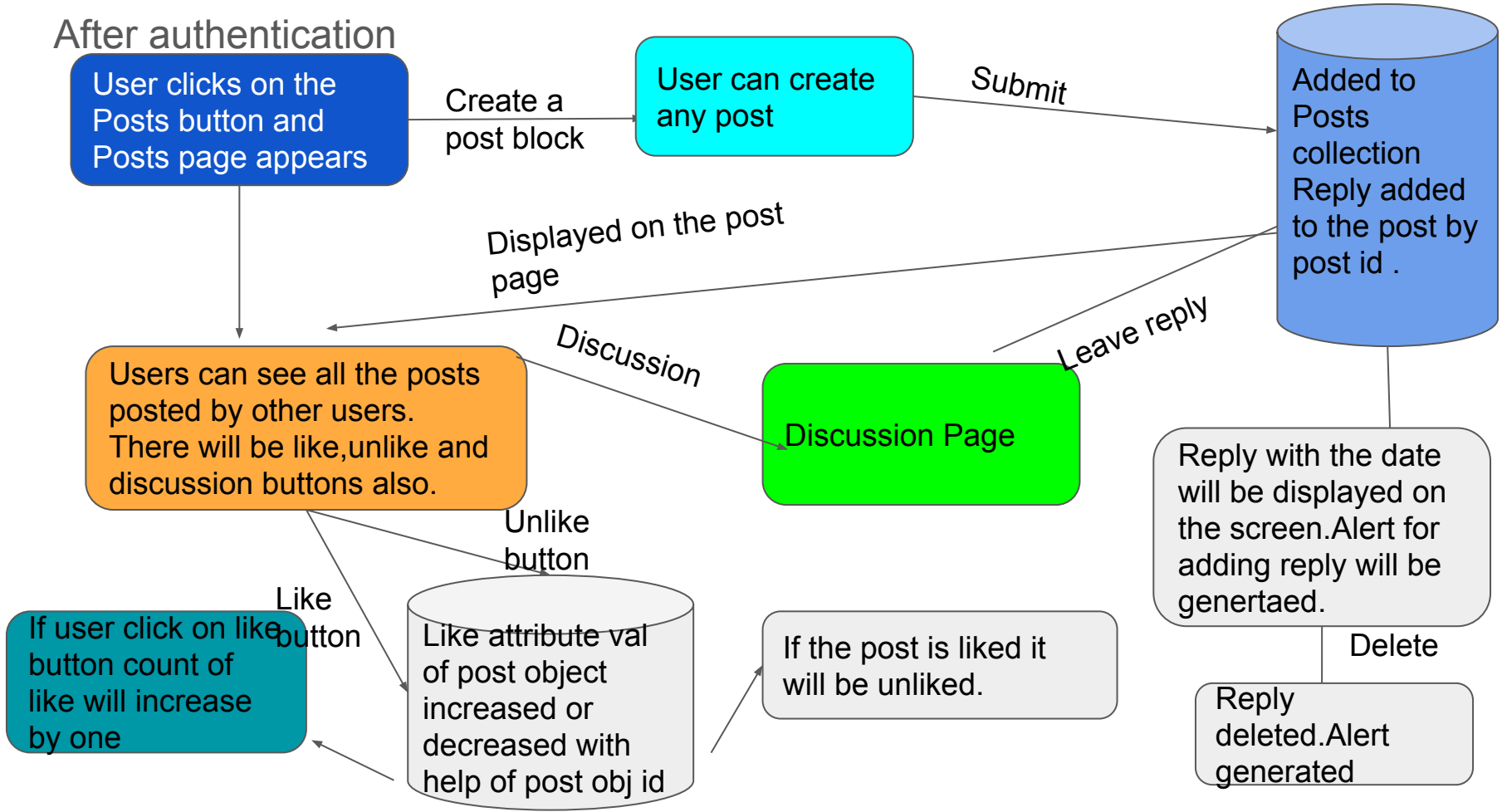
Justification based on SRS

We have created a class for posts that is a separated datastore for storing posts of all the user along with user id. All the post are object with object id which include attribute like and unlike and reply.

Hence satisfying the last 2 functional requirements in SRS i.e.post like and comment.

Process View

After authentication



Description

Authenticated user clicks on the post button and gets navigated to the post page.

- User can create any post there and on submitting it gets added to post collection in database.
- User can like someone’s post by clicking on like button. The count of like on the post will increase by one.
- User can unlike someone’s post if he has liked the post earlier otherwise unlike will not work. After clicking on unlike count of like on the post will decrease by one.
- User can create any post by entering the post and clicking on submit button.After this post will be displayed on the screen with date on which it is posted.
- User can also delete his post after creating it by just clicking on delete button present below the post.

- User can also reply on other’s posts by clicking on Discussion button. After entering the comment and clicking on submit button comment will be displayed on the screen with the current date.
- The count of Discussion on the post will increase by 1.
- User can delete his comment also by just clicking on delete button present below the comment.
- After deleting the comment the count of Discussion on the post will decrease by one.
- All the comments and posts will be added to posts collection.

Justification based on SRS focus on quality attribute

As we can see all the points mentioned in SRS for posts,reply and comment have been justified through this design.

Here also we are having alerts for comment added , deleted this will increase usability of the software.(operatibility)

Just on button click like and dislike feature are happening increasing the usability of the software.

Availability to discuss , to reply on posts , like it empowers the users to express their emotions , hence increasing the usability.

All the post route are authenticated. The user can delete only their posts and replies.Unlike only their liked posts.Increasing the security of the system.

Development View

Team Member 1: Pragati	Team Member 2: Sandip
• Planning and design • Creating the Database models • Profile routes to create and update the profile. • Profile routes to add experience and education. • All the necessary post related to backend • Frontend using react(Homepage, Register page, login page). • Testing • Deployment	• Planning and design • Login and Register Module • Profile Routes to delete profile • Profile Routes to delete Experience and Education • Frontend using react(dashboard, profile form, experience form, education form) • All the necessary post related to frontend. • Fix bugs • Deployment

Description

- There are 2 members in this project.
- While we are doing planning and designing together.
- In the backend modules(doing in express and node) are divided who will do what.
- Similarly frontend (doing in react and redux) is also divided.
- Member 1 will do testing , member 2 will fix bugs.
- Deployment will be done together.

•Justification based on User requirement(focus on quality attribute) :

All the functional requirements mentioned are covered in the design document and coding will happen accordingly. Usability quality attribute will be fulfilled with react (user interface aesthetics), redux (generating alerts , data flowing from one component to other),most of the user error protection can be done in html5 and css.

There are switch tag in react which increases routing performance hence increasing performance req and usability (non functional req) in SRS.

Use of redux (data flowing from one component to other) will also increasing performance req of SRS.

Bcrypt and jwt library of node will help in security.