PART A. BUILD, DEPLOY, AND RUN A CONTAINERIZED APPLICATION USING GCP.

CSCI – 5410 SERVERLESS DATA PROCESSING

ASSIGNMENT – 2

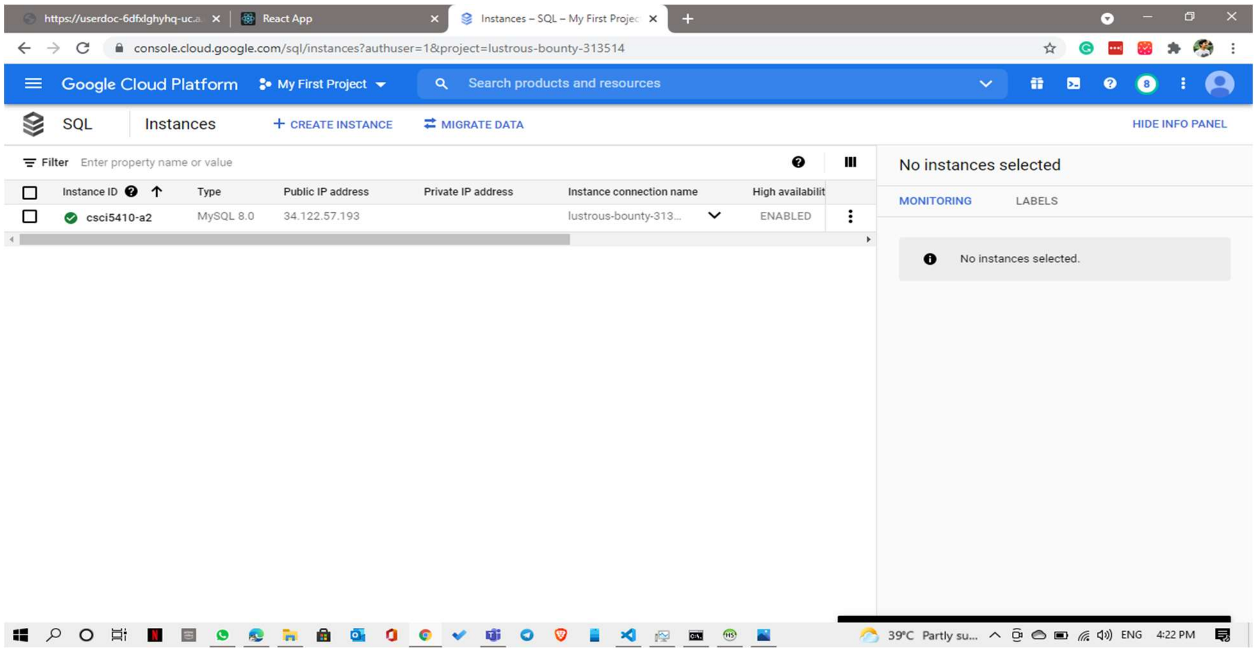
SUBMITTED BY:

DHRUV DOSHI dh72257@dal.ca

GITLAB LINK:

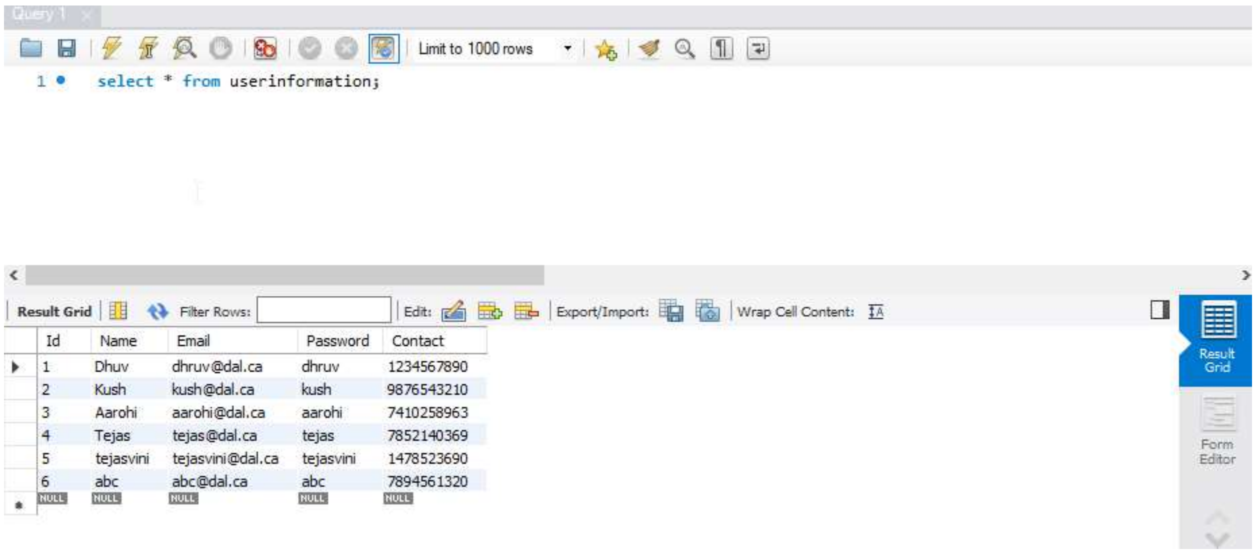
https://git.cs.dal.ca/doshi/dhruv-csci5410-summer/-/tree/master/assignment_2

DATABASE



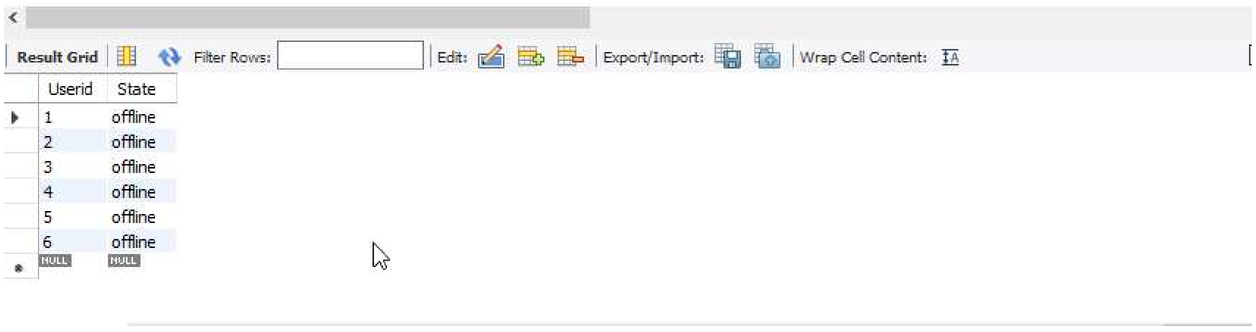
Instance ID	Type	Public IP address	Private IP address	Instance connection name	High availability
csci5410-a2	MySQL 8.0	34.122.57.193		lustrous-bounty-313...	ENABLED

As shown here I made the SQL database in the GCP portal and then with the Ip address and the password logged into SQL Workbench. By writing the SQL query I was able to create two database named user information and user state which had been shown in the screenshots.



```
select * from userinformation;
```

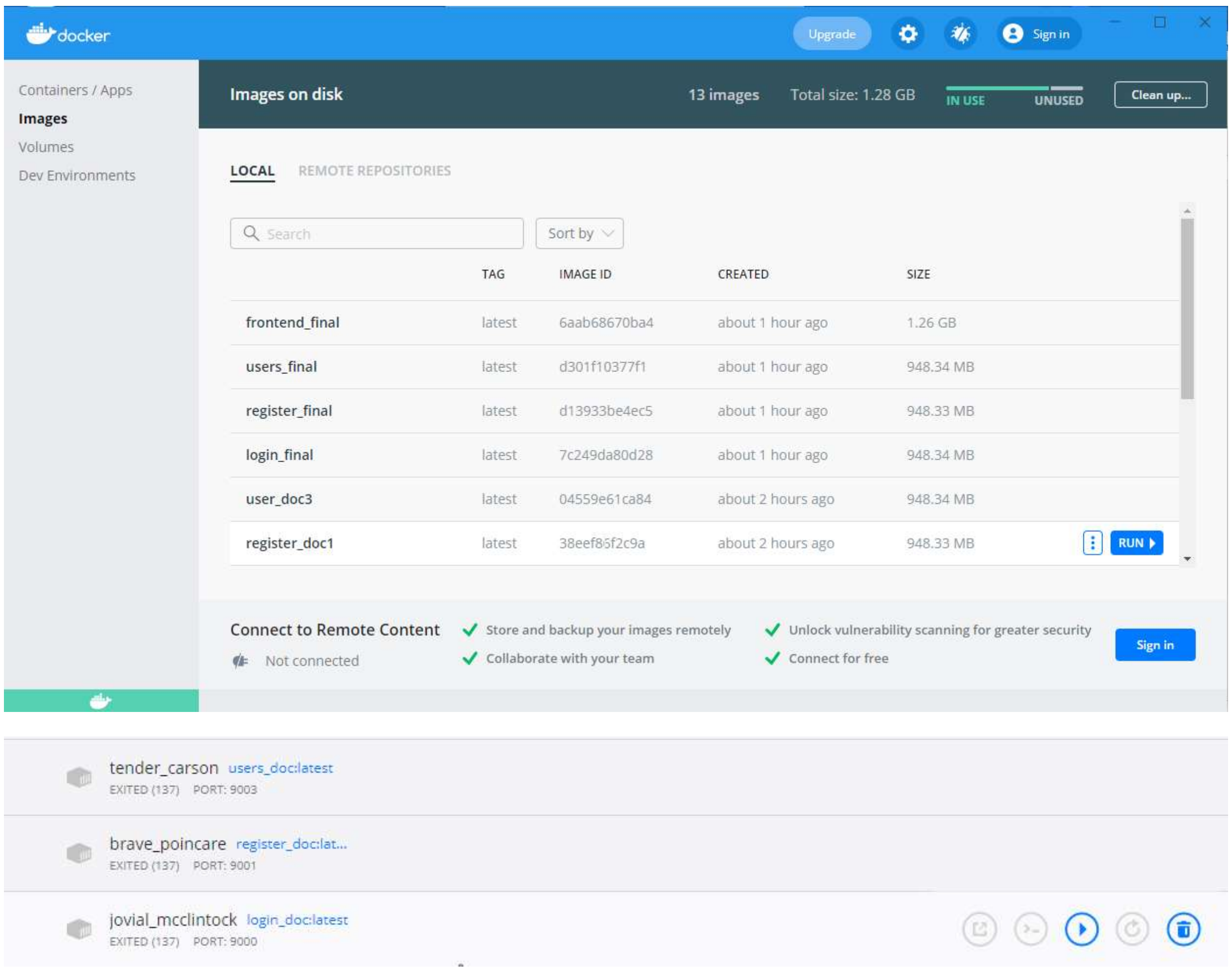
Id	Name	Email	Password	Contact
1	Dhuv	dhruv@dal.ca	dhruv	1234567890
2	Kush	kush@dal.ca	kush	9876543210
3	Aarohi	aarohi@dal.ca	aarohi	7410258963
4	Tejas	tejas@dal.ca	tejas	7852140369
5	tejasvini	tejasvini@dal.ca	tejasvini	1478523690
6	abc	abc@dal.ca	abc	7894561320
NULL	NULL	NULL	NULL	NULL



Userid	State
1	offline
2	offline
3	offline
4	offline
5	offline
6	offline
NULL	NULL

DOCKER IMAGES

After the development of the system on the local system I developed multiple docker images with command docker build -t <Tag Name> .



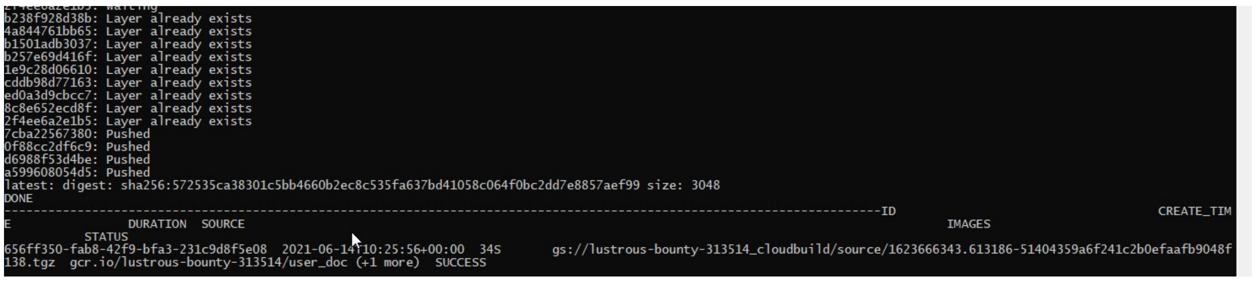
Images on disk — 13 images — Total size: 1.28 GB

	TAG	IMAGE ID	CREATED	SIZE
frontend_final	latest	6aab68670ba4	about 1 hour ago	1.26 GB
users_final	latest	d301f10377f1	about 1 hour ago	948.34 MB
register_final	latest	d13933be4ec5	about 1 hour ago	948.33 MB
login_final	latest	7c249da80d28	about 1 hour ago	948.34 MB
user_doc3	latest	04559e61ca84	about 2 hours ago	948.34 MB
register_doc1	latest	38eef85f2c9a	about 2 hours ago	948.33 MB

```
tender_carson     users_doc:latest      EXITED (137)  PORT: 9003
brave_poincare    register_doc:lat...   EXITED (137)  PORT: 9001
jovial_mcclintock login_doc:latest      EXITED (137)  PORT: 9000
```

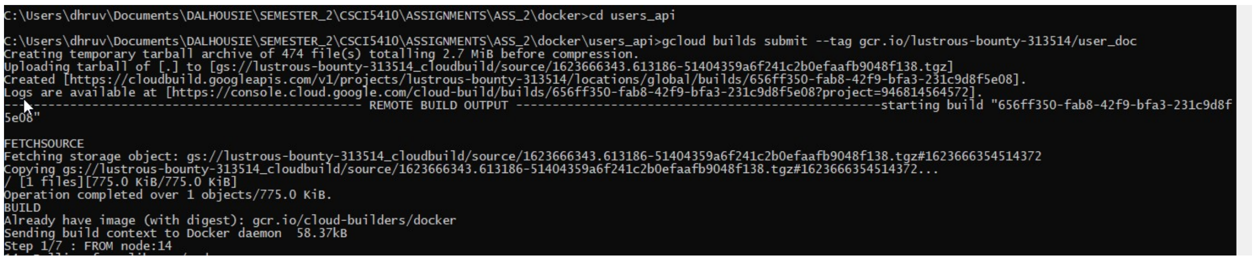
GOOGLE CLOUD SDK SHELL:

Uploading the docker images using GCP SDK is extremely easy I just need to fire a single command in the redundant repository and the work is done.



```
b238f928d38b: Layer already exists
4a844761bb65: Layer already exists
b1501adb3037: Layer already exists
b257e69d416f: Layer already exists
1e9c28d06610: Layer already exists
cddb98d77163: Layer already exists
ed0a3d9cbcc7: Layer already exists
8c8e652ecd8f: Layer already exists
2f4ee6a2e1b5: Layer already exists
7cba22567380: Pushed
0f88cc2df6c9: Pushed
d6988f53d4be: Pushed
a599608054d5: Pushed
latest: digest: sha256:572535ca38301c5bb4660b2ec8c535fa637bd41058c064f0bc2dd7e8857aef99 size: 3048
DONE
------------------------------------------------------------------------------------------------------------------------ID                                      CREATE_TIM
E                DURATION  SOURCE                                                                                                    IMAGES
          STATUS
656ff350-fab8-42f9-bfa3-231c9d8f5e08  2021-06-14T10:25:56+00:00  34S       gs://lustrous-bounty-313514_cloudbuild/source/1623666343.613186-51404359a6f241c2b0efaafb9048f
138.tgz  gcr.io/lustrous-bounty-313514/user_doc (+1 more)  SUCCESS
```

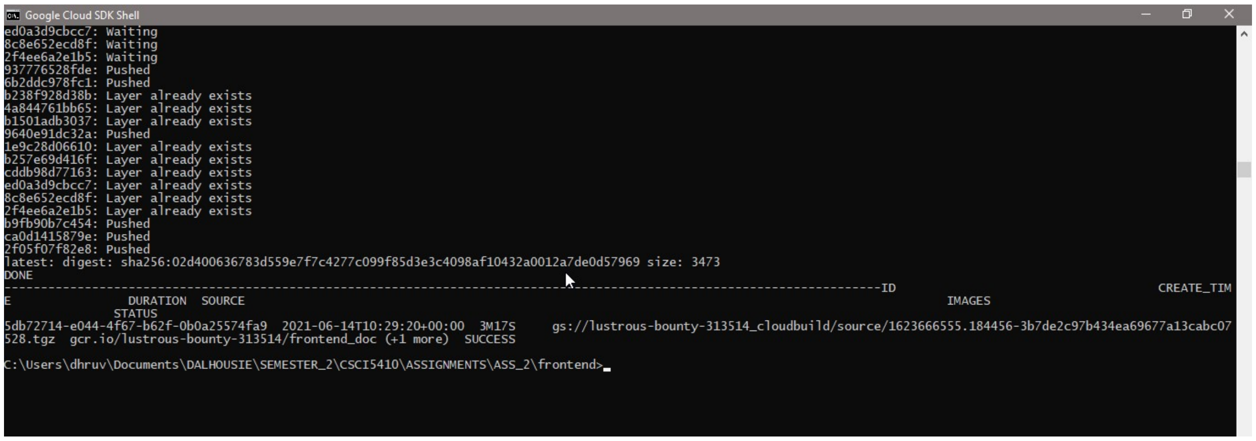
In screenshots there is an example of one API being pushed to the GCP containers.



```
C:\Users\dhruv\Documents\DALHOUSIE\SEMESTER_2\CSCI5410\ASSIGNMENTS\ASS_2\docker>cd users_api

C:\Users\dhruv\Documents\DALHOUSIE\SEMESTER_2\CSCI5410\ASSIGNMENTS\ASS_2\docker\users_api>gcloud builds submit --tag gcr.io/lustrous-bounty-313514/user_doc
Creating temporary tarball archive of 474 file(s) totalling 2.7 MiB before compression.
Uploading tarball of [.] to [gs://lustrous-bounty-313514_cloudbuild/source/1623666343.613186-51404359a6f241c2b0efaafb9048f138.tgz]
Created [https://cloudbuild.googleapis.com/v1/projects/lustrous-bounty-313514/locations/global/builds/656ff350-fab8-42f9-bfa3-231c9d8f5e08].
Logs are available at [https://console.cloud.google.com/cloud-build/builds/656ff350-fab8-42f9-bfa3-231c9d8f5e08?project=946814564572].
------------------------------------------------- REMOTE BUILD OUTPUT --------------------------------------------------starting build "656ff350-fab8-42f9-bfa3-231c9d8f
5e08"

FETCHSOURCE
Fetching storage object: gs://lustrous-bounty-313514_cloudbuild/source/1623666343.613186-51404359a6f241c2b0efaafb9048f138.tgz#1623666354514372
Copying gs://lustrous-bounty-313514_cloudbuild/source/1623666343.613186-51404359a6f241c2b0efaafb9048f138.tgz#1623666354514372...
/ [1 files][775.0 KiB/775.0 KiB]
Operation completed over 1 objects/775.0 KiB.
BUILD
Already have image (with digest): gcr.io/cloud-builders/docker
Sending build context to Docker daemon  58.37kB
Step 1/7 : FROM node:14
```



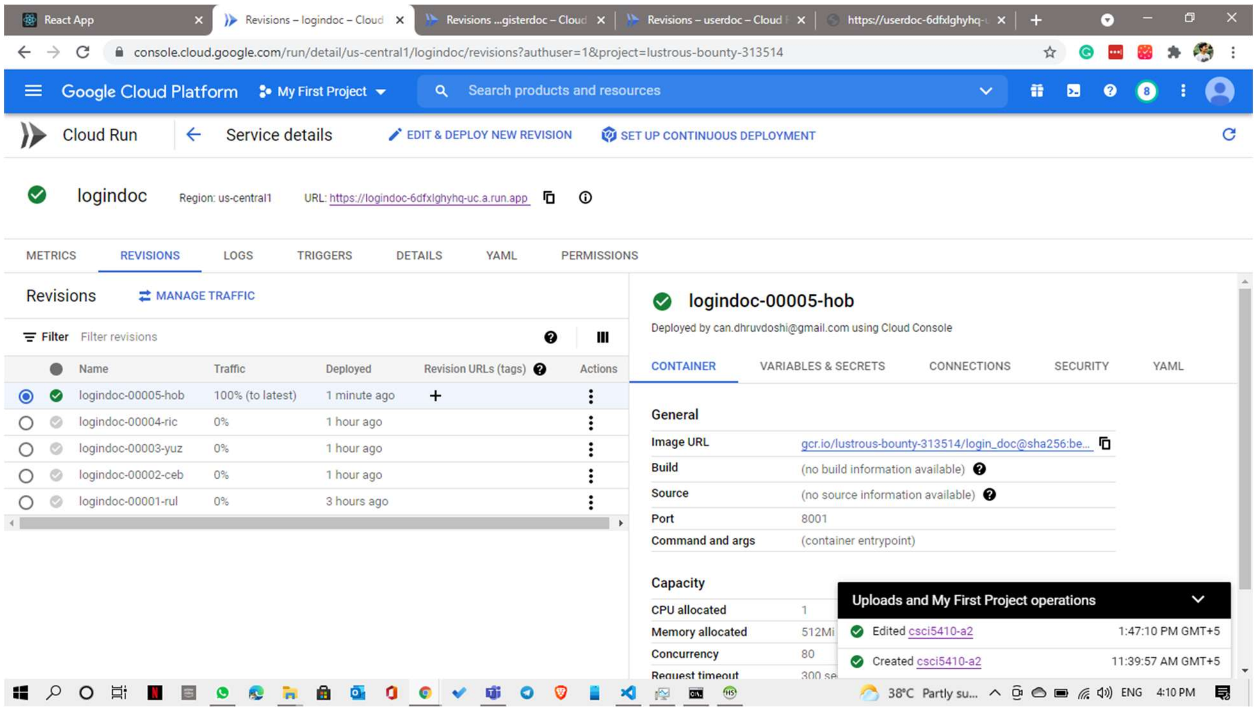
Google Cloud SDK Shell

```
ed0a3d9cbcc7: Waiting
8c8e652ecd8f: Waiting
2f4ee6a2e1b5: Waiting
937776528fde: Pushed
6b2ddc978fc1: Pushed
b238f928d38b: Layer already exists
4a844761bb65: Layer already exists
b1501adb3037: Layer already exists
9640e91dc32a: Pushed
1e9c28d06610: Layer already exists
b257e69d416f: Layer already exists
cddb98d77163: Layer already exists
ed0a3d9cbcc7: Layer already exists
8c8e652ecd8f: Layer already exists
2f4ee6a2e1b5: Layer already exists
b9fb90b7c454: Pushed
ca0d1415879e: Pushed
2f05f07f82e8: Pushed
latest: digest: sha256:02d400636783d559e7f7c4277c099f85d3e3c4098af10432a0012a7de0d57969 size: 3473
DONE
------------------------------------------------------------------------------------------------------------------------ID                                      CREATE_TIM
E                DURATION  SOURCE                                                                                                    IMAGES
          STATUS
5db72714-e044-4f67-b62f-0b0a25574fa9  2021-06-14T10:29:20+00:00  3M17S     gs://lustrous-bounty-313514_cloudbuild/source/1623666555.184456-3b7de2c97b434ea69677a13cabc07
528.tgz  gcr.io/lustrous-bounty-313514/frontend_doc (+1 more)  SUCCESS

C:\Users\dhruv\Documents\DALHOUSIE\SEMESTER_2\CSCI5410\ASSIGNMENTS\ASS_2\frontend>
```

GCP CONSOLE:

In following images there would be information regarding the services which I made from the images which we uploaded from the Google Cloud SDK Shell.



console.cloud.google.com/run/detail/us-central1/logindoc/revisions?authuser=1&project=lustrous-bounty-313514

Cloud Run ← Service details EDIT & DEPLOY NEW REVISION SET UP CONTINUOUS DEPLOYMENT

logindoc Region: us-central1 URL: https://logindoc-6dfxlghyhq-uc.a.run.app

Name	Traffic	Deployed	Revision URLs (tags)
logindoc-00005-hob	100% (to latest)	1 minute ago	
logindoc-00004-ric	0%	1 hour ago	
logindoc-00003-yuz	0%	1 hour ago	
logindoc-00002-ceb	0%	1 hour ago	
logindoc-00001-rul	0%	3 hours ago	

logindoc-00005-hob

Deployed by can.dhruvdoshi@gmail.com using Cloud Console

General

Image URL	gcr.io/lustrous-bounty-313514/login_doc@sha256:be...
Build	(no build information available)
Source	(no source information available)
Port	8001
Command and args	(container entrypoint)

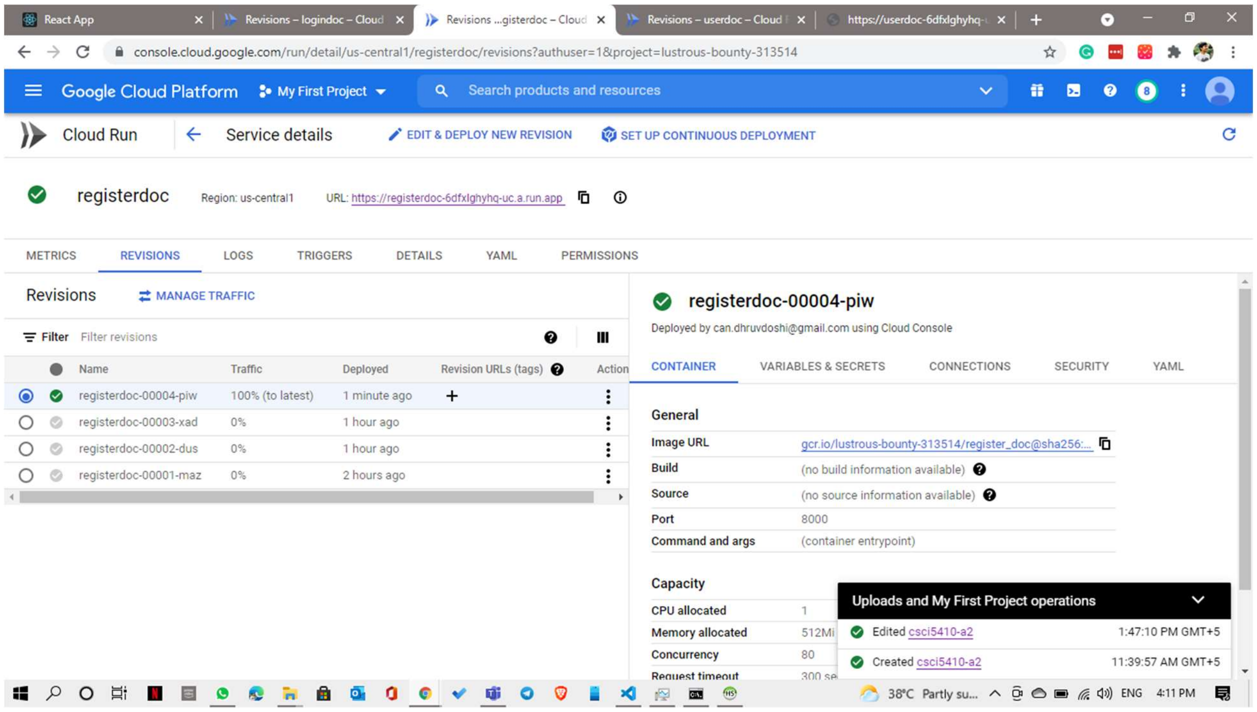
Capacity

CPU allocated	1
Memory allocated	512Mi
Concurrency	80

Uploads and My First Project operations

Edited csci5410-a2 1:47:10 PM GMT+5

Created csci5410-a2 11:39:57 AM GMT+5



console.cloud.google.com/run/detail/us-central1/registerdoc/revisions?authuser=1&project=lustrous-bounty-313514

Cloud Run ← Service details EDIT & DEPLOY NEW REVISION SET UP CONTINUOUS DEPLOYMENT

registerdoc Region: us-central1 URL: https://registerdoc-6dfxlghyhq-uc.a.run.app

Name	Traffic	Deployed	Revision URLs (tags)
registerdoc-00004-piw	100% (to latest)	1 minute ago	
registerdoc-00003-xad	0%	1 hour ago	
registerdoc-00002-dus	0%	1 hour ago	
registerdoc-00001-maz	0%	2 hours ago	

registerdoc-00004-piw

Deployed by can.dhruvdoshi@gmail.com using Cloud Console

General

Image URL	gcr.io/lustrous-bounty-313514/register_doc@sha256:...
Build	(no build information available)
Source	(no source information available)
Port	8000
Command and args	(container entrypoint)

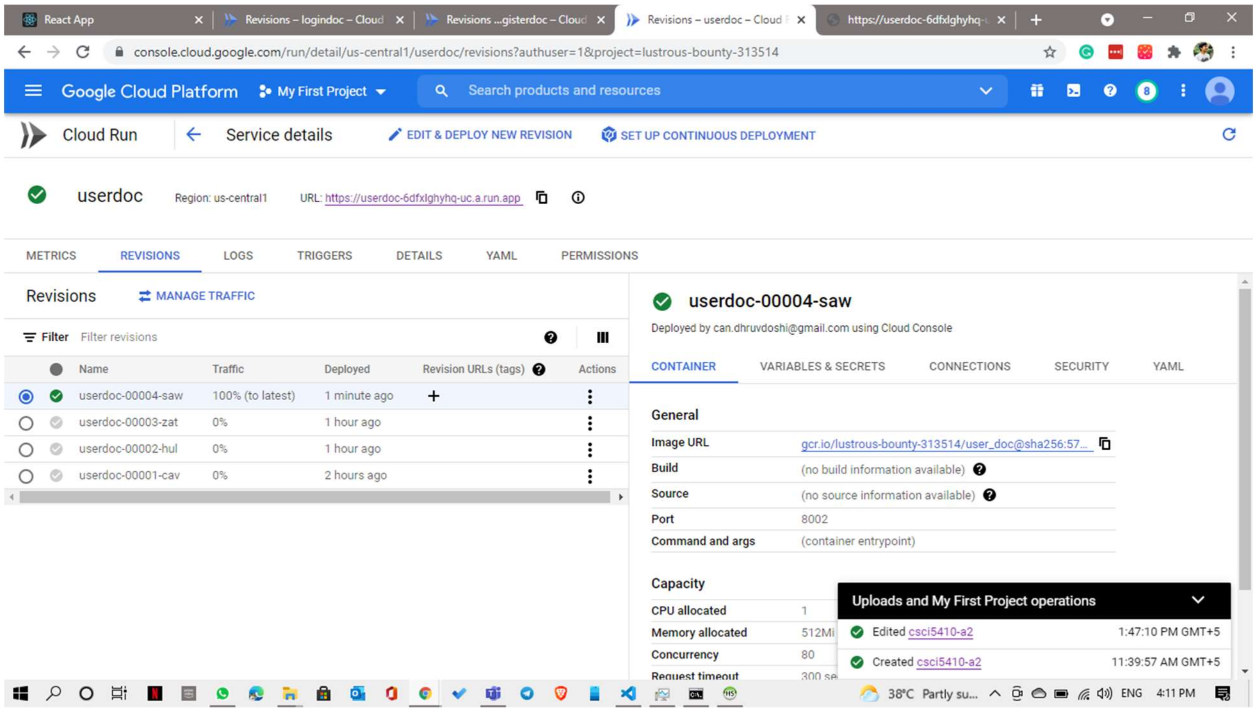
Capacity

CPU allocated	1
Memory allocated	512Mi
Concurrency	80

Uploads and My First Project operations

Edited csci5410-a2 1:47:10 PM GMT+5

Created csci5410-a2 11:39:57 AM GMT+5



console.cloud.google.com/run/detail/us-central1/userdoc/revisions?authuser=1&project=lustrous-bounty-313514

Cloud Run — Service details — EDIT & DEPLOY NEW REVISION — SET UP CONTINUOUS DEPLOYMENT

userdoc Region: us-central1 URL: https://userdoc-6dfxlghyhq-uc.a.run.app

METRICS REVISIONS LOGS TRIGGERS DETAILS YAML PERMISSIONS

Revisions — MANAGE TRAFFIC

Name	Traffic	Deployed	Revision URLs (tags)	Actions
userdoc-00004-saw	100% (to latest)	1 minute ago	+	⋮
userdoc-00003-zat	0%	1 hour ago		⋮
userdoc-00002-hul	0%	1 hour ago		⋮
userdoc-00001-cav	0%	2 hours ago		⋮

userdoc-00004-saw

Deployed by can.dhruvdoshi@gmail.com using Cloud Console

CONTAINER VARIABLES & SECRETS CONNECTIONS SECURITY YAML

General

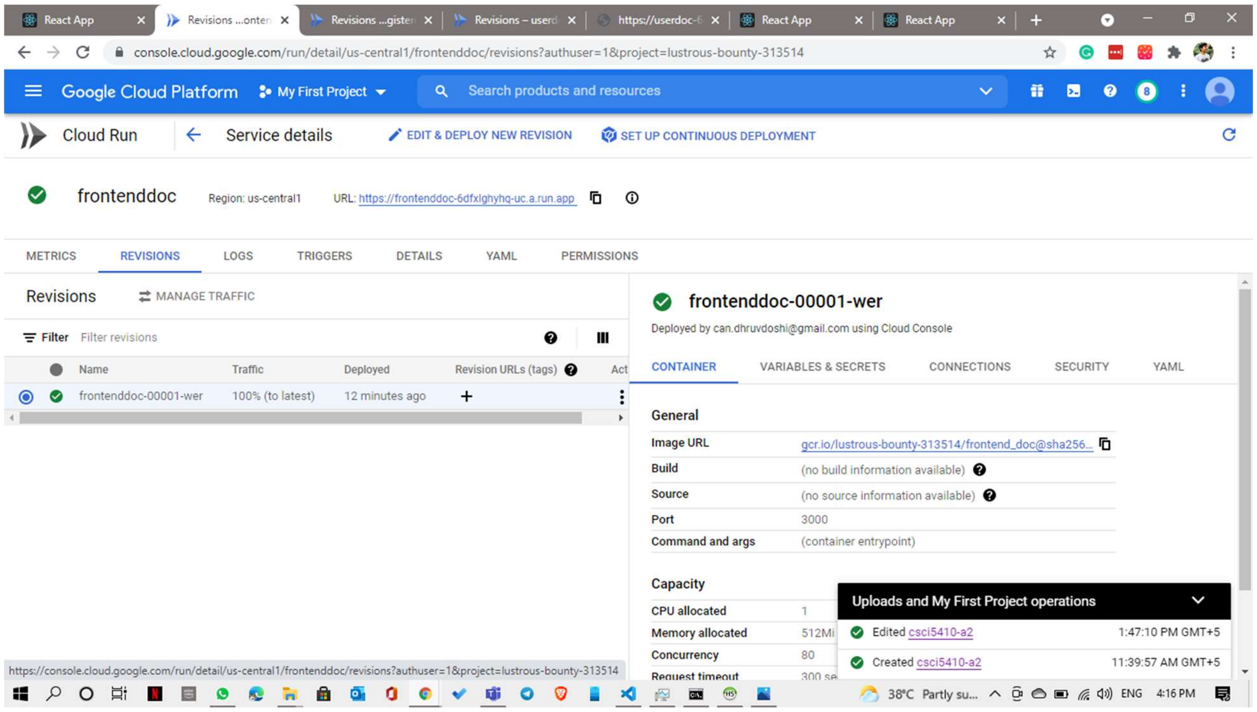
Image URL	gcr.io/lustrous-bounty-313514/user_doc@sha256:57...
Build	(no build information available)
Source	(no source information available)
Port	8002
Command and args	(container entrypoint)

Capacity

CPU allocated	1
Memory allocated	512Mi
Concurrency	80
Request timeout	300 se

Uploads and My First Project operations

Edited csci5410-a2	1:47:10 PM GMT+5
Created csci5410-a2	11:39:57 AM GMT+5



console.cloud.google.com/run/detail/us-central1/frontenddoc/revisions?authuser=1&project=lustrous-bounty-313514

Cloud Run — Service details — EDIT & DEPLOY NEW REVISION — SET UP CONTINUOUS DEPLOYMENT

frontenddoc Region: us-central1 URL: https://frontenddoc-6dfxlghyhq-uc.a.run.app

METRICS REVISIONS LOGS TRIGGERS DETAILS YAML PERMISSIONS

Revisions — MANAGE TRAFFIC

Name	Traffic	Deployed	Revision URLs (tags)	Act
frontenddoc-00001-wer	100% (to latest)	12 minutes ago	+	⋮

frontenddoc-00001-wer

Deployed by can.dhruvdoshi@gmail.com using Cloud Console

CONTAINER VARIABLES & SECRETS CONNECTIONS SECURITY YAML

General

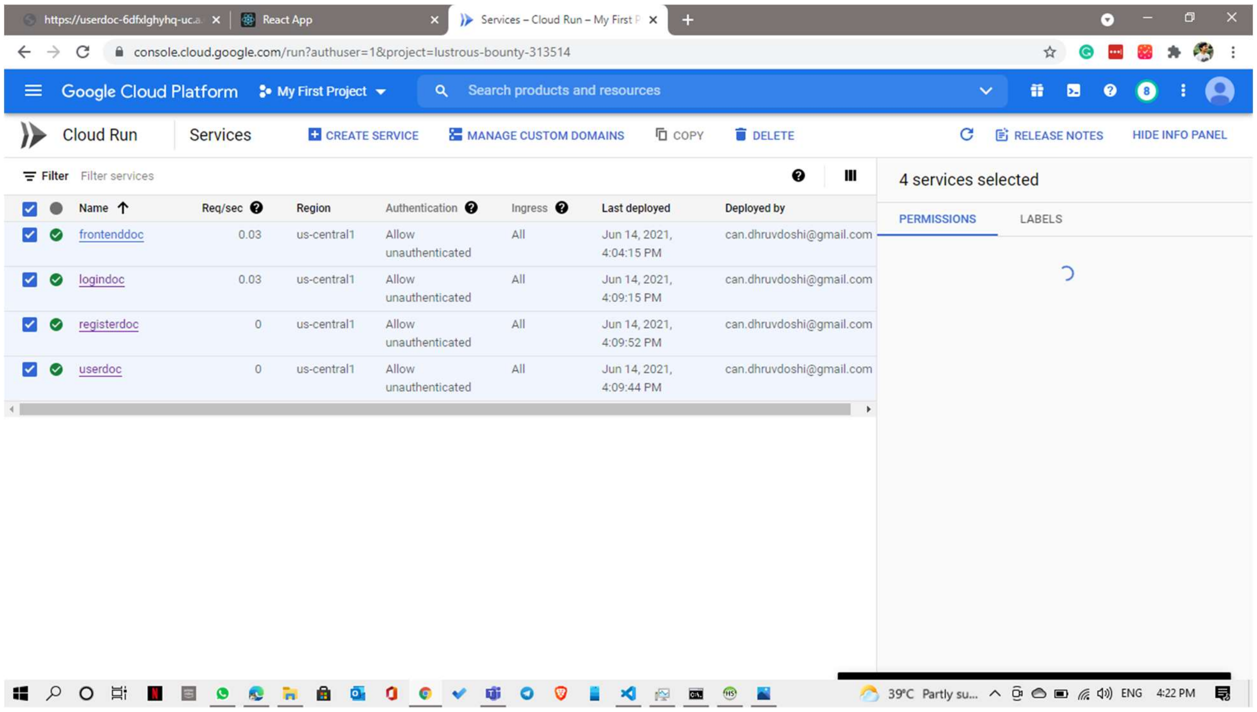
Image URL	gcr.io/lustrous-bounty-313514/frontend_doc@sha256...
Build	(no build information available)
Source	(no source information available)
Port	3000
Command and args	(container entrypoint)

Capacity

CPU allocated	1
Memory allocated	512Mi
Concurrency	80
Request timeout	300 se

Uploads and My First Project operations

Edited csci5410-a2	1:47:10 PM GMT+5
Created csci5410-a2	11:39:57 AM GMT+5



Cloud Run — Services

4 services selected

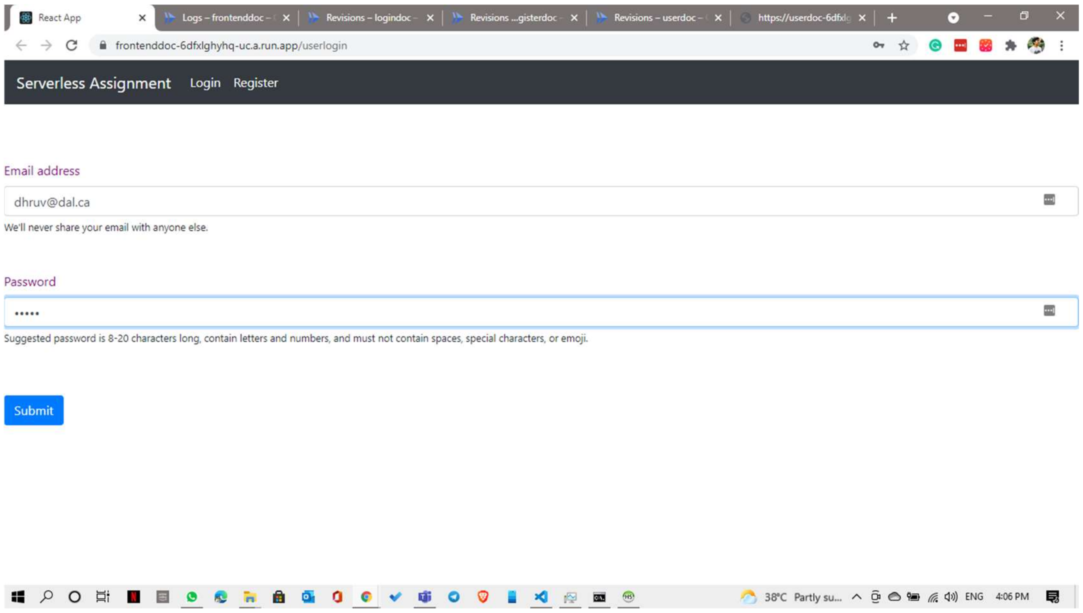
Name	Req/sec	Region	Authentication	Ingress	Last deployed	Deployed by
frontenddoc	0.03	us-central1	Allow unauthenticated	All	Jun 14, 2021, 4:04:15 PM	can.dhruvdoshi@gmail.com
logindoc	0.03	us-central1	Allow unauthenticated	All	Jun 14, 2021, 4:09:15 PM	can.dhruvdoshi@gmail.com
registerdoc	0	us-central1	Allow unauthenticated	All	Jun 14, 2021, 4:09:52 PM	can.dhruvdoshi@gmail.com
userdoc	0	us-central1	Allow unauthenticated	All	Jun 14, 2021, 4:09:44 PM	can.dhruvdoshi@gmail.com

WORKING APPLICATION

This module contains the screenshots of the working applciation, consiting Login page, User Registration page and User Status page.

LOGIN PAGE

Here in this login page user need to enter the email adress and password. After hitting the submit button the navbar will be refreshed and there will be three options register, users and logout.



frontenddoc-6dfxlghyhq-uc.a.run.app/userlogin

Email address

dhruv@dal.ca

We'll never share your email with anyone else.

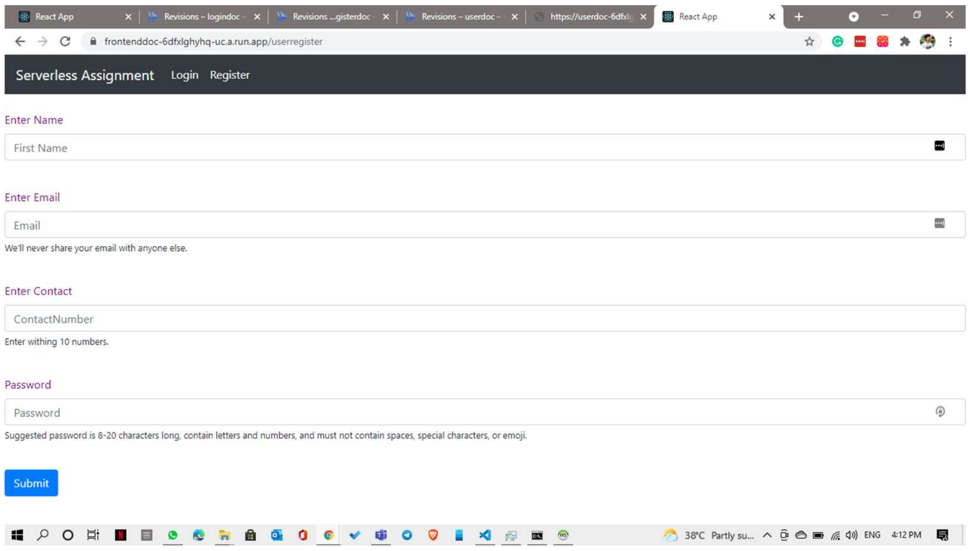
Password

•••••

Suggested password is 8-20 characters long, contain letters and numbers, and must not contain spaces, special characters, or emoji.

REGISTRATION PAGE

This page will reuire Name, Mail-id, Contact Info and Password from the user to make their entry in database table and then let them to login.



frontenddoc-6dfxlghyhq-uc.a.run.app/userregister

Enter Name

First Name

Enter Email

Email

We'll never share your email with anyone else.

Enter Contact

ContactNumber

Enter withing 10 numbers.

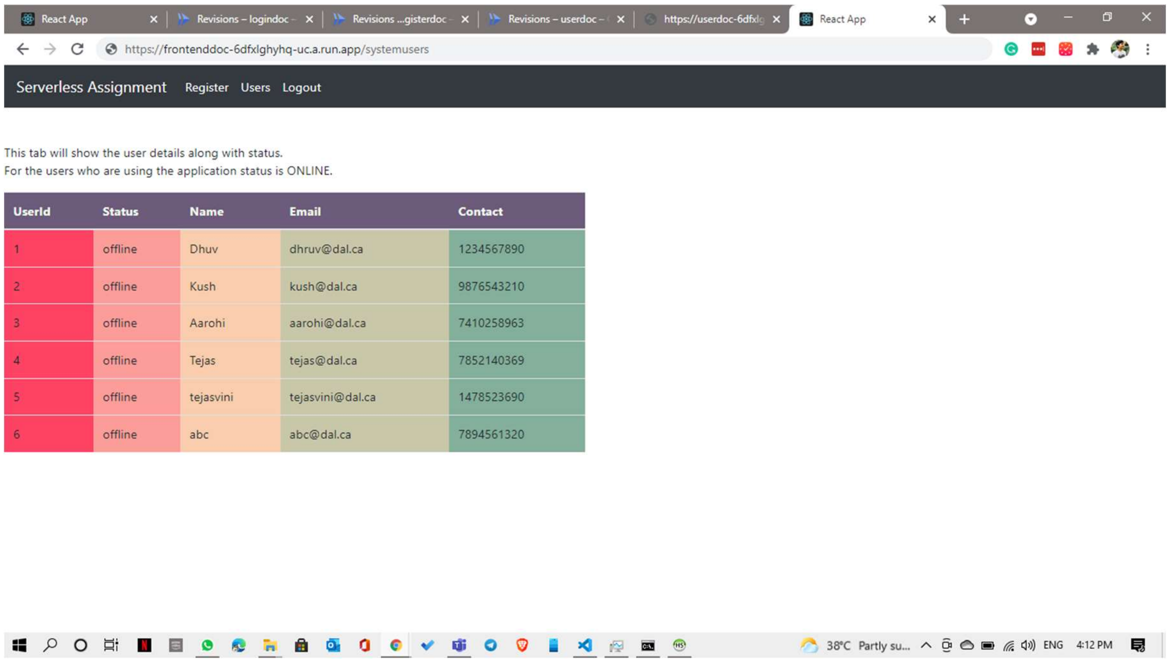
Password

Password

Suggested password is 8-20 characters long, contain letters and numbers, and must not contain spaces, special characters, or emoji.

USERS PAGE

This page will fetch information from the API and present it to the logged in User.



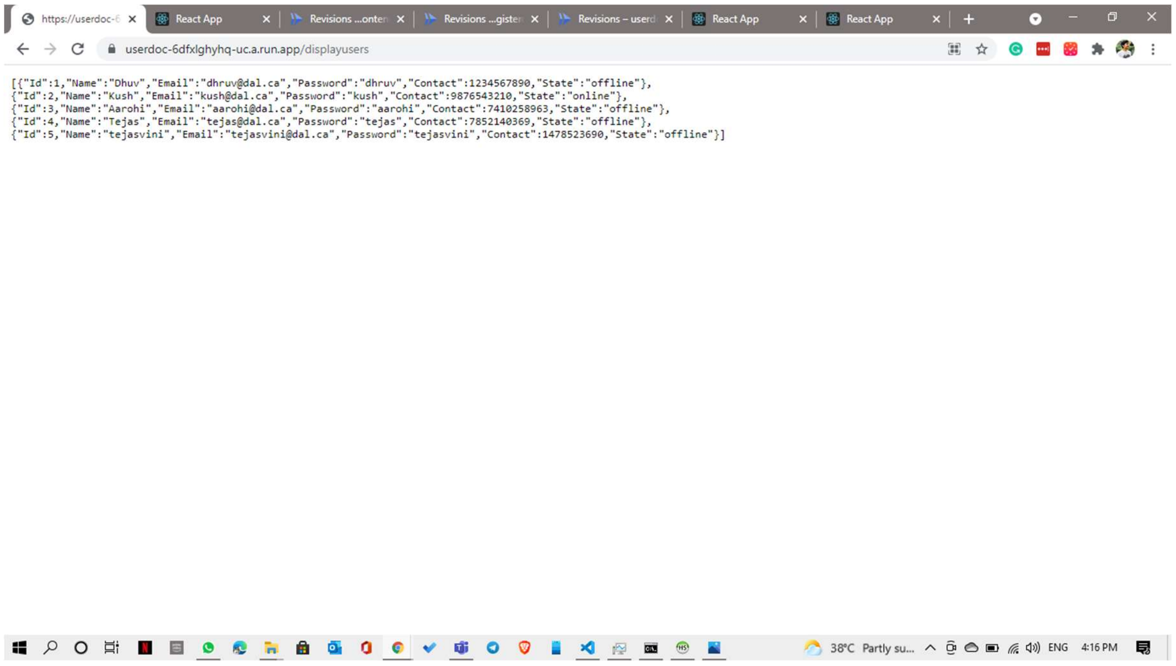
https://frontenddoc-6dfxlghyhq-uc.a.run.app/systemusers

Serverless Assignment Register Users Logout

This tab will show the user details along with status.
For the users who are using the application status is ONLINE.

UserId	Status	Name	Email	Contact
1	offline	Dhuv	dhruv@dal.ca	1234567890
2	offline	Kush	kush@dal.ca	9876543210
3	offline	Aarohi	aarohi@dal.ca	7410258963
4	offline	Tejas	tejas@dal.ca	7852140369
5	offline	tejasvini	tejasvini@dal.ca	1478523690
6	offline	abc	abc@dal.ca	7894561320

Example how the data comes from the API



userdoc-6dfxlghyhq-uc.a.run.app/displayusers

[{"Id":1,"Name":"Dhuv","Email":"dhruv@dal.ca","Password":"dhruv","Contact":1234567890,"State":"offline"},
{"Id":2,"Name":"Kush","Email":"kush@dal.ca","Password":"kush","Contact":9876543210,"State":"online"},
{"Id":3,"Name":"Aarohi","Email":"aarohi@dal.ca","Password":"aarohi","Contact":7410258963,"State":"offline"},
{"Id":4,"Name":"Tejas","Email":"tejas@dal.ca","Password":"tejas","Contact":7852140369,"State":"offline"},
{"Id":5,"Name":"tejasvini","Email":"tejasvini@dal.ca","Password":"tejasvini","Contact":1478523690,"State":"offline"}]

Summary & Technology Used:

Developing this application, I used React.js as frontend as I stated before along with that for backend handling, I had used Express.js along with Node.js. As described in the requirements we needed to use GCP SQL for database and that has been used through the whole application. For the deployment and hosting of the application I used the GCP containers and services along with google cloud run. Hence everything in this part had been done using GCP only.

Initially I had started the development in Python language but there were some issues with the dockers and python on my end so I moved to NodeJS as developing serverless application in is would be easy in my view. Initially I developed the code on the local system and with local database. After the successful working on the localhost, Then I developed database on GCP SQL and created the same database there also with help of SQL Workbench. After that I made an image of each API with help of Docker and then moved them to the google cloud. For this I used Google Cloud SDK Shell.

After successful uploading of every image in separate containers I started developing services on GCP for each image. Each service will give a unique URL to access them. Initially on local host I had to run multiple consoles and host every API individually and then use them now with these URL react is the only part which needed to be start from server. Now after updated the links I checked for the issues and there were some which I solved via configuration and public IP updating. After that I made the docker image for the frontend and with the help of Google Cloud SDK Shell I uploaded them to a container and then made a service. This URL is what we needed. Completely serverless system is running on GCP.

REFERENCES:

[1]"Documentation | Google Cloud", *Google Cloud*, 2021. [Online]. Available: https://cloud.google.com/docs. [Accessed: 14- Jun- 2021]

[2]"Quickly Deploy Your React App On Google's App Engine", *Medium*, 2021. [Online]. Available: https://javascript.plainenglish.io/quickly-deploy-your-react-app-on-googles-app-engine-6bb97480cc9c. [Accessed: 14- Jun- 2021]

[3]"Express 5.x - API Reference", *Expressjs.com*, 2021. [Online]. Available: https://expressjs.com/en/5x/api.html. [Accessed: 14- Jun- 2021]

[4]"body-parser", *npm*, 2021. [Online]. Available: https://www.npmjs.com/package/body-parser. [Accessed: 14- Jun- 2021]

[5]"jquery", *npm*, 2021. [Online]. Available: https://www.npmjs.com/package/jquery. [Accessed: 14- Jun- 2021]

[6]a. Mark Otto, "Introduction", *Getbootstrap.com*, 2021. [Online]. Available: https://getbootstrap.com/docs/5.0/getting-started/introduction/. [Accessed: 14- Jun- 2021]