DALHOUSIE UNIVERSITY

ASSIGNMENT-3 REPORT
Date: 14 June 2019
CSCI 5709- Cloud Computing

- **Theja Manasa Thatiparti**
B00813459

Application Scenario description:

To Satisfy the criteria for this assignment, I created a small application which stores user details such as FirstName, LastName, EmailAddress, and details column in User table. Details column serves as an index to profile table from User table and helps to merge these two tables and retrieve data accordingly. Profile table consists of details such as Interests, gender and Age of users.

Operations performed as part of this assignment:

- Deleting user records based on given EmailAddress
- Retrieving the interest of users above given age
- Retrieving the complete user details by performing merge operation of User and profile tables for given unique EmailAddress.

Software Description:

Backend overview: Sails framework is used for the development. It is a standalone application which provides Web Services that can be consumed by different applications.

Routes: Routes provide us means to connect from browser or frontend to webservice

```
'GET /deleteUser459' : 'UserController.deleteUser459',
```

Here deleteUser459 is the endpoint URL invoked via browser to connect with webservice and route the operation to perform deleteUser function.

```
'GET /findolderusers459': 'UserController.AgeCount',
```

Here findolderuser459 is the endpoint URL invoked via browser to connect with webservice and route the operation to perform AgeCount operation which delivers user interests details above certain age.

```
'GET /findProfiledetails459': 'UserController.retrieveprofilebymerge459',
```

Here findProfiledetails459 is the endpoint URL URL invoked via browser to connect with webservice and route the operation to perform retrieveprofilebymerge operation which delivers complete user details by merging user and profile tables.

To connect with database sails comes with pre-installed powerful object relational mapper (ORM) called Waterline. ORM abstracts the layer on top of database allowing users to query and manipulate data easily. Sails support multiple SQL/NO SQL databases. So, for this assignment I used MySQL cloud supported by Azure.

```
adapter: 'sails-mysql',
    host: 'cloudassign.mysql.database.azure.com',
    port: 3306,
    user:"manasa@cloudassign”
    password:"xxxxxxx",
    database:"user"
```

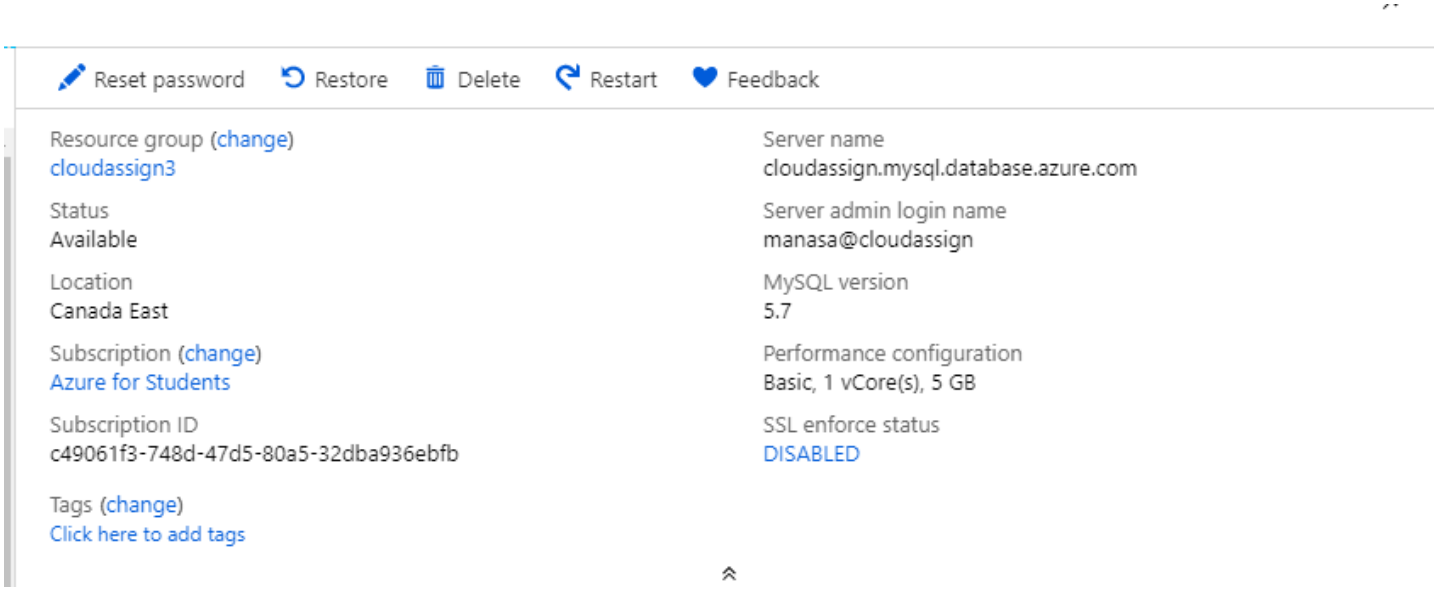


Figure 1:Cloud MYSQL database by Azure

DB tables:

MySQl workbench from local is connected as client to the Azure cloud server by disabling ssh and adding client IP addresses as per the change in network.

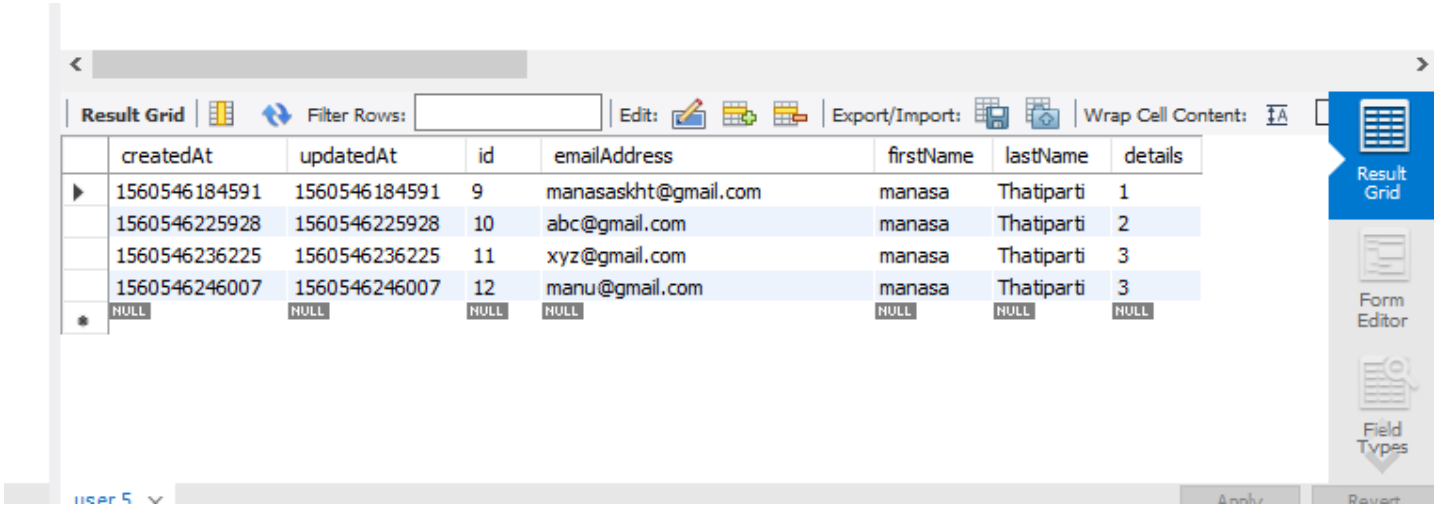


Figure 2:User table

User table with firstname, lastname ,emailaddress and details column. Details column is used to map user to profile table.

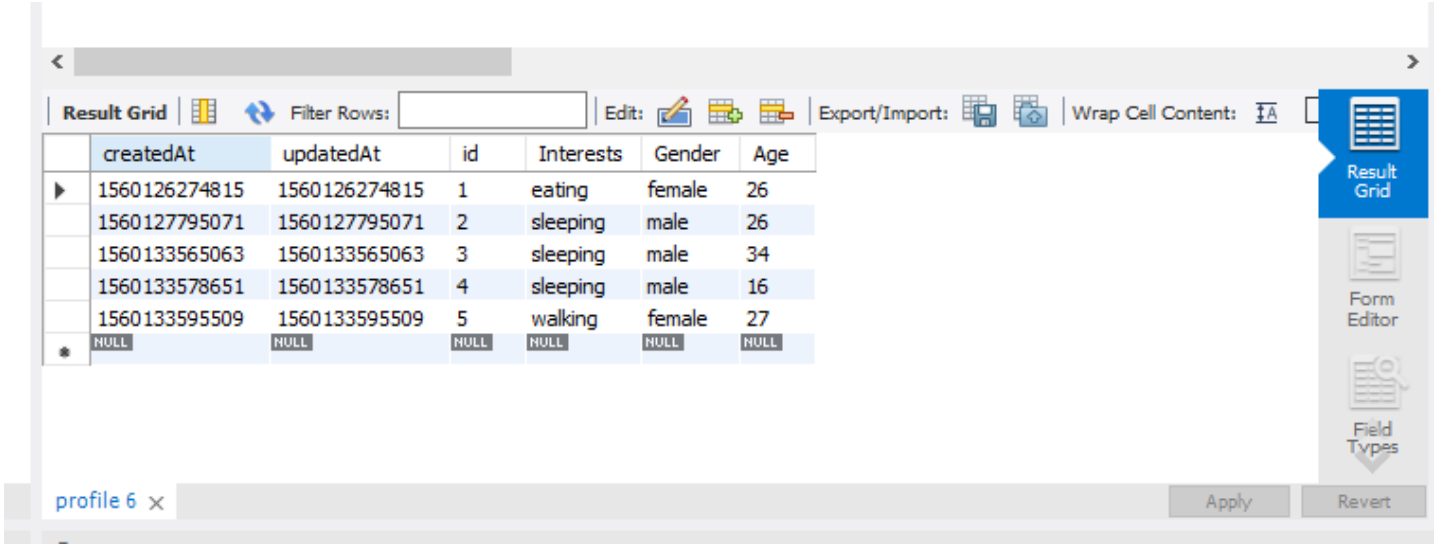


Figure 3: Profile table

Profile table created with details such as user interests, gender and Age. It is mapped to the respective Id of user table.

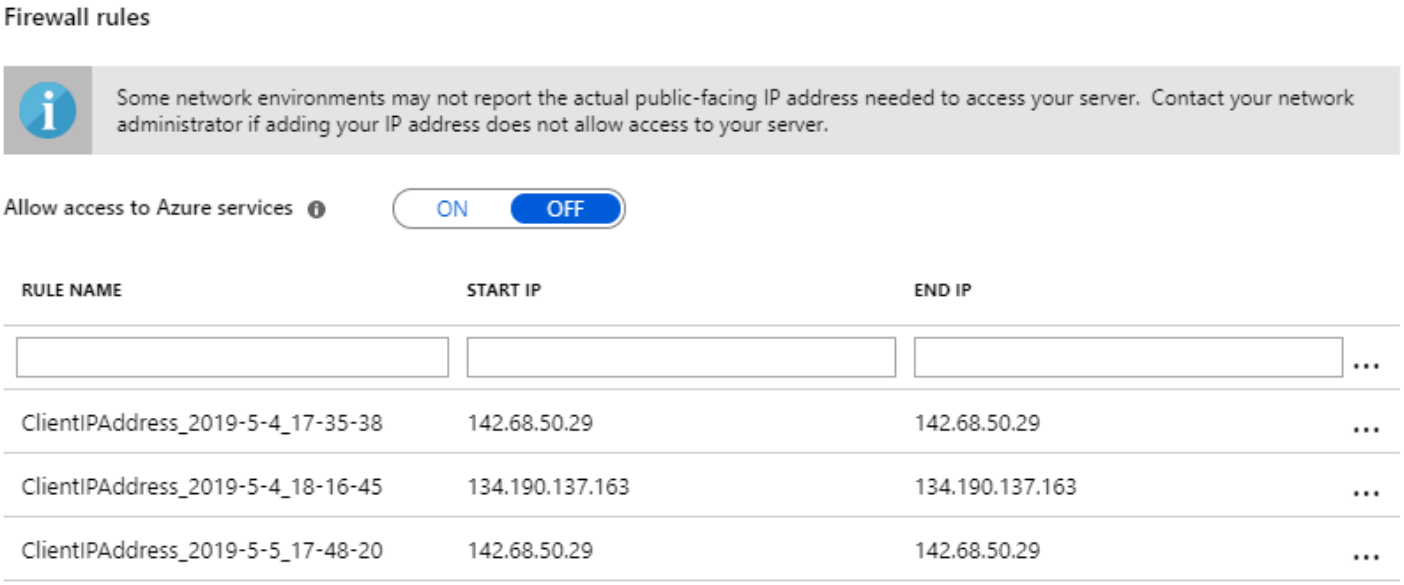


Figure 4:Client addresses to access

As per change in network, we need to add client IP address in the cloud database in order to connect with local client database and webservice(local).

Frontend Overview: Angular is used for frontend. Simple form is used to get inputs from user such as emailAddress or age in the textbox and select the appropriate action in the dropdown. It is a standalone application and it can access all the endpoints and invoke webservice to route and perform respective operation inside the controller.

Backend URl's to invoke services:

Delete:

```
http://localhost:1337/deleteUser459?email
```

Retrieve complete user details:

```
http://localhost:1337/findProfiledetails459?email
```

Retrieve user details above given age

```
http://localhost:1337/findolderusers459?age
```

Http connectivity to communicate frontend to webservice.

```
constructor(private http:Http) { }
import { Http } from '@angular/http';
```

ORM uses: ORM provides us the effect of virtual database that can be used within any framework, which supports it It abstracts the technical and traditional SQL queries and provides us with API that does these queries on behalf of us.

For example, in our application, to delete users based on the given email address

SQl query : delete * from user where emailaddress=manasaskht@gmail.com

ORM provides us api which does this query when we use just delete keyword as shown below.

```
var result = await User.destroy({emailAddress:id}).fetch();
```

.fetch is used to fetch the deleted data to display in the browser or to log .

Similarly, few other keywords such as populate, to populate child records when merged and find keyword to find the records in the database.

Description of Deployments

Frameworks:

Angular 7.1.4:

Angular is a Java Script framework for developing web application, it provides various built in features for easy development of Front-end applications. In addition, its capability of strong MVC structure built with the help of components eases the process while building large enterprise applications. In my assignment it is used to develop Web application (Frontend).

Sails:
Sails is an MVC framework for developing end to end Web applications, it is built upon Node.js. It provides features like automatic generation of controller, pages, api etc. Additionally, inbuilt security features in sails avoids various attacks such as Cross Site Scripting (CSS). In my assignment it is used to develop Web Services (Backend).

Libraries:

Http module (Angular): It provides functions of different HTTP services and in the assignment is used to make GET request to Backend.

CodeSnippets:

Source code and link: https://sailsjs.com/documentation/reference/waterline-orm/models/find

```
var usersNamedFinn = await User.find({name:'Finn'});
```

My code:

```
var result = await Profile.find({Age:{'>' :id}});

        return res.json(result);
      },
```

Source code and link: https://sailsjs.com/documentation/reference/waterline-orm/queries/populate

```
var usersNamedFinn = await User.find({name:'Finn'}).populate('dad');
sails.log('Wow, there are %d users named Finn.', usersNamedFinn.length);
return res.json(usersNamedFinn);
```

My code:

```
retrieveprofilebymerge459:async function(req,res)
      {
        var id=req.param('email');
        var result = await User.find({emailAddress:id}).populate ('details');
        return res.json(result);
      }
```

Source code and link: https://stackoverflow.com/questions/43956076/how-to-use-join-to-join-two-table-in-sails

```
age: {
      type: 'integer'
    },
    pony:{
      model: 'pet'
    }
  }
}
```

My code:

```
},
  details:
  {
    model:'Profile'
  }
```

Source code and link: https://www.w3schools.com/js/js_json_stringify.asp

```
 var foo = {foundation: 'Mozilla', model: 'box', week: 45, transport: 'car',
month: 7};
JSON.stringify(foo, replacer);
```

My code:

```
if(data.action == "query")
  {

this.http.get('http://localhost:1337/findolderusers459?age='+data.details).subscribe((data)
=> {this.result=data.json()},(err)=>{alert("Please enter age")});

    this.result=JSON.stringify(this.result);
    this.result=this.result.firstName;
   }
```

Source code and link: https://www.tutorialspoint.com/angular4/angular4_forms.html

```
<form #userlogin = "ngForm" (ngSubmit) = "onClickSubmit(userlogin.value)" >
<input type = "text" name = "emailid" placeholder = "emailid" ngModel>
```

My code:

```
            <form #user = "ngForm" (ngSubmit) = "onClickSubmit(user.value)" >
                <div class="row">
                 <input type = "text" name = "details" placeholder = "ex:68" ngModel>
```

DB software:

MySQL Workbench provides services such as SQL development, administration, database design, creation and maintenance. It provides easy access to cloud Mysql by providing Azure servicename, admin login name as shown below in figure.

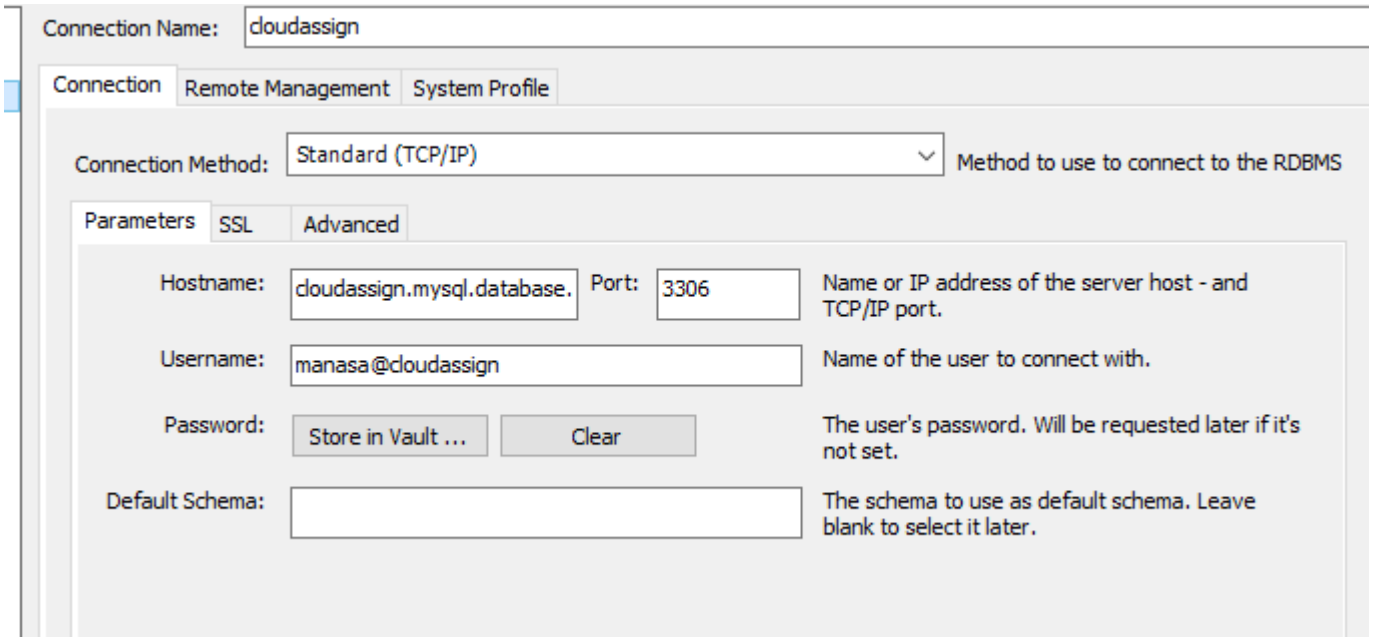


Figure 5: Workbench SQL connection to Cloud

Local deployment of frontend and webservice:

Steps to run Front End:
1. Install node.js
2. npm install -g @angular/cli
3. npm I @angular/http
4. cd Assignment1_Client
5. ng serve

Application URL: http://localhost:4200

Steps to run Web Service (backend):
Assuming that sails is installed:
1. cd Assignemnt1_WebServices
2. sails lift

Application End point: http://localhost:1337/?

? - changes as per operation and query, preconfigured in frontend as discussed above.

Cloud deployment of frontend and webservice:

Dockers are used to deploy both frontend and webservice in cloud.

Docker file sails:
Use node 10.15.1 LTS
FROM node:10.15.

```
# Change working directory
WORKDIR /users

# Copy source code
COPY . /users

# Install dependencies
RUN npm -g install sails

# Expose API port to the outside
EXPOSE 1337

# Launch application
CMD ["sails","lift"]
```

Docker file Angular:

```
# Use node 10.15.1 LTS
FROM node:10.15.1

# Change working directory
WORKDIR /userfrontend

# Copy source code
COPY . / userfrontend

# Install dependencies

RUN npm install -g @angular/cli
RUN npm I @angular/http
RUN ls

# Expose API port to the outside
EXPOSE 4200
# Launch application
CMD ["ng","serve"]
```

Docker compose file:

README.md | docker-compose.yml

```
1  version: "3"
2  services:
3    web:
4      image: assign2frontend
5      ports:
6        - "4200:4200"
7    backend:
8      image: weservice
9      ports:
10       - "1337:1337"
```

Testing:

Both frontend and backend deployed locally:

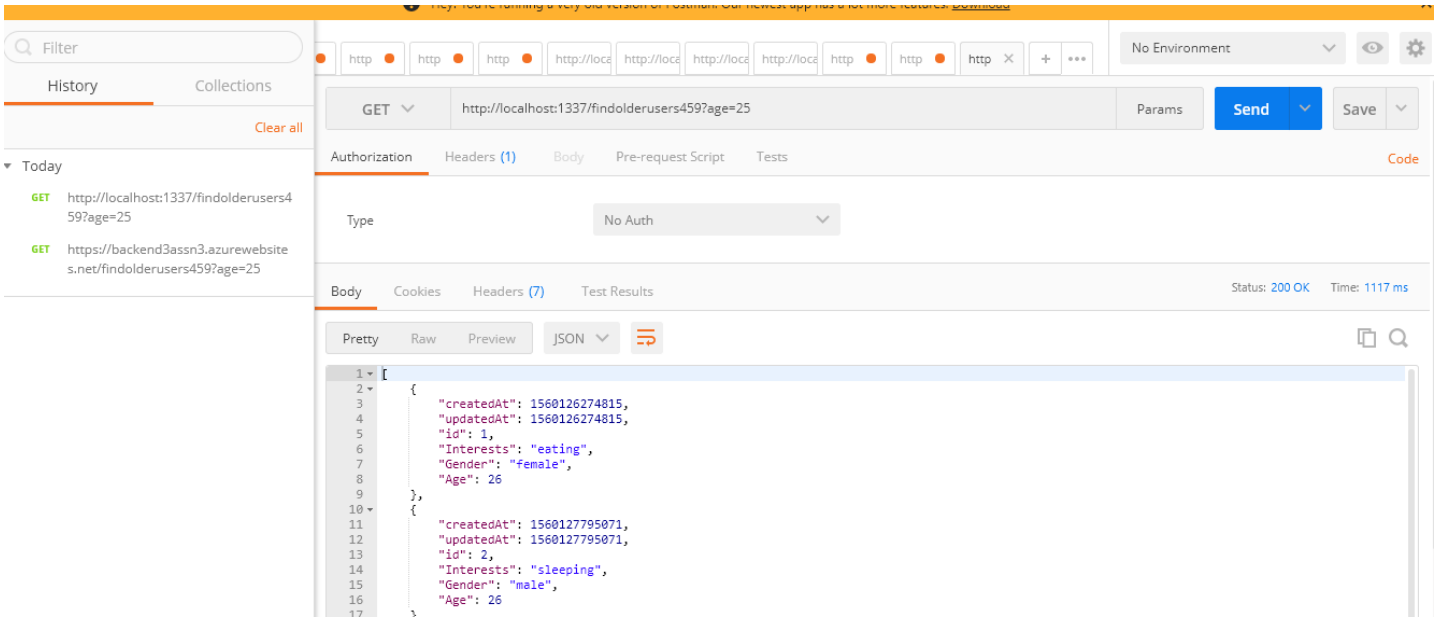
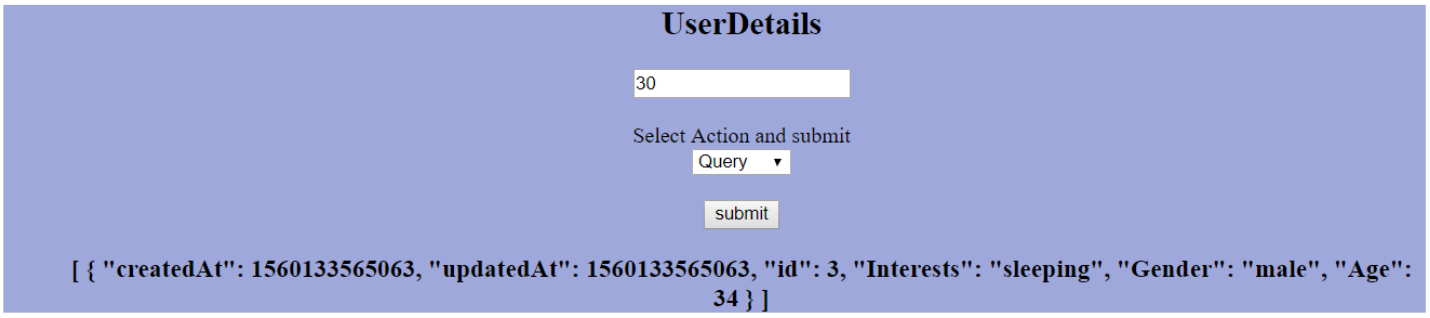


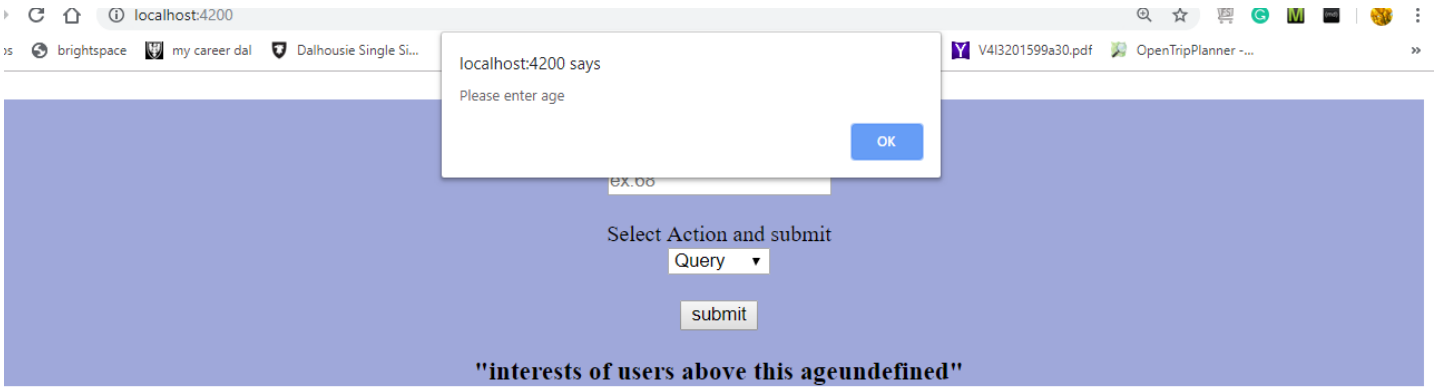
Figure 6:webservice deployed local tested via postmanl

User details above given age is displayed if action in dropdown is selected as Query. Here age is given as 25 so, user details who has age above 25 are displayed in json format.

User details above given age is displayed if action in dropdown is selected as Query. Here age is given as 30 so, user details who has age above 30 are displayed in json format.

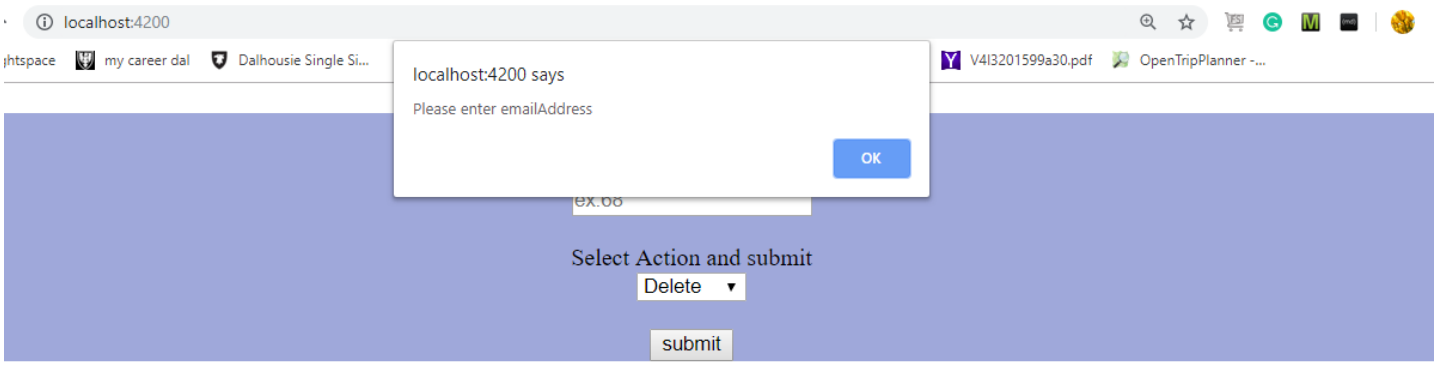


If age is given as empty, then it displays error message as below



Retrieve complete user details for given username (manasaskht@gmail.com)

If emailAddress is given as empty then it throws error message as below



If delete action is selected and provided with emailaddress then it delete the user from database and displays same in json format.

Deployed frontend and backend in cloud:

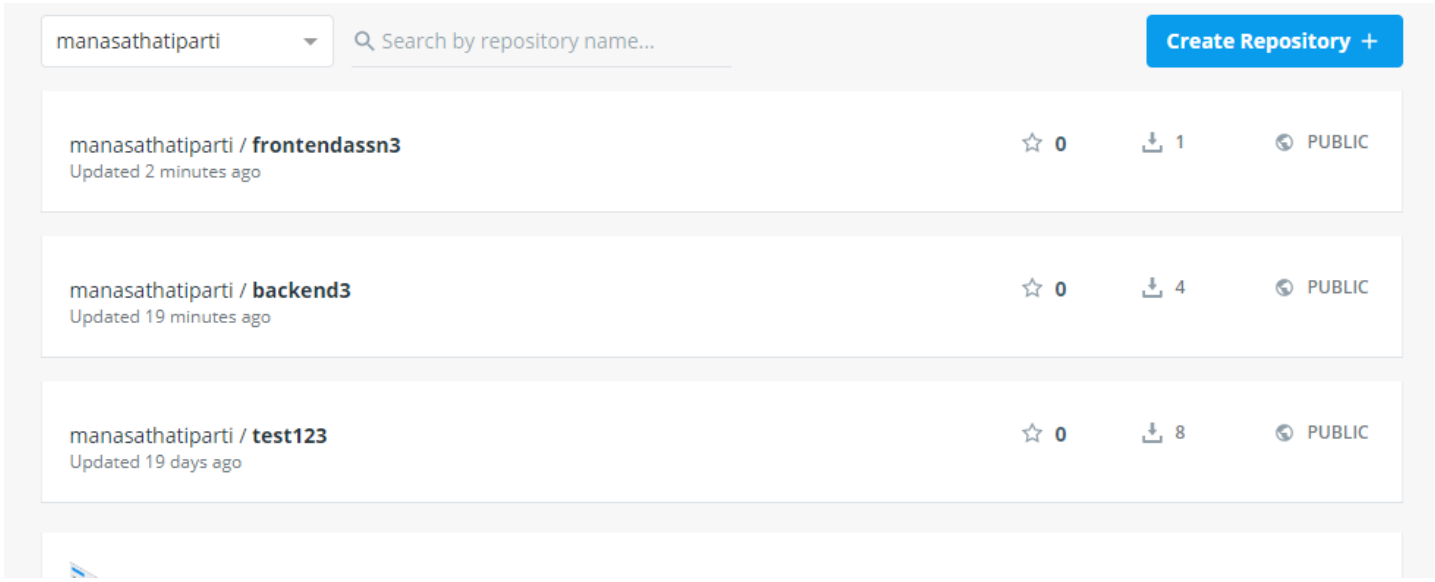
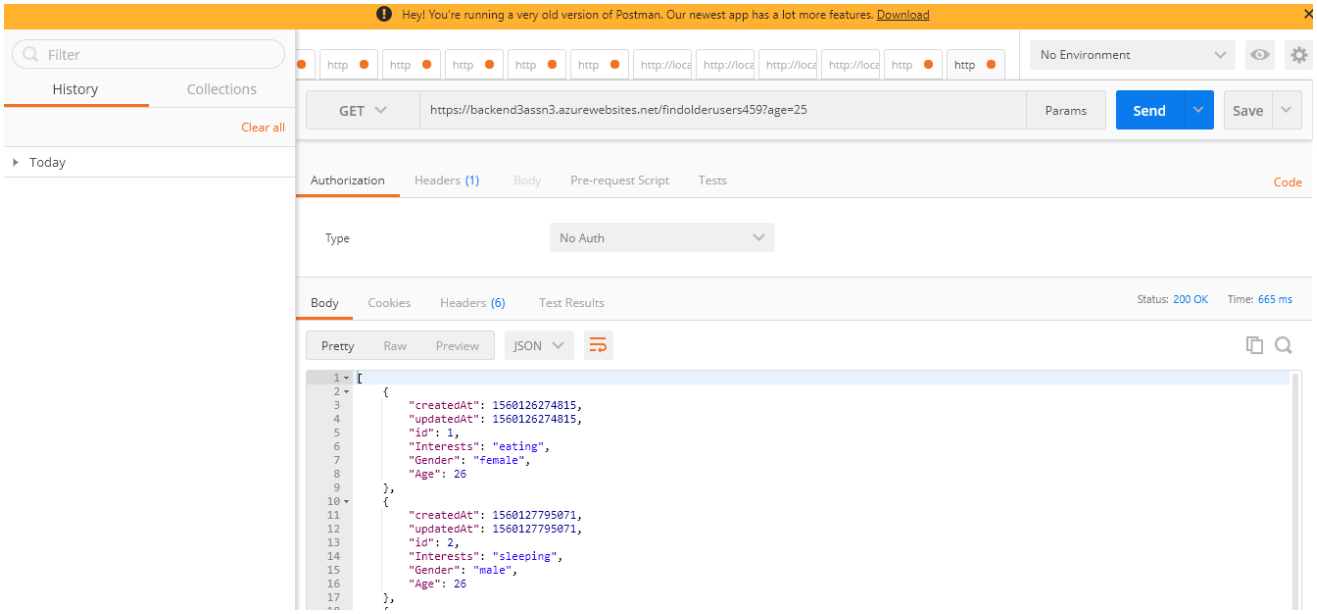
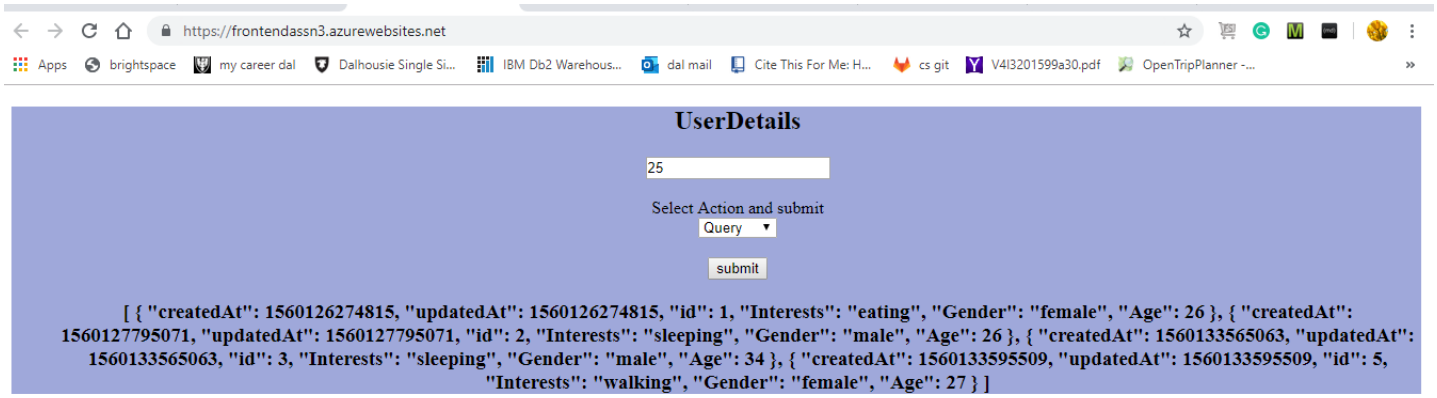


Figure 7: dockerhub images of frontend and backend

Tested using postman after deploying webservice in cloud using containers.
https://backend3assn3.azurewebsites.net/findolderusers459?age=25



Tested frontend deployed in cloud to invoke webservices deployed in cloud:



Git Link - backend: https://git.cs.dal.ca/thatiparti/clouda3backend.git

Frontend: https://git.cs.dal.ca/thatiparti/cloudassign3_frontend.git