**Firebase Notes**

* **Upload Data to Firebases**
  + <https://levelup.gitconnected.com/firebase-import-json-to-firestore-ed6a4adc2b57>
  + Need to be in json format
    - {collection\_name: [{‘id’: 0, ‘attr1’: attr1, …}, {‘id’: 1, ‘attr2’: attr2, …}, ….]}
  + Run the setup as specified in the article and upload the json file
* **Generate Service Account on Firebase**
  + Settings -> Users and Permissions -> Service Account -> Select Language -> generate private key
* **Get Started on Firebase Python**
  + <https://firebase.google.com/docs/firestore/quickstart#python>

**Mapbox + React Notes**

* Library: react-map-gl
* To convert latlong to place name (geocoding), and place name to latlong (reverse geocoding)
* Examples: <https://visgl.github.io/react-map-gl/examples>

**Tutorials Followed**

* Upload JSON to Firebase: <https://levelup.gitconnected.com/firebase-import-json-to-firestore-ed6a4adc2b57>
* SearchBar: <https://levelup.gitconnected.com/building-a-simple-dynamic-search-bar-in-react-js-f1659d64dfae>
* Mapbox: <https://www.youtube.com/watch?v=JJatzkPcmoI>
* Interactive Star Ratings: <https://github.com/fedoryakubovich/react-awesome-stars-rating>
* React-geolocated: <https://www.npmjs.com/package/react-geolocated>
* Forms and validation (React-hook-forms): <https://react-hook-form.com/get-started>
* React tutorial: <https://reactjs.org/docs/hello-world.html>

**Front End Application**

* **Components folder**
  + Business.js: Renders the business page that contains business info, and reviews.
  + Header.js: Renders the Header and the images, and cards in home page
  + Home.js: Gets current location and store in sessionStorage, and calls header component
  + Login.js: Renders login page
  + Maincomponent.js: Main page and creates routes to pass on to other components
  + Searchresults.js: Returns div of all search results from header.js
  + Signup.js: Renders the signup page
  + Writeareview.js: Renders page to write and post reviews and get prediction from BERT model.

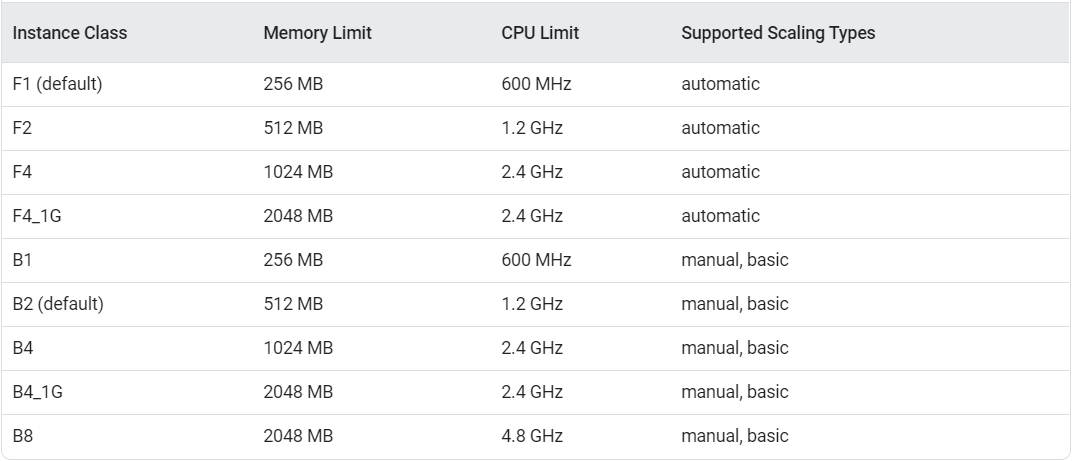
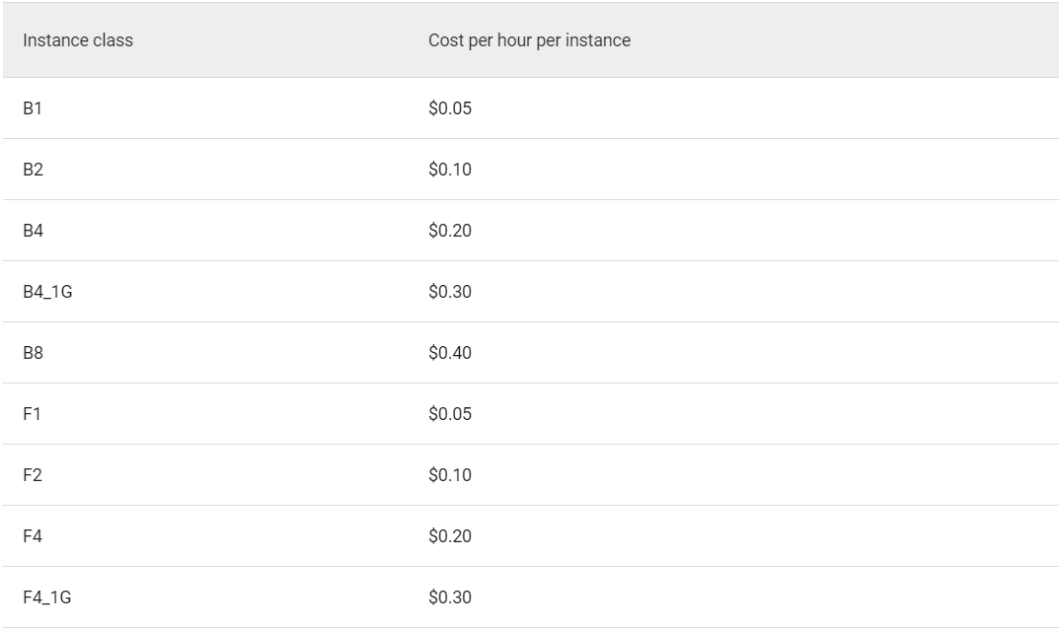
**Back End Flask**

* **Methods**
  + GET “/”: checks if server is running or not
  + GET “/getAllBusiness”: gets all businesses
  + GET “/getThreeNearestBusiness?lat=40.7128&lon=74.0060”: get three nearest businesses
  + GET “/getOneBusinessData?business\_id=4o2DEgSgY2sDZ16g6U03-w”: get one business data
  + GET “/getOneBusinessReview?business\_id=4o2DEgSgY2sDZ16g6U03-w”: get one business data review joined with user data
  + GET “/getOneUserData?user\_id=Iq9EiUS7FmFB6Uw3PBghsQ”: get one user data
  + GET “/getOneUserReview?user\_id=Iq9EiUS7FmFB6Uw3PBghsQ”: get one user review
  + POST “/postreview”: post user review on current business id
  + POST “/authenticateLogin”: authenticate user login (checks for password)
  + POST “/signupuser”: post signup user data
  + DELETE “/deleteUserReviews?user\_id=admin”: delete one user review
  + DELETE “/deleteOneUser?name=test”: delete one user
  + POST “/predict”: predict the current review using BERT model

**Deploy to GCP**

* Create app.yaml file
  + Documentation: <https://cloud.google.com/appengine/docs/standard/python3/config/appref#instance_class>
  + basic\_scaling: Use this type
    - max\_instances: 1 (don’t want too many instances)
    - idle\_timeout: 10m (timeout after 10 minutes)
  + instance\_class: B4 (use the B4 instance)
  + runtime: python37 (python 3.7 run time environment)
  + entrypoint: "gunicorn -b:$PORT firebase:app" (run flask app default gcloud app engine is 8080, maps the file firebase.py to app and use gunicorn to deploy the flask app)
* In flask app, put **app.run(port=8080)**
* **ALSO NEED CORS ENABLED!!** (Flask-cors package in python)
* Commands:
  + gcloud auth login = login and select account
  + gcloud projects list = display all projects
  + gcloud config set project yelp-sentiment-analysis-286600 = change current project
  + gcloud app deploy = deploy
* app link backend: <https://yelp-sentiment-backend-286616.ue.r.appspot.com>
* check if server is running: [https://yelp-sentiment-backend-286616.ue.r.appspot.com/](https://yelp-sentiment-backend.ue.r.appspot.com/)

**Notes on GCP Free Tier**

* App Engine
  + 28 hours per day of "F" instances
  + 9 hours per day of "B" instances
  + 1 GB of egress per day
* How Instances are managed?
  + <https://cloud.google.com/appengine/docs/standard/go/how-instances-are-managed>
  + F instances need to be running all the time
  + B instances supports Manual/Basic scaling which will shutdown after idle time (default 5min)
* Instance classes
  + <https://cloud.google.com/appengine/docs/standard#instance_classes>
* Cost after Free Tier
  + <https://cloud.google.com/appengine/pricing>

**React Deployment**

* <https://medium.com/better-programming/how-to-deploy-your-react-app-to-heroku-aedc28b218ae>
* Deployment steps
  + Create build folder = npm run build
  + In root folder (yelp-project) create server folder
  + In server folder, create server.js file (will use express)
  + To install express: npm install express --save
  + In server.js file add

const path = require('path');  
const express = require('express');  
const app = express();  
const publicPath = path.join(\_\_dirname, '..', ‘build’);  
const port = process.env.PORT || 3000;

app.use(express.static(publicPath));

**app.get('\*', (req, res) => {  
 res.sendFile(path.join(publicPath, 'index.html'));  
});**

app.listen(port, () => {  
 console.log('Server is up!');  
});

* + To test if server is running inside root folder (yelp-project) -> node server/server.js
  + Next is to deploy to Heroku
    - Change “start” in package.json from “react-scripts start” TO “node server/server.js”
  + Initialize git repo -> git add . -> git commit -m “commit message”
  + heroku git:clone -a yelp-application-clone (to add existing git repo to Heroku)
  + git push Heroku master (to build application)
* AppLink: <https://yelp-application-clone.herokuapp.com/>