# Welcome to the Airgraft backend programming exercise

In this exercise, we will assess your technical knowledge, creativity, and fit.

The exercise is called City Autocomplete API Design and Implementation, and we're excited to see how you'd bring your own flavours to your solution!

# The Task

Design a REST API endpoint that provides auto-complete suggestions for large cities.

- The endpoint is exposed at `/suggestions`
- The partial (or complete) search term is passed as a querystring parameter `q`
- The caller's location can optionally be supplied via querystring parameters `latitude` and `longitude` to help improve relative scores
- The endpoint returns a JSON response with an array of scored suggested matches
    - The suggestions are sorted by descending score
    - Each suggestion has a score between 0 and 1 (inclusive) indicating confidence in the suggestion (1 is most confident)
    - Each suggestion has a name which can be used to disambiguate between similarly named locations
    - Each suggestion has a latitude and longitude

## Bonus points

- Can you think of more scoring parameters to make the suggestions better? Add them to your documentation
- Sketch out a design for per user API keys and billing for our future city-suggestion-service startup. What are the implications for scalability of your implementation?

# The Rules

- For the server side we are most comfortable with Java and nodejs (either JavaScript or TypeScript), so it would be best if you stuck to one of these stacks.

- Besides that you have pretty much full leeway over your solution. We recommend developing your solution in a way that is easy to run locally from a fresh checkout, without requiring any third party API keys or account.

## Advice

- Try to design and implement your solution as you would do for real production code. Show us how you create clean, maintainable code that does awesome stuff. Build something that we'd be happy to contribute to. This is not a programming contest where dirty hacks win the game.
- Documentation and maintainability are a plus, and don't you forget those unit tests.
- We don’t want to know if you can do exactly as asked (or everybody would have the same result). We want to know what you bring to the table when working on a project, what is your secret sauce. More features? Best solution? Thinking outside the box?

## Deliverable

Please submit a working & runnable solution either in a .zip file or a git repository to checkout.

The reviewer of your solution will be a developer, so you can make some assumptions about the runtime, e.g.

- there will be a recent JVM / nodejs
- the reviewer will be able to run maven/gradle/npm outside an IDE
- the reviewer will have intellij/vscode installed
- docker will be available to run additional services you need (e.g. a database)

Ideally your solution should be "offline" in the sense that it does not require any third party services. If it absolutely must, please explain the reasoning and be aware of any rate limits that might stop your solution from working on our side.

## Can I use a database?

If you wish, it's OK to use external systems such as a database, an Elastic index, etc. in your solution. But this is certainly not required to complete the basic requirements of the

challenge. Keep in mind that our goal here is to see some code of yours; if you only implement a thin API on top of a DB we won't have much to look at.

Our advice is that if you choose to use an external search system, you had better be doing something really truly awesome with it.

## Sample responses

These responses are meant to provide guidance. The exact values can vary based on the data source and scoring algorithm

### Near match

```
GET /suggestions?q=Londo&latitude=43.70011&longitude=-79.4163
```


```
{
  "suggestions": [
    {
      "name": "London, ON, Canada",
      "latitude": "42.98339",
      "longitude": "-81.23304",
      "score": 0.9
    },
    {
      "name": "London, OH, USA",
      "latitude": "39.88645",
      "longitude": "-83.44825",
      "score": 0.5
    },
    {
      "name": "London, KY, USA",
      "latitude": "37.12898",
      "longitude": "-84.08326",
      "score": 0.5
    },
    {
      "name": "Londontowne, MD, USA",
```

```
      "latitude": "38.93345",
      "longitude": "-76.54941",
      "score": 0.3
    }
  ]
}
```

### No match

```
GET /suggestions?q=SomeRandomCityInTheMiddleOfNowhere
```

```
{
  "suggestions": []
}
```

## References

- Geonames provides city lists for Canada and the USA
  http://download.geonames.org/export/dump/readme.txt