

Neighbourhood Checker

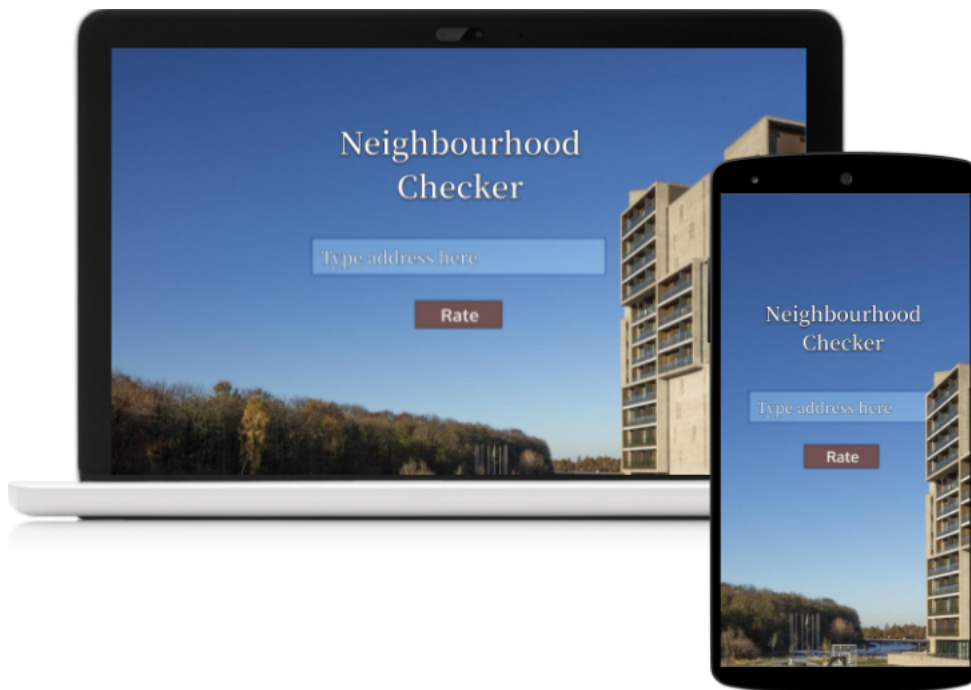
Joakim Houmøller - 150898 - Jonoe19

Mathias Jensen - 010499 - Maje419

Nicklas Jacobsen - 211098 - Njaco19

Collectively responsible for all 445 (295 excluding post-requests) lines of code in the project

December 2021



Link to source code: <https://github.com/Maje419/InnovationProject>

Contents

1	Our idea	1
2	Value Propositions	2
3	Customer segments	2
4	Customer relationships	2
5	Channels	2
6	Key Activities	3
7	Key Resources	3
8	Key Partners	3
9	Cost structure	4
10	Revenue streams	4
11	Potential datasets	5
12	The program	6
12.a	The design idea	6
12.b	Calling the API(s)	6
12.c	Scoring the data	7
13	Prototype interface	8
14	End user interface	9

1 Our idea

We, the developers, have all experienced the confusion of moving to a new address firsthand. As 3 young men from three different corners of the country, we have all recently moved to Odense, an area of which none of us had any prior knowledge. Searching around many different sources, such as the Danish Building Registry or DinGeo, yielded a tonne of information, more than any average human could reasonably digest. Looking through data about asbestos in the walls, noise pollution graphs and different kinds of air pollution from roads nearby, we felt the confusion. Likewise, we found little to no information about things we as young people actually cared about, such as distance to the inner city, or ratings of nearby restaurants. Drawing on these experiences, we came up with a product we believed would make it intuitive to review an area before deciding to move in.

Our solution is to aggregate public data into one easy to understand rating, to be accessed by anyone looking for a new home. This rating would be leased out to real estate businesses as a badge to add to their estate listings, as well as private individuals wanting to customize the rating to match their own priorities.

All the user has to do is type in an address, select a standard rating from a roster of free attributes or pay a premium for more relevant ones, and get a rating from one to ten reflecting how the address scores. If they are curious, they can scroll down and view the individual attribute ratings.

2 Value Propositions

When moving to a new address, you'll want to know as much as possible about the area. However, the amount of data out there is often overwhelming. How will the consumer decide between a house with an impeccable energy rating in a crime ridden neighbourhood, and a house with asbestos in the ceiling in a safe neighbourhood? Our product aims to provide a simple way for people considering moving to an address to ensure the local environment around their new home is going to feel safe, healthy, and secure. We want to accomplish this by giving an address simple ratings for different metrics one might consider when moving into a new house - allowing for comparison between two addresses.

3 Customer segments

The first and most obvious customer is the end user, which is any private individual looking to move to a new address. The other customer segments will be in the corporate segment. These will include real-estate agencies, who will pay a membership to access our front-end app/widget to integrate the rating along with eye-catching attributes about the listing. It will also include insurance companies, which will also be a valuable customer to us. They already have access to the exact same information as us, and have already integrated the same data into their algorithms for calculating insurance premiums. However, as value is inherently subjective, the neighbourhood rating seen on a real-estate company website will sway the end user to value the neighbourhood in a skewed light, leading the insurance company to skew their rating to reflect the subjective value of the listing. Therefore, insurance customer is a valuable customer segment.

4 Customer relationships

Since the average Dane only moves six times on average in their lifetime, it will be nearly impossible to retain the end user with a subscription service. We expect the end user to pay to use the service once, and then neglect it until the next time they're looking to move. Therefore, the relationship with this segment will be a non-recurring payment structure in combination with a targeted marketing strategy aimed at customers frequenting real-estate websites and related internet searches. On the contrary, real estate and insurance companies will be recurring customers, which means we can implement a recurring revenue stream. Rather than relying on direct digital marketing, we want to attract one market leader such as EDC or Home for real estate, and Tryg or TopDanmark for insurance. Once it becomes apparent that our service strengthens their market position, we expect the competitors to follow suit.

5 Channels

The end user will be reached primarily through a web service, since users cannot be expected to download a piece of software in the form of an app or desktop application. However, the developers of the corporate customer

segments will need to have access to our software widget, as well as our API.

6 Key Activities

One key activity will be collecting data from as many sources as we can. At first this will be about implementing as many public data libraries as possible, but as our business grows, we might want to consider paying for private data, as well as collecting our own. As an example, Google provides useful API's for measuring distance to nearest hospitals using Google Maps, and restaurant reviews from Google Reviews.

Another key activity will be designing, implementing and maintaining an intuitive interface for the different kinds of end users. This will include a sleek web application for private individuals, an easily integratable widget with a rating badge for real estate firms, and an API for insurance companies. For each of these interfaces, the main focus will be balancing ease of use with exhaustiveness of the data we display.

7 Key Resources

The one key resource we rely on is all the different sources we gather data from. The most important one of these is the Danish Address Web API, which we use to fetch basic data such as area and region codes about the address provided by the user. However, the rest of our data will come from a diverse list of sources, meaning there will be no single point of failure for the different metrics. I.E. if Statistics Denmark decides to go dark out of the blue, the rating will simply have a few less attributes to account for, rather than rendering our service useless. Another key resource will be a server/hosting provider, since we don't have the capital to invest in our own servers.

8 Key Partners

Our business will greatly benefit from collaborating with the customers, extending to much more than just charging them for their membership. Our business will greatly benefit from having our rating integrated as seamlessly as possible into the real-estate agencies' front-end, meaning we will be in constant partnership with their software developers to optimize their UX, and in turn optimizing our own software. Similarly, we will also want to partner closely with the insurance providers, as their input is also highly valuable.

9 Cost structure

Where will we spend money?

- Software development
 - Developing and maturing the various software applications before the initial release will be the primary startup cost.
- Maintenance
 - Once released, the software will need to be maintained, bugs fixed for bugs, as well as adding further features.
- Servers
 - Hardware will be a big cost as well, regardless of whether we rent a domain somewhere like AWS, or provide server hosting ourselves.
- Marketing
 - Targeted ads for people who are looking at real-estate websites, or have made moving-related google searches. Additionally, we will need a salesperson to pitch our product and provide customer service to our corporate clients.
- Paid API keys
 - Some of the data we would like to collect is trapped behind a paywall, and usually the price depends on the amount of requests. This introduces a new cost for our product which will grow linearly with our user-base, and this is a hidden expense we need to keep a close eye on.

10 Revenue streams

Insurance firms will pay a subscription fee to have our rating integrated into their algorithms, meaning access to our API would be a paid service. The API would be directed at insurance companies, for requests with detailed information, since they will want detailed information on the different topics. For real-estate agents we will implement a paid widget/app for websites, that through visual representations of our data will be used to show their customers a neat interface with information about the given house/place. This would make it easier for real-estate agents to integrate into their front-end, since they won't have to create their own visualisations.

For private users we will have an affordable option, that via our website gives access to all filters and metrics, and additionally a compare function. This will be a one time payment, since a subscription option would not make sense for non-returning customers. As a standard for private users they will have access to 5-10 free metrics as well as 3 free searches, meaning if they want to look up more than 3 houses/places, they have to pay for the premium service.

11 Potential datasets

These are datasets we considered adding to our program, but for one reason or another didn't make it into the prototype.

- **Traffic fatalities:** Traffic fatalities in the city or area could be nice to know, especially for families with schoolkids
- **Performance of closest school:** Families might want to know the average grades of students from nearby schools
- **Covid-19 mortality:** For older people or people with weakened immune systems, seeing the mortality for covid-19 patients in the area might be important
- **Proximity to hospital:** Always nice to know in case of injury, but especially for elderly people and people with obstructive disabilities
- **Noise pollution:** No one wants to live right next to a train track, which might not be noticeable when first viewing the house if no trains come through
- **Energy rating of the house:** Saving energy saves money and the planet
- **Risk of flooding, harsh wind, snowstorms etc:** The environment can be a cruel mistress
- **Pollution of air or water:** Can be pulled from 'Danmarks Miljøportal'
- **Asbestos and Radon:** These harmful toxins might be present in the building, and no one wants that
- **Distance to closest park, shopping centre, inner city, public transport etc:** These could be interesting to a niche group of people, and we would therefore implement them at some point

The ones we implemented:

- **Crime data:** This metric is a collection of any crimes involving a victim which have been committed in the area. This includes everything from murders to breaking and entering.
- **Net data:** For us as data scientists, WiFi speed is crucial. Therefore, we included this metric.
- **Educational data:** Some people might only want to move to a "High Class" neighbourhood. This is the metric for those with (a bit too much) privilege.

12 The program

12.a The design idea

Our prototype asks the user for an address, calls a bunch of APIs fetching data about that address, scoring this data on a 1-10 metric, and then returning an easily digestible overall score. We also show scores for different individual attributes, such as "Internet connectivity" and "Crime rate". The prototype is implemented in Python3, as we are familiar with the language. Another boon arising from using Python3 is that we will be able to relatively easily expand our prototype to an MVP by use of a framework such as DJANGO, which will allow us to expand our prototype to handle HTTPS requests.

We decided to separate the fetching of the data from the scoring of the data, as to allow for future implementations of our product to score on new or different metrics which we currently discard after having received the data.

Currently, our prototype only works for addresses in Denmark, and we have not yet implemented any way to apply or exclude filters.

12.b Calling the API(s)

We delegated calling each API to its own function. These functions, which we call 'fetch' functions, all include the same 3 steps.

```
def fetch_data_DAWA(address):
    ## HERE BEGINS STEP 1 ##
    address, housenumber, postalnumber, city = address[0], address[1], address[2], address[3]
    data_to_post = "https://api.dataforsyningen.dk/datavask/adresser?"+\
        "betegnelse={}, {}, {} {}".format(address, housenumber, postalnumber, city)

    ## HERE BEGINS STEP 2 ##
    r = rq.get(data_to_post)
    if (r.status_code != 200):
        print("Not successful")

    ## HERE BEGINS STEP 3 ##
    json_response = pd.json_normalize(r.json())
    data = json_response['resultater']
    first_result = data.values[0]
    dataframe_result = pd.json_normalize(first_result)
    actual_address_page = rq.get(dataframe_result['adresse.href'][0])
    data = pd.json_normalize(actual_address_page.json())
    return(data['adgangsadresse.kommune.navn'].item(), data['adgangsadresse.id'].item())
```

Figure 1: Three steps in fetching data

1. **Designate parameters:** In the first step, we designate what data we want. In some cases, this is done by entering data into a JSON format, which will be posted to an endpoint. In others, it is done by entering the address into a URL, to which we will send a get request.
2. **Send the request:** Then, we fetch the data from the API using the 'Requests' library from Python. As mentioned, sometimes this means posting to an endpoint, while other times it simply means sending a get request to a URL.
3. **Process the data** When the response is received, we need to get it to a state where we can start using it. For this task, the Python package 'Pandas' proved extremely useful. Pandas allows for extracting both CSV

and JSON data from a string, and saving it in the form of a so-called 'Data Frame'. We then manipulate these data frames to only return the data we are interested in from the API call.

An example of these 3 steps can be seen in the code snippet in 1, which is an example of how we fetch data from the Danish Address's Web API. First, we insert the address in the URL string. Next, we send a get request to that URL. In return, we get data on one or more addresses, all of which more or less match the address in the query.

We want the first of these addresses, since this is the most accurate match. From the first match, we get a hyperlink to the data for that specific address. We can then send another get request to this hyperlink, and from the response extract the 'kommune' and address ID, which is returned.

12.c Scoring the data

For scoring the data we needed to find a way to transform the numbers from our data into a scale from 0-10. Since the data couldn't just be transformed directly into numbers from 0-10, we had to figure out a way to interpret the data and create a scale that represented a good outcome at 10, and bad outcome at 0. For internet this was easy. We simply check if the address is able to get above a certain download or upload speed, and deduct points from 10 depending on what the address was able to get.

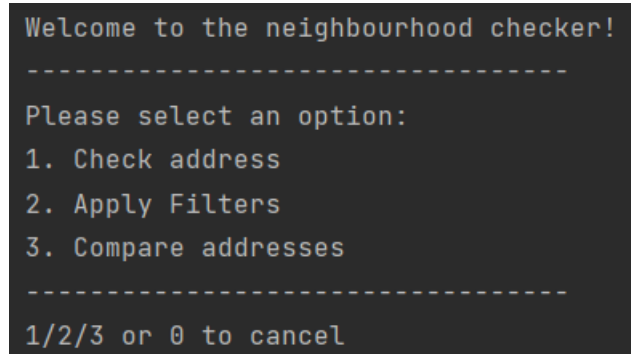
Evaluating the education score, was a little more complex. We want people with a high education to count more towards a good score than people with a low education. For this we created a function with a weight which increased with the level of education. We then divided the sum of education scores with the total number of educated people to get a score. For some cases like Frederiksberg and Copenhagen which are notoriously inhabited by academics and white collar workers, this yielded a score that was above 10. For these cases, we merely rounded the score down to 10.

For scoring the crime rate, we decided to compare with the national average. We wanted 0% crime rate per capita to be a perfect 10 points, and twice the national crime rate be 0 points. Therefore, we came up with the equation $Crime_rate = 10 - \min\left(\frac{communal_crime^2}{national_crime^2} * 5, 10\right)$. This formula means that if any city has the same crime rate as the national average, it scores 5 points. The crime rate is calculated based on population, so the size of the city is theoretically unimportant¹.

¹Though mashing a bunch of people together in big cities might increase crime rate in practice

13 Prototype interface

Since we had limited time to implement our prototype, we created a simple command line interface to show off the features of our prototype. It presents the user with 4 options to interact with our prototype.



```
Welcome to the neighbourhood checker!
-----
Please select an option:
1. Check address
2. Apply Filters
3. Compare addresses
-----
1/2/3 or 0 to cancel
```

Figure 2: Prototype interface

Our prototype provides the user with 3 options: Check address, apply filters, and compare addresses. The check address is the basic feature of our program, which takes one input and presents the user with a table of information, whereas the compare addresses function takes two addresses as argument and compares their scores in a table. The last feature, apply filters, has not been implemented yet, but would allow the user to choose between different metrics to only see the information of their choosing.

14 End user interface

Although we pride ourselves on our lovely command line application, one of our main value propositions is that the interface needs to be attractive and intuitive for the average person to use. Therefore, we have designed a sleek vision for our UI once we move past the prototype.

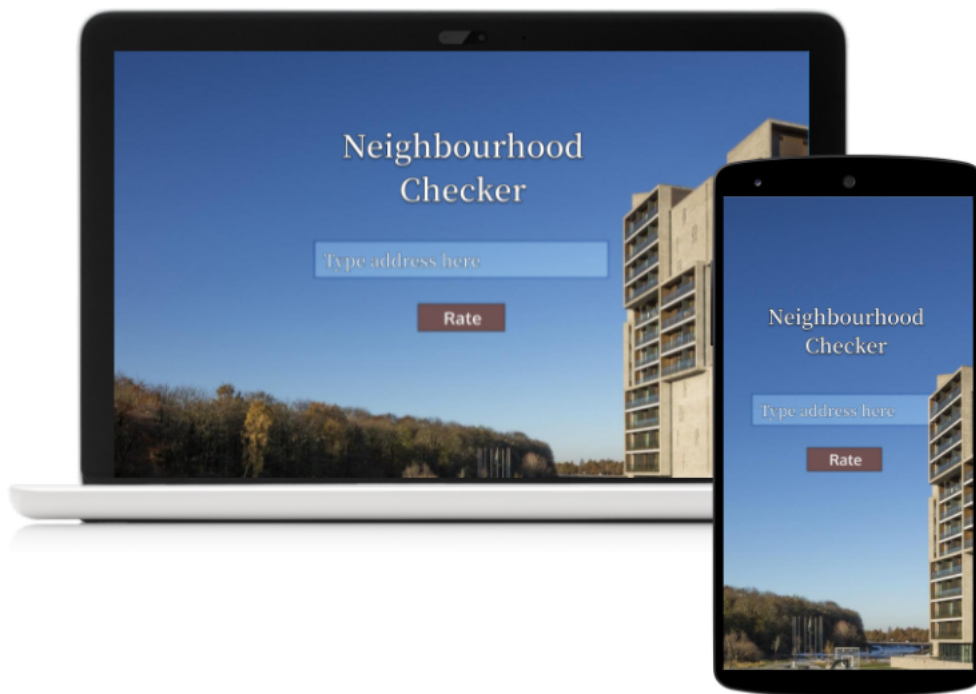


Figure 3: The landing page for the web application is one simple interface for the user to enter an address.

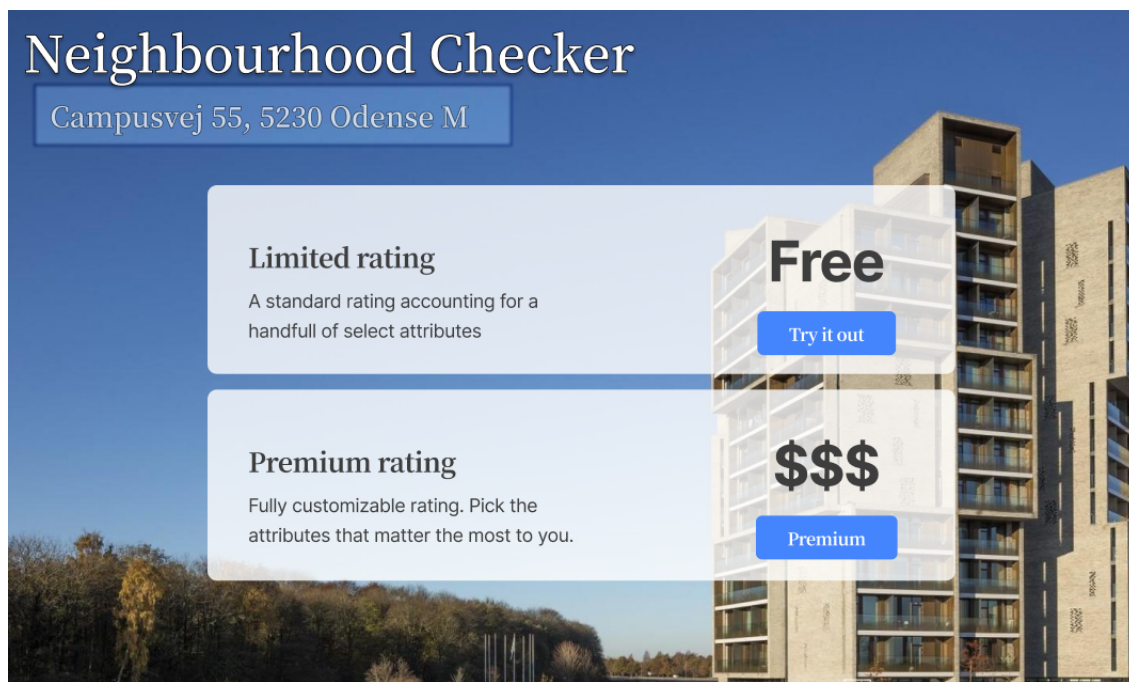


Figure 4: Next, they're presented with the two different payment options, which they proceed to choose from.

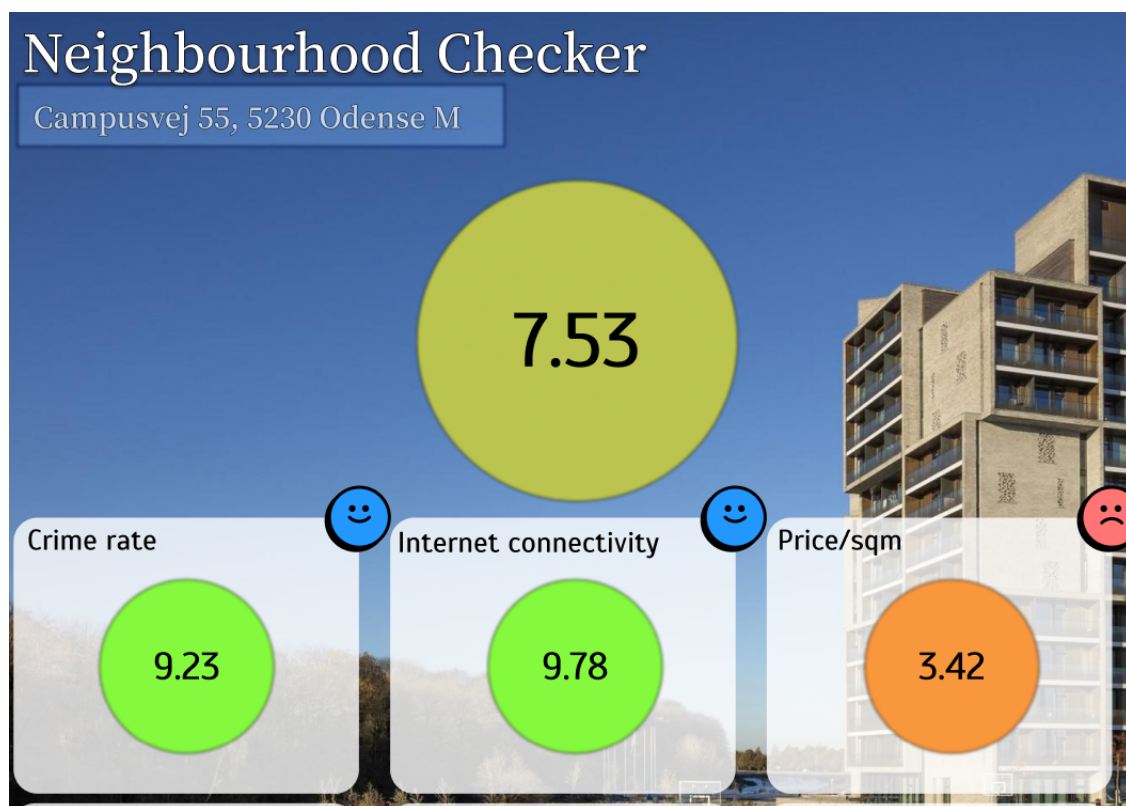


Figure 5: Finally, they are presented with the core of the user experience, the rating page. The purpose of this page is to draw the user to the overall score of the neighbourhood, while simultaneously inviting the user to scroll down to see the individual ratings.