My Experience @

KuDiBa - BREAKATHON

27-29 NOV 2020

Presented by Adam TAN

Agenda

Friday, 27.11.2020

18.00 – 20.00 Zoom Call: „BREAKATHON Kick-off “

- Getting to know each other.
- Introduction of the mentors and explanation of the hackathon challenges.
- Q & A.
- Afterwards, team-building for the individual challenges in breakout rooms.
- Mentors are still available as contact persons in main room of Zoom-call.

20.00 – 21.30 Meeting on Slack

- Move to the Slack „BREAKATHON Workspace “.
- Teams gather in the corresponding channels of their challenges.
- Teams organize themselves and collect ideas for solutions.
- Mentors are available until 22.00 o’clock to help with questions.
- Hackers make plan-to-action and distribute tasks in the teams.

Saturday, 28.11.2020

10.00 – 12.00 Zoom-Call: Presentation of ideas

- Meeting of all participants on Zoom.
- Teams present their ideas for solutions on which they will work for the next hours.
- Mentors ask questions and give tips.

12.00– 24.00 BREAKATHON auf Slack

- All participants and mentors meet on Slack.
- Mentors will be available for questions and assistance throughout the day.
- Teams work on the Hackathon-Challenges in the respective channels.
- A zoom room is always available for quick communication.

Sunday, 28.11.2020

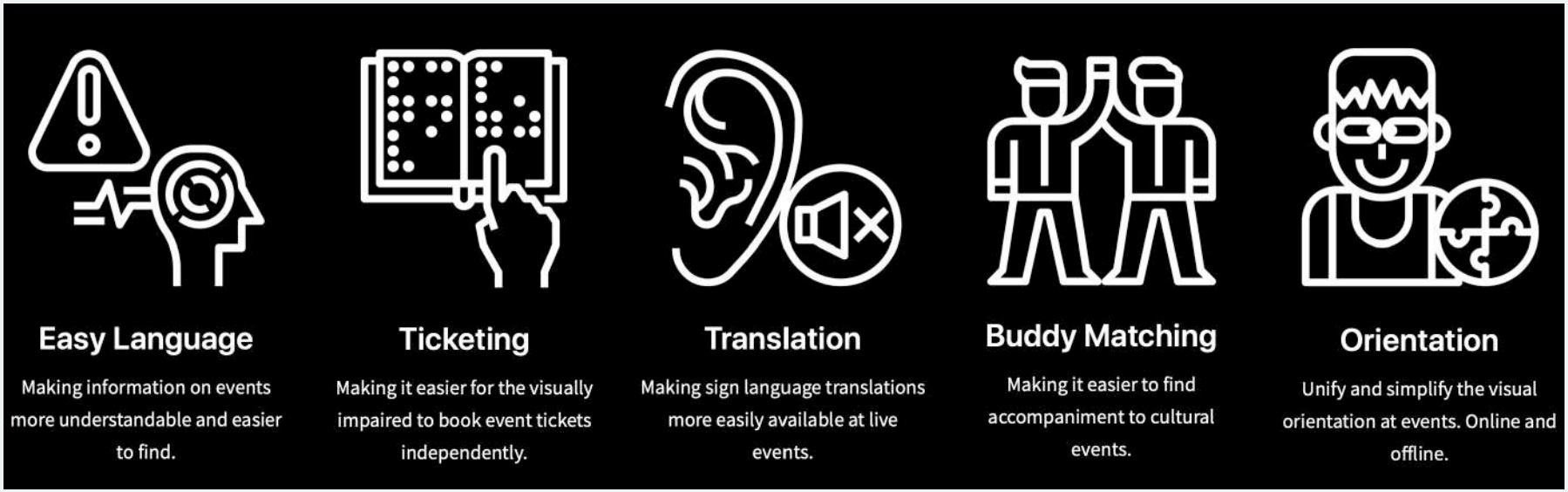
11.00 – 17.30 Hacking & Slacking

- Short feedback session to discuss any problems that might have occurred.
- Mentors will be available on Slack the rest of the day.
- Teams work on the Hackathon-Challenges in the channels.
- A zoom room is always available for quick communication.
- At 15.00 o’clock coordination between team and mentor for the respective challenge.

16.00 – 17.30 Final Presentations on Zoom

- Team-Presentations of the hacking results (concepts, prototypes, MVPs, etc.)
- 10 minutes team for each presentation/team.
- After each presentation 10 minutes Q & A with mentors und jury.

Challenges


Easy Language
Making information on events more understandable and easier to find.
Ticketing
Making it easier for the visually impaired to book event tickets independently.
Translation
Making sign language translations more easily available at live events.
Buddy Matching
Making it easier to find accompaniment to cultural events.
Orientation
Unify and simplify the visual orientation at events. Online and offline.

PLAIN LANGUAGE - CHALLENGE

Existing web app for to help writers to write in more understandable language:

http://www.hemingwayapp.com/

https://languagetool.org/

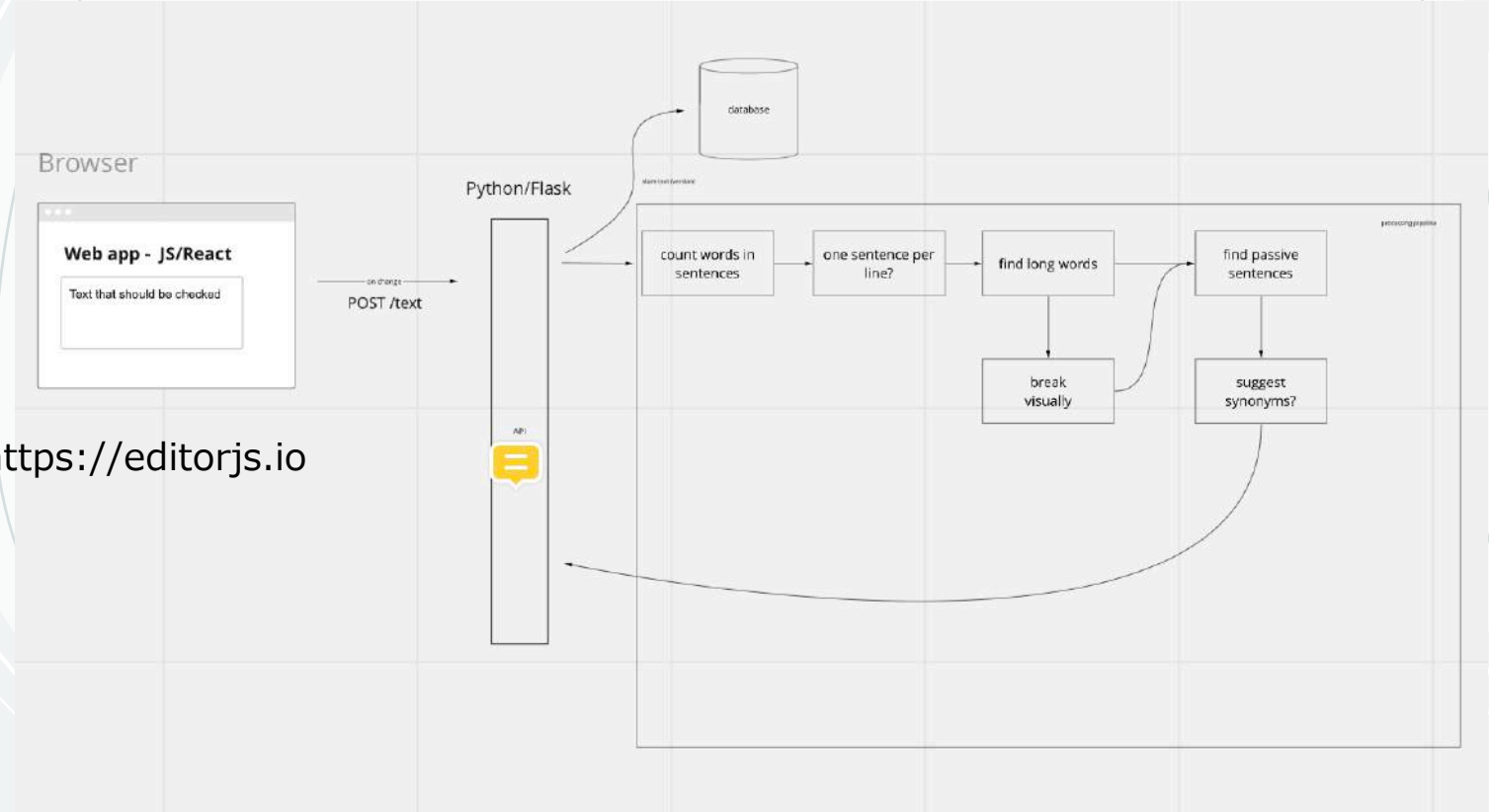
Example of website with easy language/plain language

https://www.dnb.de/EN/Kulturell/kulturell_node.html

https://mog61.de/ueber-uns/leichte-sprache/

Leichter Sprache

Initial Concept



https://editorjs.io

https://miro.com/app/board/o9J_ldOVV2s=/

Ideas

- A tool to assist writers with writing texts in plain language
- Combine differrent tools:
  - editor + readable score for words usage
  - words lookup for simple definition
  - picture lookup for words
- create a database to save plain language for further machine learning model to generate plain language

My Coding Experience

Natural Language Processing tools use to build the app - SpaCy , NLTK.

- **Steps:**
    1. **to split long text to sentences**
    2. **to count the length of each sentences**
    3. **to flag sentences more than 6 words**
    4. **to detect passive sentence**
    5. **to identity rare words usage in the sentence**

Existing words dataset available for generating score for german.

- Using pandas to clean the dataset

PLangTool/nlp.py

split long text to sentences

```
def sentence_length(text):
    '''
    split text into sentence by using regex

    param: text (string)

    return: length of each sentence

    '''
    s = re.split(r'(?<=[^A-Z].[.?]) +(?=[A-Z])', text)
    length = []
    for i in range(len(s)):
        length.append(len(s[i].split(' ')))

    return length
```

count the lengths of each sentences

```
def long_sentence(text,number_of_words:int):
    '''
    to show the text has long sentence

    param: text (string)
    param: number_of_words (integer)

    return: long sentences

    '''
    length = sentence_length(text)

    s = re.split(r'(?<=[^A-Z].[.?]) +(?=[A-Z])', text)

    print(f'The following sentences have more than {number_of_words} words:')
    print(f'Die folgenden Sätze haben mehr als {number_of_words} Wörter:')

    sentence = []

    for i in range(len(length)):
        if length[i] > number_of_words: # number of words
            sentence.append(s[i])
            print(f'Sentence {i+1} ({length[i]} words): {s[i]}')
        else: pass

    return sentence
```

detect passive sentence

```
def passive_en(sentences):

  '''
  using spacy to check parser if passive verb is used in english

  return passive sentences

  '''
  nlp = spacy.load('en_core_web_sm')
  for i, sent in enumerate (sentences,1):
    doc = nlp(sent)
    for j ,word in enumerate (doc,1):
      if word.dep_ == 'auxpass':
        print(f'Sentence {i} is a passive sentence with word {j} is passive verb : {sent}')
    return

def passive_de(sentences):
    '''
    using spacy to check parser if passive verb is used in german

    return passive sentences

    '''
    nlp = spacy.load('de_core_news_sm')
    for i, sent in enumerate (sentences,1):
        doc = nlp(sent)
        for j ,word in enumerate (doc,1):
            if word.dep_ == 'og':
                print(f'Satz {i} ist ein passiver Satz mit Wort {j} ist Passivverb : {sent}')
    return
```

Processing words

```
def remove_numbers(text):
    '''remove numbers from the text'''
    text = re.sub(r'\d+', '', text)
    return text

def split_word(text):
    '''split words from the text'''
    return text.split()

def remove_punctuation(words):
    """Remove punctuation from the text"""
    new_words = []
    for word in words:
        new_word = re.sub(r'[^\w\s]', '', word)
        if new_word != '':
            new_words.append(new_word)
    return new_words

def to_lowercase(words):
    """Convert all characters to lowercase from list of words"""
    new_words = []
    for word in words:
        new_word = word.lower()
        new_words.append(new_word)
    return new_words
```

```
def normalize(sentence):
    words = []
    for text in range(len(sentence)):
        word = remove_numbers(sentence[text])
        word = split_word(word)
        word = remove_punctuation(word)
        word = to_lowercase(word)
        #word = stemmed_word(word)
        words.append(word)
    return words
```

Scoring words

```python
import numpy as np
import pandas as pd

de_vocab = pd.read_csv('https://raw.githubusercontent.com/kaiyungtan/PLangTool/adam/de_vocab_rev1.csv')

filt  = de_vocab['pos'] == 'NN'

de_vocab_nn = de_vocab[filt]

# mean normalization for frequency
max_value = de_vocab_nn['frequency'].max()
min_value = de_vocab_nn['frequency'].min()
de_vocab_nn['frequency_normalized_nn'] = (de_vocab_nn['frequency'] - min_value) / (max_value - min_value)

de_vocab_nn = de_vocab_nn.reset_index()
```

```python
def score(text):

    score = []

    for word in range(len(text)):
        for i in range(de_vocab_nn.shape[0]):
            if text[word] == de_vocab_nn['word'][i]:
                score.append({text[word]:float(de_vocab_nn['frequency_normalized_nn'][i])})
                break
            continue

    return score
```

identity rare words

```python
def check_rare_word(rare_word):

    for i in range(len(rare_word)):
        key = list(rare_word[i].keys())[0]
        if rare_word[i][key] < 0.000402:
            print(f'This is a rare word: {key}')
    return
```

Takeaway

The takeaway from the Breakathon:

1. **To clarify the purpose of the challenge right from the start**
2. **To identify expertise in certain domain (NLP) to participate**
3. **To communicate progress by using tools available**
4. **To experience working with/for needs of people with special needs**
5. **To work virtually with others**
6. **To have fun!**

Other Challenges

Buddy matching:

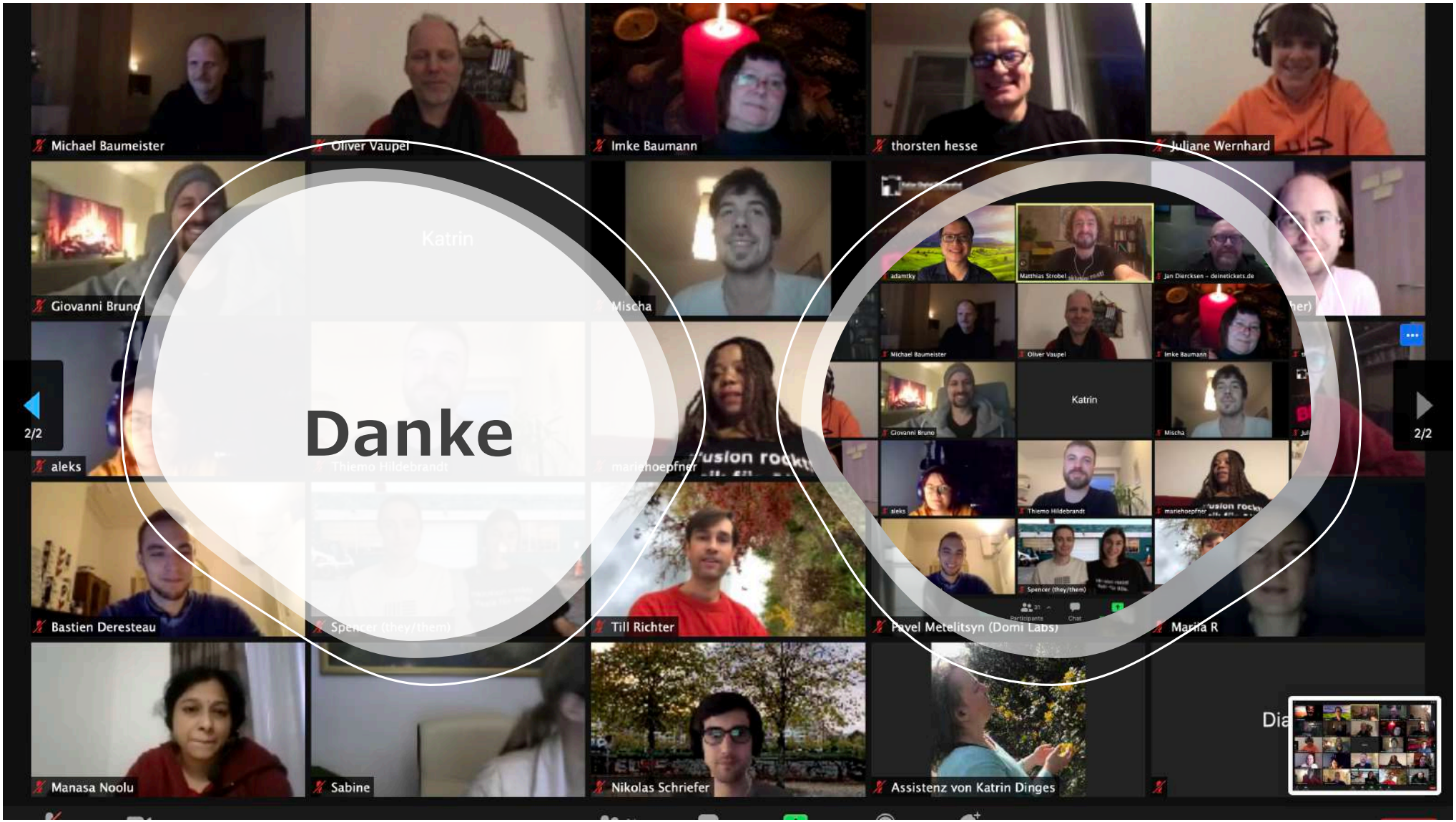
https://wingbuddy.herokuapp.com

Ticketing:

https://7b85fbe5548b.ngrok.io

Accessibility guide

https://cms.fokus-d.de/preview/1b8a34fe?device=all


Michael Baumeister
Oliver Vaupel
Imke Baumann
thorsten hesse
Juliane Wernhard
Giovanni Bruno
Mischa
Katrin
2/2
aleks
Thiemo Hildebrandt
mariehoepfner
Bastien Deresteau
Spencer (they/them)
Till Richter
Pavel Metelitsyn (Domi Labs)
Marita R
Manasa Noolu
Sabine
Nikolas Schriefer
Assistenz von Katrin Dinges
Danke