



# Dr. Tobit Esch

KI seit 8 Jahren:

- Softwareentwicklung
- Digitalisierungsberatung
- Förderberatung

Geschäftsführer und Gründer der  
7 Hills Neo GmbH seit 2018

<https://www.linkedin.com/in/tobitesch>



[https://www.destatis.de/DE/Presse/Pressemitteilungen/2023/11/PD23\\_453\\_52911.html](https://www.destatis.de/DE/Presse/Pressemitteilungen/2023/11/PD23_453_52911.html)

12%



Was ist GPT?



# Was ist GPT?

**Large Language Model**

Universal approximator

Generative AI

Neural Network

Artificial intelligence

Simulator

Text predictor

Writing assistant

Content generator

Agent

Chatbot

Transformer

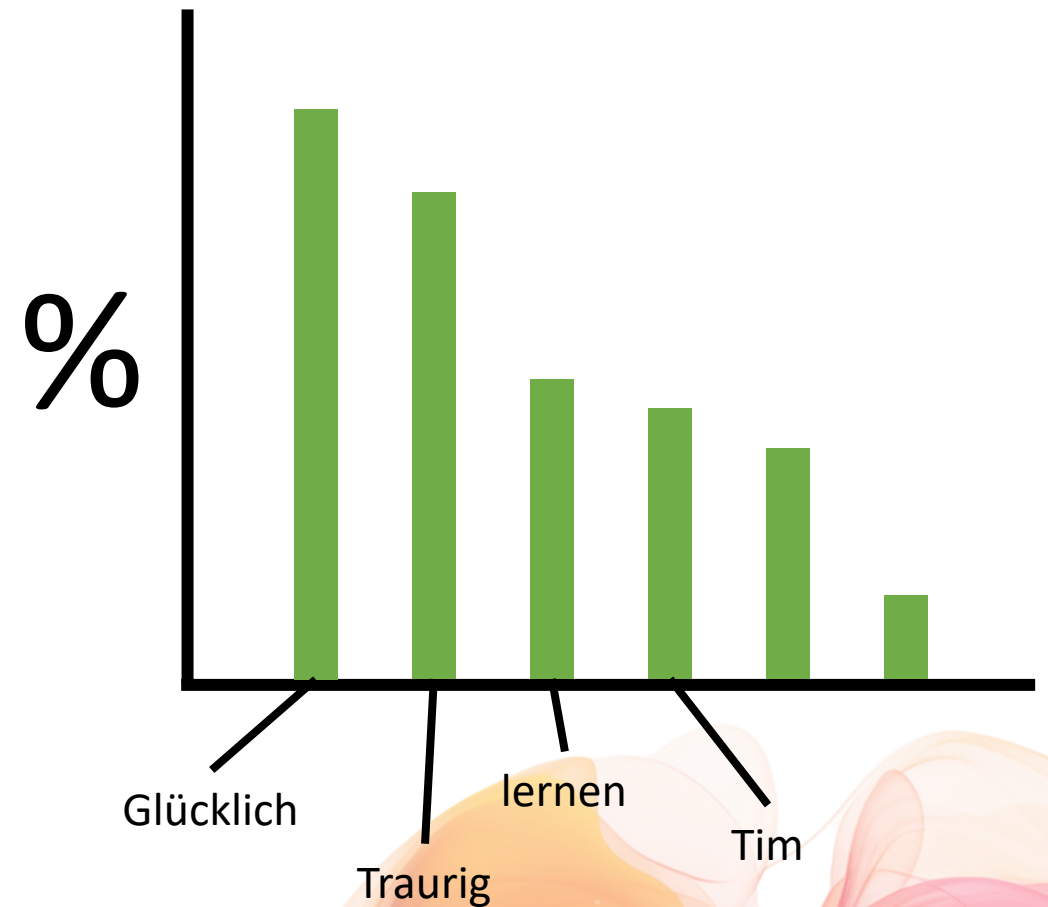




# Language Models

## Berechnet Wahrscheinlichkeiten

- Was folgt „Ich bin ...“?
- Vokabeln = alle Wörter
- kann das vorhergesagte Wort wiederholt abfragen, um Text zu erzeugen

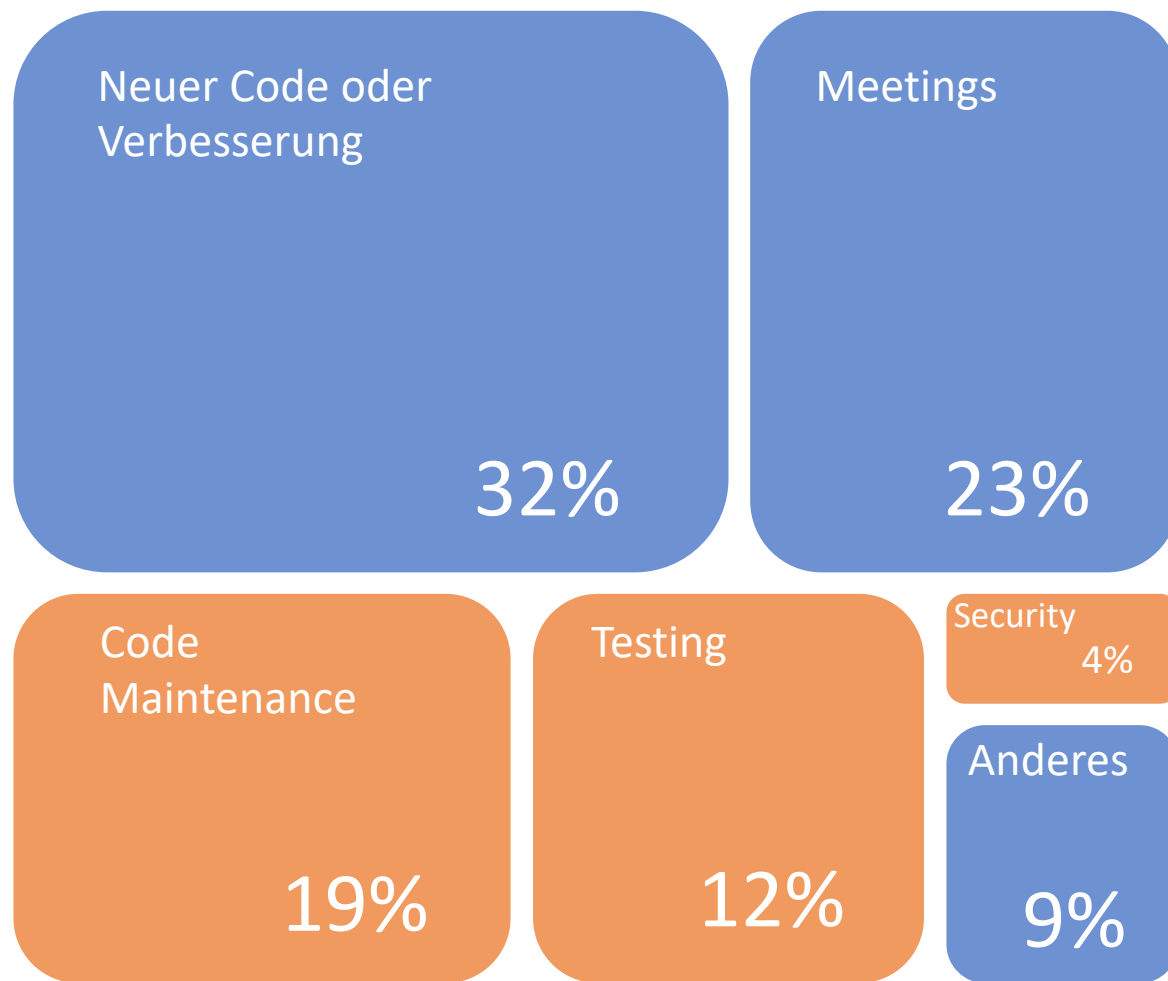


# Language Models

## Berechnet Wahrscheinlichkeiten

- Was folgt „Ich bin ...“?
- Vokabeln = alle Wörter
- kann das vorhergesagte Wort wiederholt abfragen, um Text zu erzeugen





## Suche nach Lösungen

63% 30-60 Minuten

25% 1-2 Stunden

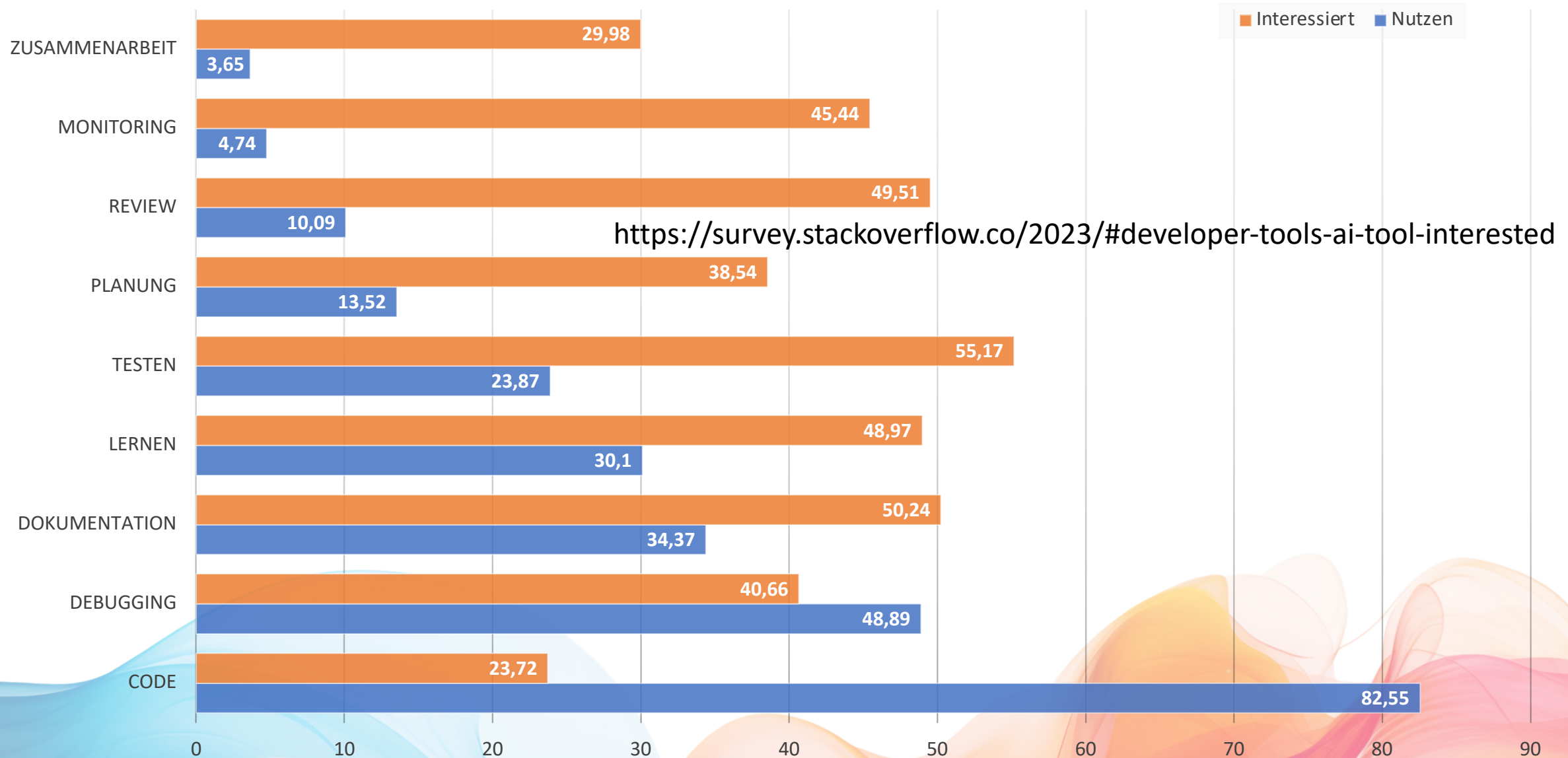
## Fragen beantworten

48% 30-60 Minuten

18% 1-2 Stunden

<https://thenewstack.io/how-much-time-do-developers-spend-actually-writing-code/>

# AI im Workflow





```
1 @login_required(login_url="/tournament/admin/login/")
2 def tournament_access_areas_all(request, tournament_uuid=None):
3     user = request.user
4     tournaments = TournamentAdmin.objects.filter(user=user)
5     tournaments = [tournament.tournament for tournament in tournaments]
6     if not tournament_uuid:
7         messages.error(request, "No tournament selected.")
8         return HttpResponseRedirect("/tournament/admin/tournament-select")
9
10    tournament = Tournament.objects.get(uuid=tournament_uuid)
11    tournament_access_areas = TournamentAccessArea.objects.filter(tournament=tournament)
12    return render(
13        request,
14        "tournamentadmin/tournament_access_areas_all.html",
15        context={
16            "tournament": tournament,
17            "tournament_access_areas": tournament_access_areas,
18            "tournaments": tournaments,
19        },
20    )
```

# Beschreiben der Funktion



ChatGPT

Sure, here's the detailed description of the function `tournament_access_areas_all`.

The function `tournament_access_areas_all` is a Django view function that is used to handle a web request and produce a web response. This function's purpose is to display all access areas of a selected tournament in a web page. It does the following:

1. **User Authentication:** With the `@login_required` decorator, it first checks if the user is authenticated. If the user is not authenticated, it redirects them to the login page (`/tournament/admin/login/`).
2. **Fetch User's Tournaments:** Then it fetches the list of tournaments associated with the currently logged in user. The `TournamentAdmin` model seems to be a many-to-many association table that links the `User` model with the `Tournament` model.
3. **Tournament Selection:** The function takes `tournament_uuid` as an optional argument. If `tournament_uuid` is not provided in the request, it returns an error message "No tournament selected." and redirects the user to the tournament selection page (`/tournament/admin/tournament-select`).
4. **Fetch Selected Tournament:** If `tournament_uuid` is provided, it fetches the corresponding tournament using the `Tournament` model.

5. **Fetch Access Areas:** Then it fetches all the access areas associated with the selected tournament using the `TournamentAccessArea` model. An access area likely refers to some sort of privileged area or feature within a tournament.
6. **Render Template:** Finally, it renders the `tournament_access_areas_all.html` template, passing the selected tournament, its access areas, and the list of tournaments associated with the current user as context data to the template.

This function requires a user to be authenticated to access it, and displays information related to tournaments and their access areas. This function seems to be part of the administration views of a web application dealing with tournaments, likely for some type of competitive event.



You

how should I comment this function to help other developers and can you write me a test for this function

# Lernen von Codebasis 49%

# Code Dokumentation



ChatGPT

To add comments to this function, you would usually detail the purpose of the function, its arguments, its main operations, and its return. Here's how you might do it:

python

Copy code

```
@login_required(login_url="/tournament/admin/login/")
def tournament_access_areas_all(request, tournament_uuid=None):
    """
    A view function to fetch and display all access areas of a selected tournament.

    Args:
        request (HttpRequest): The request object.
        tournament_uuid (str, optional): The UUID of the tournament. Defaults to None.

    Steps:
        1. Fetches all tournaments associated with the current user.
        2. Checks if a tournament_uuid is provided.
           - If not, returns an error message and redirects to tournament selection.
        3. Fetches the tournament corresponding to the provided UUID.
        4. Fetches all access areas associated with this tournament.
        5. Renders 'tournament_access_areas_all.html' template with the tournament, a
           and user's tournaments as context.

    Returns:
        HttpResponse: The response object with rendered HTML.
    """
```

```
# Fetch user's tournaments
user = request.user
tournaments = TournamentAdmin.objects.filter(user=user)
tournaments = [tournament.tournament for tournament in tournaments]

# Check if tournament UUID is provided
if not tournament_uuid:
    messages.error(request, "No tournament selected.")
    return HttpResponseRedirect("/tournament/admin/tournament-select")

# Fetch selected tournament and its access areas
tournament = Tournament.objects.get(uuid=tournament_uuid)
tournament_access_areas = TournamentAccessArea.objects.filter(tournament=tournament)

# Render the template with context
return render(
    request,
    "tournamentadmin/tournament_access_areas_all.html",
    context={
        "tournament": tournament,
        "tournament_access_areas": tournament_access_areas,
        "tournaments": tournaments,
    },
)
```

# Dokumentation von Code 50%



# Schreibe Tests

For the unit test, you can use Django's built-in testing framework. Please replace the `UserFactory`, `TournamentFactory` and `TournamentAdminFactory` with your own factory definitions or create instances of the models manually if you're not using a factory library like `factory_boy`.

python

Copy code

```
from django.test import TestCase, RequestFactory
from django.contrib.auth.models import AnonymousUser
from django.urls import reverse
from django.contrib.messages.storage.fallback import FallbackStorage
from yourapp.models import User, TournamentAdmin, Tournament, TournamentAccessArea
from yourapp.views import tournament_access_areas_all
from yourapp.factories import UserFactory, TournamentFactory, TournamentAdminFactory

class TournamentAccessAreasAllViewTest(TestCase):
    def setUp(self):
        self.factory = RequestFactory()
        self.user = UserFactory.create()
        self.tournament = TournamentFactory.create()
        self.tournament_admin = TournamentAdminFactory.create(user=self.user, tournam
```

```
def test_tournament_access_areas_all(self):
    request = self.factory.get(reverse('tournament_access_areas_all', kwargs={'to
    request.user = self.user
    setattr(request, 'session', 'session')
    messages = FallbackStorage(request)
    setattr(request, '_messages', messages)

    response = tournament_access_areas_all(request, tournament_uuid=self.tourname

    self.assertEqual(response.status_code, 200)
    self.assertTemplateUsed(response, 'tournamentadmin/tournament_access_areas_al

def test_tournament_access_areas_all_no_uuid(self):
    request = self.factory.get(reverse('tournament_access_areas_all'))
    request.user = self.user
    setattr(request, 'session', 'session')
    messages = FallbackStorage(request)
    setattr(request, '_messages', messages)

    response = tournament_access_areas_all(request)

    self.assertEqual(response.status_code, 302)
    self.assertRedirects(response, '/tournament/admin/tournament-select')
```

# Testing Code 55%



# Verbesserungen & Lernen




ChatGPT

Sure, here are some suggestions to improve the code and your practices as a developer:

1. **Avoid Multiple Database Hits:** In your current code, you are hitting the database twice to get the list of tournaments associated with the current user. First, you fetch the list of `TournamentAdmin` objects associated with the user, and then you extract the tournament from each of these objects. This can be done in one step by using the `values_list` method with the `flat` parameter set to `True`. This returns a flat list of the desired value, reducing the number of database hits.

```
python
```

 Copy code

```
tournaments = TournamentAdmin.objects.filter(user=user).values_list('tournament', flat=True)
```

# Verbesserungen & Lernen

1. **Handle Exceptions:** When you use `get()` method to fetch a tournament, it will raise a `DoesNotExist` exception if no matching record is found. You should handle this exception and show an appropriate message to the user.

```
python Copy code  
  
try:  
    tournament = Tournament.objects.get(uuid=tournament_uuid)  
except Tournament.DoesNotExist:  
    messages.error(request, "The selected tournament does not exist.")  
    return HttpResponseRedirect("/tournament/admin/tournament-select")
```

2. **Use Class-Based Views:** Instead of using function-based views, consider using Django's class-based views (CBVs). They're reusable and provide a lot of common functionalities right out of the box.
3. **Code Formatting:** Follow PEP-8, the Python style guide, to ensure your code is easily readable by other developers.
4. **Writing Tests:** You should write unit tests and functional tests for your code. Tests help you make sure your code is working as expected, and they're also useful for other developers to understand what your code is supposed to do.
5. **Use Descriptive Variable Names:** Good variable names can make your code self-explanatory, reducing the need for comments.
6. **Code Review:** Participate in code reviews. They are an excellent way to learn from other developers and to get different perspectives on your code. This can provide insights on how to write cleaner and more efficient code.

7. **Keep Learning:** Always be open to learning new things. Technologies and best practices keep evolving. Subscribe to relevant blogs, join Python and Django communities, and attend webinars and conferences if possible.
8. **Master Debugging:** Effective debugging can save a lot of time and effort. Learn to use Python debugging tools effectively.
9. **Understand the Domain:** The better you understand the problem you're solving, the better your software will be. Take the time to understand the domain of your application.
10. **Documentation:** Good documentation makes your code accessible to other developers. Include docstrings for your modules, classes, and functions. If you're working on a larger project, consider creating separate documentation using tools like Sphinx.

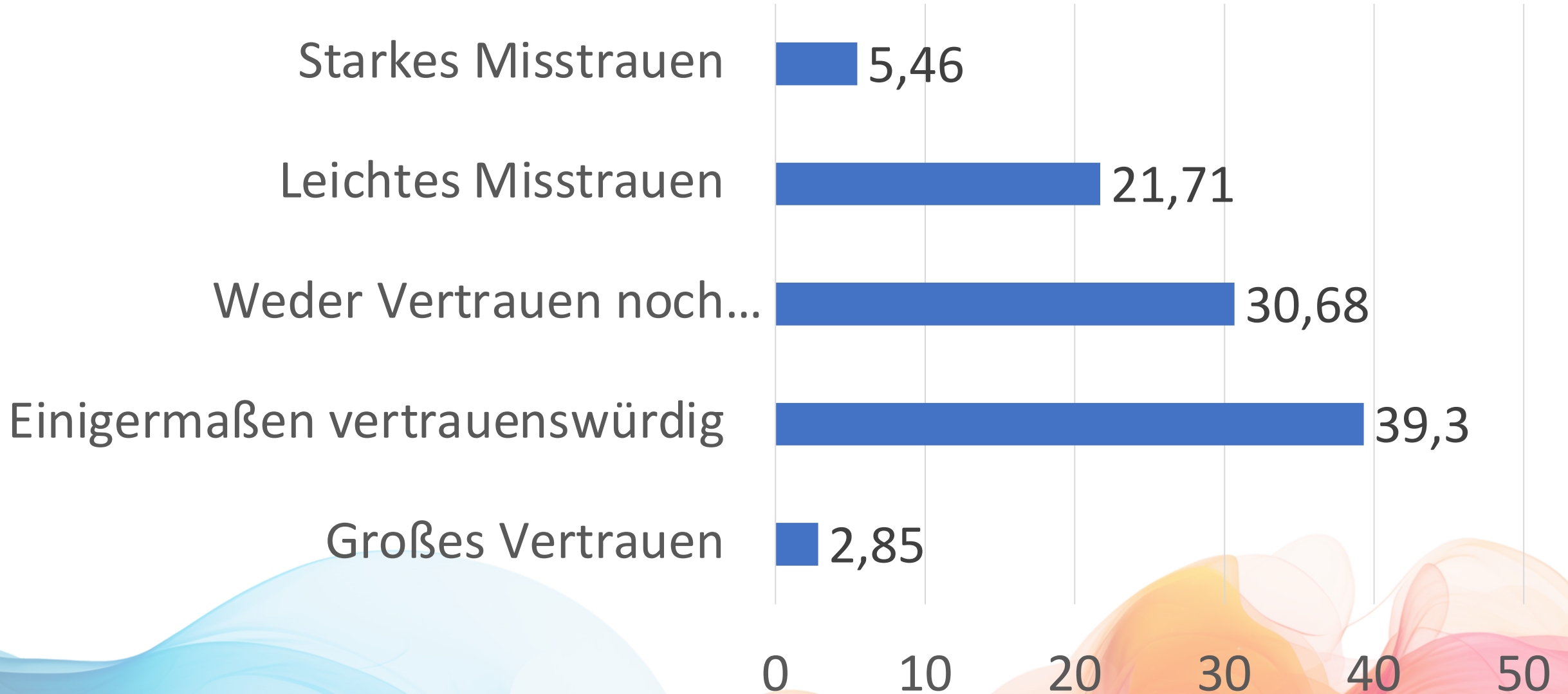
**T** You

can you provide me the best code version of this function you are able to do?

# Vertrauen



# Genauigkeit von AI Tools





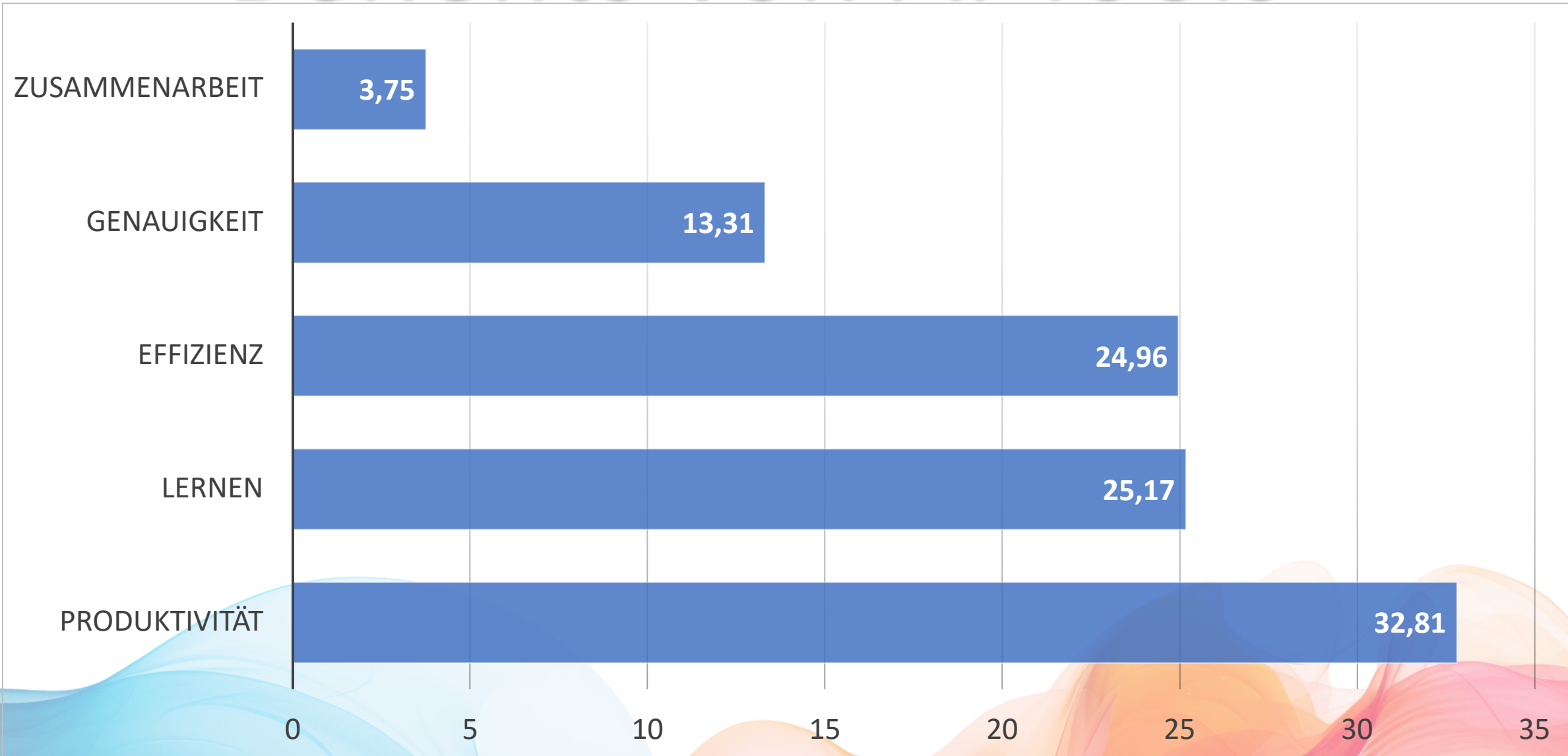
<https://www.zdnet.com/article/the-most-copied-stackoverflow-java-code-snippet-contains-a-bug/>

# The most copied StackOverflow Java code snippet contains a bug

Nine years later, developer corrects code snippet.



# Benefits von AI Tools



Hallo, ich habe ein Programm geschrieben. Dieses Programm hat unterschiedliche Ausgaben, wenn ich es mit gcc oder mit arm-linux kompiliere, kannst du mir erklären warum?

```
1 #include <stdio.h>
2
3 #include <stdbool.h>
4
5 int createMessage(bool success) {
6     const char * pointer;
7     if (success) {
8         const char positiveAnswer[] = "positive";
9         pointer = positiveAnswer;
10    } else {
11        const char negativeAnswer[] = "negative";
12        pointer = negativeAnswer;
13    }
14    printf("answer: %s \n ", pointer);
15 }
16 int main(int argc, char
17 const * argv[]) {
18     createMessage(true);
19     return 0;
20 }
```

#### ChatGPT

Es gibt mehrere Gründe, warum dieses Programm unterschiedliches Verhalten aufzeigen könnte, wenn es mit verschiedenen Compilern oder für unterschiedliche Architekturen kompiliert wird. Hier sind ein paar Dinge, die du überprüfen könntest:

#### 1. Lebensdauer von lokalen Variablen

In der `createMessage`-Funktion werden `positiveAnswer` und `negativeAnswer` lokal deklariert. Diese Variablen haben eine Lebensdauer, die nur innerhalb des Blocks gilt, in dem sie deklariert werden. Nachdem der Block abgeschlossen ist, ist der Zugriff auf diese Variablen undefiniert, da sie nicht mehr existieren. Verschiedene Compiler und Architekturen können unterschiedliches Verhalten bei Zugriffen auf nicht mehr existierende Variablen aufweisen. Das kann dazu führen, dass unterschiedliche Werte oder sogar Programmabstürze auftreten.

#### 2. Compiler-Optimierungen

GCC und ARM-Linux-Compiler können unterschiedliche Optimierungsstrategien verwenden, die das Verhalten des Programms beeinflussen können.

Hallo, ich habe ein Programm geschrieben. Dieses Programm hat unterschiedliche Ausgaben, wenn ich es mit gcc oder mit arm-linux kompiliere, kannst du mir erklären warum?

```
1 #include <stdio.h>
2
3 #include <stdbool.h>
4
5 int createMessage(bool success) {
6     const char * pointer;
7     if (success) {
8         const char positiveAnswer[] = "positive";
9         pointer = positiveAnswer;
10    } else {
11        const char negativeAnswer[] = "negative";
12        pointer = negativeAnswer;
13    }
14    printf("answer: %s \n ", pointer);
15 }
16 int main(int argc, char
17 const * argv[]) {
18     createMessage(true);
19     return 0;
20 }
```

### Lösung

Um sicherzustellen, dass der Zeiger auf einen definierten und existierenden Wert zeigt, könntest du statische Zeichenketten oder Zeiger auf Zeichenketten-Literale verwenden:

```
c Copy code
#include <stdio.h>
#include <stdbool.h>

int createMessage( bool success ) {
    const char *pointer;
    if( success ) {
        pointer = "positive";
    } else {
        pointer = "negative";
    }
    printf( "answer: %s \n", pointer );
}

int main( int argc, char const *argv[] ) {
    createMessage( true );
    return 0;
}
```

In diesem Fall sind "positive" und "negative" Zeichenketten-Literale, und sie haben eine Lebensdauer, die für die gesamte Ausführungszeit des Programms gültig ist.



# Schreibarbeit minimieren

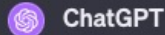
```
1- [
2- {
3   "leagueId": "agsed",
4   "queueType": "RANKED_SOLO5x5",
5   "tier": "DIAMOND",
6   "rank": "I",
7   "summonerId": "XXX",
8   "summonerName": "Midlane",
9   "leaguePoints": 65,
10  "wins": 56,
11  "losses": 39,
12  "veteran": false,
13  "inactive": false,
14  "freshBlood": false,
15  "hotStreak": false
16 }
17 ]
```

```
1- [
2- {
3   "puuid": "ASDFGA",
4   "leagueId": "slskb",
5   "queueType": "RANKEDTFT",
6   "tier": "DIAMOND",
7   "rank": "III",
8   "summonerId": "XXX",
9   "summonerName": "Midlane",
10  "leaguePoints": 28,
11  "wins": 42,
12  "losses": 29,
13  "veteran": false,
14  "inactive": false,
15  "freshBlood": false,
16  "hotStreak": false
17 }
18 ]
```

# Schreibarbeit minimieren

```
1- [
2- {
3   "leagueId": "agsed",
4   "queueType": "RANKED_SOLO5x5",
5   "tier": "DIAMOND",
6   "rank": "I",
7   "summonerId": "XXX",
8   "summonerName": "Midlane",
9   "leaguePoints": 65,
10  "wins": 56,
11  "losses": 39,
12  "veteran": false,
13  "inactive": false,
14  "freshBlood": false,
15  "hotStreak": false
16 }
17 ]
```

```
1- [
2- {
3   "puuid": "ASDFGA",
4   "leagueId": "slskb",
5   "queueType": "RANKEDTFT",
6   "tier": "DIAMOND",
7   "rank": "III",
8   "summonerId": "XXX",
9   "summonerName": "Midlane",
10  "leaguePoints": 28,
11  "wins": 42,
12  "losses": 29,
13  "veteran": false,
14  "inactive": false,
15  "freshBlood": false,
16  "hotStreak": false
17 }
18 ]
```



ChatGPT

The provided JSON r

Here's a Django moc

```
python
```

```
from django.db
```

```
class SummonerL
```

```
    puuid = mod
```

```
    leagueId =
```

```
    queueType =
```

```
    tier = mode
```

```
    rank = mode
```

```
    summonerId :
```

```
    summonerNam
```

```
    leaguePoints = models.IntegerField()
```

```
    wins = models.IntegerField()
```

```
    losses = models.IntegerField()
```

```
    veteran = models.BooleanField(default=False)
```

```
    inactive = models.BooleanField(default=False)
```

```
    freshBlood = models.BooleanField(default=False)
```

```
    hotStreak = models.BooleanField(default=False)
```

```
    def __str__(self):
```

```
        return self.summonerName + " - " + self.queueType
```

A few points to note:

1. I've assumed the provided JSONs represent the same data structure, so I've created a single model.
2. The `puuid` field appears only in the first JSON response, and not in the second. So, I've made it nullable with `blank=True, null=True`. If all entries should have this field, you can remove these arguments.
3. The length of the `CharField` fields are set to 256 as a generic value. You might want to adjust these according to the actual expected length of the data.
4. The boolean fields have a default value of `False`.

To use this model in your Django project, you would need to:

1. Add it to a Django app's `models.py` file.
2. Include that app in the `INSTALLED_APPS` list in the Django project's settings.
3. Run the migrations (`python manage.py makemigrations` followed by `python manage.py migrate`) to create the corresponding database table.

# Schreibarbeit minimieren

A few points to note:

1. I've assumed the provided JSONs represent the same data structure, so I've created a single model.
2. The `puuid` field appears only in the first JSON response, and not in the second. So, I've made it nullable with `blank=True, null=True`. If all entries should have this field, you can remove these arguments.
3. The length of the `CharField` fields are set to 256 as a generic value. You might want to adjust these according to the actual expected length of the data.
4. The boolean fields have a default value of `False`.

To use this model in your Django project, you would need to:

1. Add it to a Django app's `models.py` file.
2. Include that app in the `INSTALLED_APPS` list in the Django project's settings.
3. Run the migrations (`python manage.py makemigrations` followed by `python manage.py migrate`) to create the corresponding database table.

 ChatGPT

The provided JSON responses seem to represent some ranked league data. Here's a Django model to represent the data:

```
python

from django.db import models

class SummonerLeague(models.Model):
    puuid = models.CharField(max_length=256, blank=True, null=True)
    leagueId = models.CharField(max_length=256)
    queueType = models.CharField(max_length=256)
    tier = models.CharField(max_length=256)
    rank = models.CharField(max_length=256)
    summonerId = models.CharField(max_length=256)
    summonerName = models.CharField(max_length=256)
    leaguePoints = models.IntegerField()
    wins = models.IntegerField()
    losses = models.IntegerField()
    veteran = models.BooleanField(default=False)
    inactive = models.BooleanField(default=False)
    freshBlood = models.BooleanField(default=False)
    hotStreak = models.BooleanField(default=False)

    def __str__(self):
        return self.summonerName + " - " + self.queueType
```



# Schreibarbeit minimieren



ChatGPT

If the goal is to track changes in these values every minute, it would be useful to store a timestamp with each record to track when the changes occurred. Additionally, you might consider setting up a model that represents changes or history for a given summoner, rather than just the current values.

Here's a suggested Django model structure to track the history of these values:

python

Copy code

```
from django.db import models

class Summoner(models.Model):
    summonerId = models.CharField(max_length=256, unique=True)
    summonerName = models.CharField(max_length=256)

    def __str__(self):
        return self.summonerName
```

```
class SummonerLeagueHistory(models.Model):
    summoner = models.ForeignKey(Summoner, on_delete=models.CASCADE)
    timestamp = models.DateTimeField(auto_now_add=True)
    queueType = models.CharField(max_length=256)
    tier = models.CharField(max_length=256)
    rank = models.CharField(max_length=256)
    leaguePoints = models.IntegerField()
    wins = models.IntegerField()
    losses = models.IntegerField()
    veteran = models.BooleanField(default=False)
    inactive = models.BooleanField(default=False)
    freshBlood = models.BooleanField(default=False)
    hotStreak = models.BooleanField(default=False)

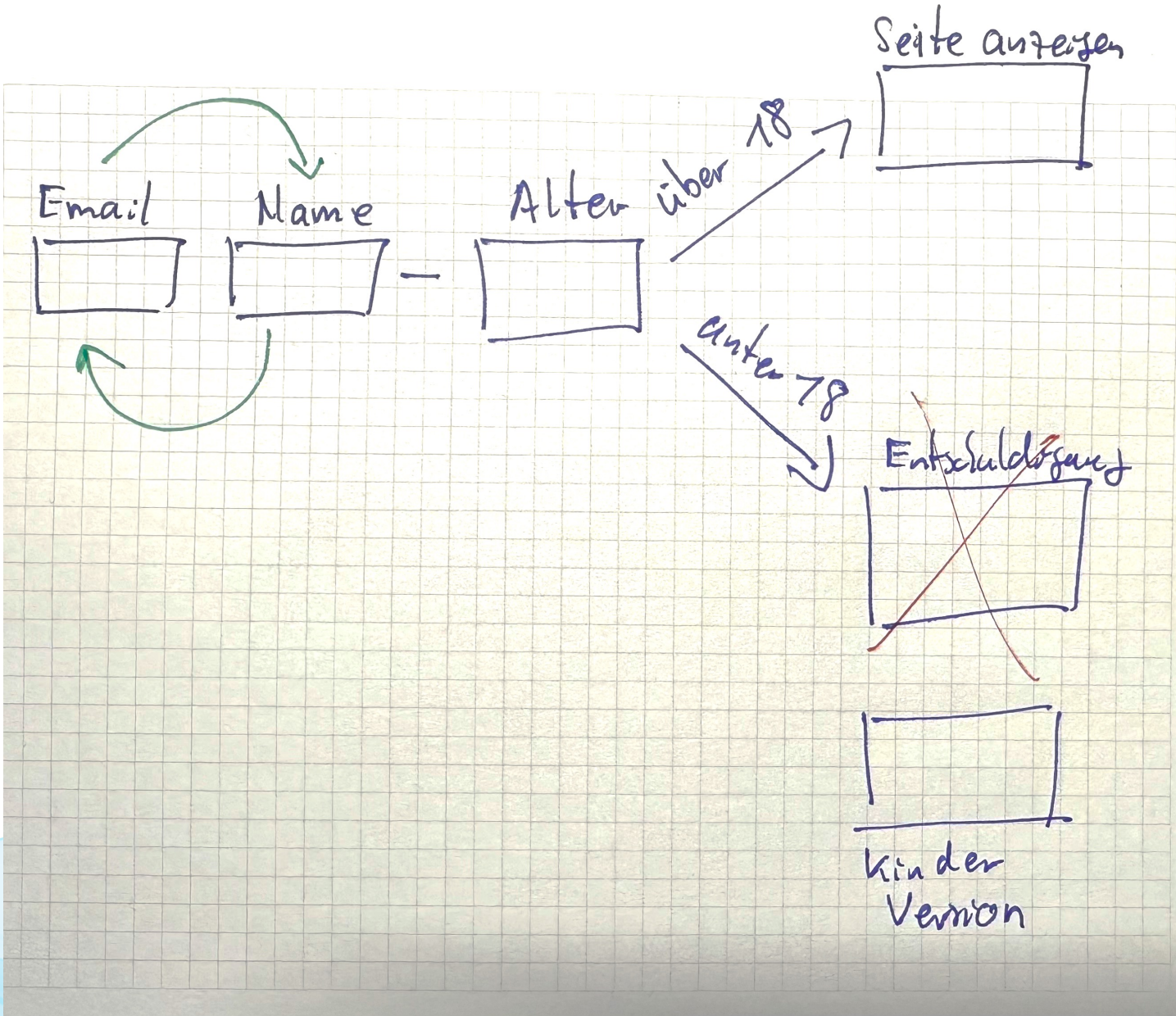
    def __str__(self):
        return self.summoner.summonerName + " - " + self.queueType + " - " + str(self)

class Meta:
    ordering = ['-timestamp']
```



Visuell





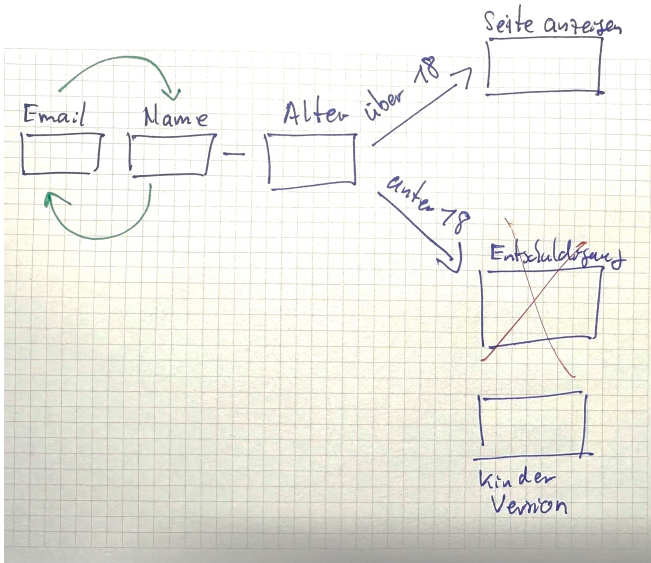
---

**Sorry, you must be over 18 to access this page.**

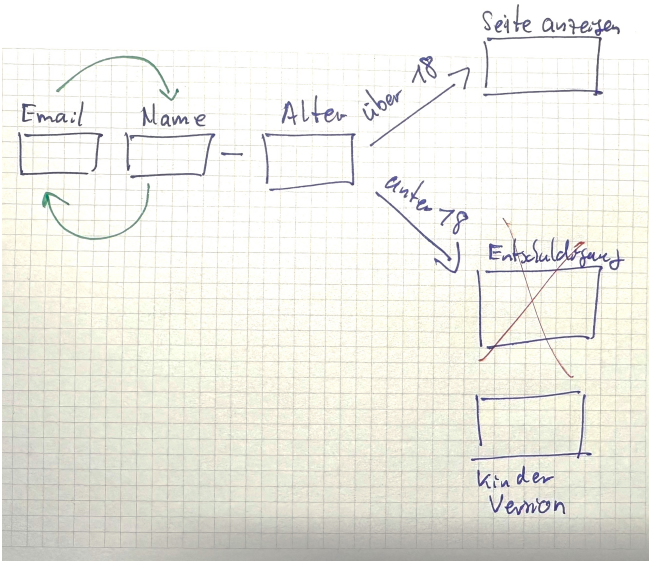
---

I am over 18 years old

**Welcome, you are over 18.**







### 3. Design the Onboarding Page:

- Use Tailwind CSS to style the input forms for email and name.
- Implement the logic to accept the email and name, and store them using a state management library or React's `useState`.
- Add a checkbox or a dropdown to confirm the user's age.
- Based on the age selection, route the user to either the over 18 page or the apology page.

### 4. Implementing Navigation Logic:

- Use Next.js's `useRouter` hook to navigate between pages.
- Handle form submission and redirect the user to the correct page based on their age.

### 5. Creating the Over 18 and Apology Pages:

- For users over 18, display the content you want them to see.
- For the apology page, display a message that they cannot proceed if they are under 18.
- Use Tailwind CSS to style these pages accordingly.

### 6. Adding the Children Page:

- Create a form to collect information about children if necessary.
- Style the page with Tailwind CSS.

### 7. Global State Management (Optional):

- If you need to maintain the user state across pages, consider using Context API or a state management library like Redux or Zustand.





**Warum jetzt  
handeln?**



## Exam Results

● ChatGPT 3.5

● ChatGPT 4.0

Percentile Rank

10<sup>th</sup>

20<sup>th</sup>

30<sup>th</sup>

40<sup>th</sup>

50<sup>th</sup>

60<sup>th</sup>

70<sup>th</sup>

80<sup>th</sup>

90<sup>th</sup>

Verbal  
GRE



The Graduate Record Exam (GRE) is a standardized test that is required for admission to many graduate schools across North America.

63<sup>rd</sup>

99<sup>th</sup>

Evidence-based  
Reading & Writing  
SAT



87<sup>th</sup>

93<sup>rd</sup>

Uniform Bar Exam  
Law



10<sup>th</sup>

90<sup>th</sup>

Math  
SAT



70<sup>th</sup>

89<sup>th</sup>

LSAT  
Law



40<sup>th</sup>

88<sup>th</sup>

Biology  
Advanced Placement



62<sup>nd</sup>

85<sup>th</sup>

Statistics  
Advanced Placement



40<sup>th</sup>

85<sup>th</sup>

Psychology  
Advanced Placement



83<sup>rd</sup>

# 21 Monate

