Testen

- Testrapport
- Stappen
- Hang Man - inleiding
- Hang man - use cases en scenario's
- Hang Man - bevindingen
- Hang Man - conclussies
- Tafels leren

Testrapport

Een testrapport bestaat uit

- **Inleiding**: Geef een korte beschrijving van de software die wordt getest, de versie van de software, het doel van het testen en andere relevante informatie, zoals de datum van de test.
- **Lijst van use cases**: Maak een lijst van de use cases die je hebt getest. De lijst van use cases heb je al vanuit je planning en ontwerp.
- **Testplan**: Geef een overzicht van je testplan. Per use case stel je scenario's op. En per scenario stel je alle stappen op.

> De volgorde (of eigenlijk hiërarchie) is dus: **use case - scenario - stappen**

- **Testresultaten**: Je voert elk scenario één of meer keer uit en beschrijft je bevindingen.
    1. **Fouten en problemen**: Documenteer alle fouten of problemen die je tegenkwam tijdens het testen. Dit kan van alles zijn, van bugs in de software tot problemen met de gebruikservaring.
- Als alles is getest beschrijf je een conclusie, daarin staat:
    1. **Bevindingen en conclusies**: Op basis van de resultaten van de tests, wat zijn je algemene bevindingen? Werkt de software zoals verwacht in alle geteste use cases? Zo niet, welke problemen heb je geïdentificeerd?
    2. **Aanbevelingen**: Geef aanbevelingen op basis van je testresultaten. Dit kunnen suggesties zijn voor het verbeteren van de software, voor aanvullende tests die moeten worden uitgevoerd, of voor het oplossen van geïdentificeerde problemen.
- **Bijlagen**: Voeg eventuele relevante bijlagen toe, zoals screenshots, logbestanden of gedetailleerde resultaten van individuele tests.

In de volgende lessen gaan we deze stappen oefenen.

Stappen

Voorbeeld stappen

Je bent een software tester voor de website van de Openbare Bibliotheek Amsterdam (https://www.oba.nl/). Je taak is om een testrapport te maken over de volgende use case:

Use Case: Als lid van de bibliotheek wil ik een Boek Zoeken en reserveren zodat ik het boek kan ophalen en kan lenen.

Scenario 1: Een gebruiker wil een specifiek boek vinden en hij weet alleen de titel van het boek.

Scenario 2: Een gebruiker wil een specifiek boek vinden en hij weet alleen het ISBN-nummer van het boek.

Scenario 3: Een gebruiker wil een specifiek boek vinden en hij weet alleen de naam van de schrijver van het boek.

Stappen voor scenario 1

1. De gebruiker gaat naar de homepage van de website.
2. De gebruiker klikt op het zoekveld in het midden van de pagina.
3. De gebruiker voert de titel van het boek in.
4. De website geeft een lijst met resultaten weer die overeenkomen met de zoekterm van de gebruiker.
5. De gebruiker klikt op de titel van het boek dat hij/zij wil reserveren.
6. De gebruiker wordt naar een pagina geleid met details over het geselecteerde boek.

Opdracht

Beschrijf het stappenplan voor scenario 1,2 en 3. Let op het stappenplan zoals het hier staat *mist een paar details*.

Inleveren

3 keer een volledig stappenplan voor scenario 1 2 en 3.

Tip

Let op je mag een stappenplan maken omdat de stappen bijna allemaal hetzelfde zijn. Op het punt waar het verschillend is, maak je dan een onderverdeling:

1. ....
2. ....
3. Hier zijn drie mogelijkheden:
    1. sub scenario 1
    2. sub scenario 2
    3. sub scenario 3
4. ....
5. ....

Hang Man - inleiding

Inleiding

Je gaat een Python spelletje installeren. Dit spel gaan we testen.

Spel

Je hebt een *hang man* spel, waarvan hier de code:

```
# versie 0.9 hangman

import random

def hangman():
    word_list = ["python", "hangman", "programming", "game", "computer"]
    word = random.choice(word_list).lower()
    guessed_letters = []
    attempts = 8

    print("Welkom bij Hangman!")
    print("Probeer het woord te raden door letters te kiezen.")
    print(f"Je hebt {attempts} pogingen voordat het spel voorbij is.")

    while True:
        hidden_word = ""
        for letter in word:
            if letter in guessed_letters:
                hidden_word += letter + " "
            else:
                hidden_word += "_ "

        print("\n" + hidden_word)

        if "_" not in hidden_word:
            print("Gefeliciteerd! Je hebt het word geraden.")
            break
```

```
        if attempts == 1:
            print(f"Helaas, je hebt het woord niet geraden. Het woord was {word}.")
            break

        guess = input("Kies een letter: ").lower()

        if len(guess) != 1 or not guess.isalpha():
            print("Invalid input. Voer één enkele letter in.")
            continue

        if guess in guessed_letters:
            print("Je hebt deze letter al geraden. Probeer het opnieuw.")
            continue

        guessed_letters.append(guess)

        if guess in word:
            print("Goed geraden!")
        else:
            attempts -= 1
            print(f"Fout! Je hebt nog {attempts} pogingen over.")

if __name__ == "__main__":
    hangman()
```

Kopieer de code en speel het spel een keer.

Opdracht 1

Maak een inleiding.

Geef een korte beschrijving van de software die wordt getest, de versie van de software, het doel van het testen en andere relevante informatie, zoals de datum van de test.

Inleveren

De inleiding van het testrapport

...

Hang man - use cases en scenario's

Use Cases

Je hebt eigenlijk maar een paar use cases:

1. Als speler wil het spel starten omdat ik het spel wil gaan spelen
2. Als speler wil een beurt spelen en een letter raden om het woord te raden
3. Als speler wil ik het spel stoppen omdat ik het niet wil afmaken
4. Als beheerder wil ik de woorden die een speler kan raden veranderen

Bedenk bij alle use cases de stappen die je wilt uitvoeren om het te testen.

Bij use case 2 moet je meerdere zaken testen, denk aan ongeldige invoer, of een letter invoeren die je al hebt geraden, of een hoofdletter invullen van een kleine letter die al is geraden, bedenk verschillende scenario's en beschrijf de stap of stappen.

Bij use case 2 eindigt het spel ook. Het eindigt of omdat je te veel beurten hebt gehad, of omdat je hebt gewonnen en het woord hebt geraden.

Opdracht

Maak een hoofdstukje van elke use case en beschrijf bij elke use case een scenario. Beschrijf de stap of stappen die je moet doorlopen voor het scenario.

Bij use case 2 beschrijf je minimaal 5 scenario's met de bijbehorende stappen.

Als alle stappen zijn beschreven dan voer je alle suit en beschrijf je per proces de bevindingen.

Inleveren

Werk alle scenario's uit en beschrijf deze. Verdeel de scenario's per use case.

--

Hang Man - bevindingen

In de vorige opgave heb je een aantal stappenplannen gemaakt.

Voer alle stappenplannen uit en beschrijf per stappen plan je bevindingen.

Je voert minimaal alle stappenplannen uit, maar mocht je tegen zaken aanlopen die misschien niet in de stappenplannen staan, dan is dat prima. Noteer ook deze bevindingen.

Hoe beschrijf je een bevinding?

Stel je hebt een scenario waarin je de letter A hebt geraden en dat de letter A is afgekeurd. Als je de A opnieuw raadt wil je niet dat dit als een beurt wordt gerekend.

Je bevinding zou er dan als volgt kunnen uitzien:

> *Nadat de letter A was ingevoerd en was afgekeurd werd de beurt geteld. Ik had één raad-poging minder over. Toen ik opnieuw de A invoerde kreeg ik een boodschap dat ik A al had geprobeerd en het aantal raad-pogingen bleef gelijk. Bij het invoeren van een a (kleine letter) nadat ik een A had geraden was het resultaat hetzelfde. De raadpogingen zijn dus case insensitive.*

Opdracht

Voer alle scenario's uit, en beschrijf je bevindingen.

Je kan de scenario's die je had opgesteld nummeren zodat je de bevindingen ook kan nummeren zodat je weet welke bevinding bij welke scenario hoort.

Nog overzichtelijker is om alles in een tabel te zetten!

Inleveren

Scenario's (die je al had) met daarbij de bevindingen.

Hang Man - conclussies

Als laatste maken we paragraaf conclusies.

In de conclusies benoem je het algemene beeld, zoals:

- Wat vind je van de kwaliteit van de software?
- Zijn er zaken die los van de use cases en scenario's aandacht nodig hebben?
- Wat zijn de belangrijkste bevindingen?
- Wat zijn je belangrijkste aanbevelingen of zijn die er niet?

Opdracht

Beschrijf de conclusies van je test.

Tafels leren

Je gaat opnieuw een Python programma testen.

Tafels leren

Met het Python-programma tafels leren .

Use cases

- Als een student wil ik mijn tafels oefenen binnen een bepaalde tijdslimiet, zodat ik mijn snelheid en nauwkeurigheid bij het vermenigvuldigen kan verbeteren."

  Het programma heeft een tijdslimiet. Bij elke nieuwe som wordt de tijdslimiet getoond.

- "Als een student wil ik in staat zijn om de moeilijkheidsgraad van mijn oefeningen te verhogen, zodat ik kan blijven uitdagen en verbeteren van mijn rekenvaardigheden."

  Na elke drie goede antwoorden wordt het niveau verhoogd en worden de vragen moeilijker.

- "Als een student wil ik via een game-achtig programma worden uitgedaagd, zodat ik ga proberen een betere score dan te halen en zodoende motivatie krijg om meer te oefenen. "

  Het 'spel' telt punten en geeft aan op welk niveau je bent uitgekomen. De punten kunnen worden vergeleken met anderen.

- "Als een student wil ik dat dezelfde vraag niet twee keer achter elkaar wordt gesteld, zodat ik een breder scala aan vragen kan ervaren en mijn vermogen om verschillende vermenigvuldigingsproblemen op te lossen kan verbeteren."

  Het programma zou niet twee maal dezelfde vraag achter elkaar moeten stellen.

Opstellen testplan

Maka een testplan zoals je dat hebt geoefend.

Maak de volgende hoofdstukken:

1. **Inleiding**,

2. **Use cases.**
3. **Scenario's** per use case.
4. S**tappenplan** per scenario.
5. Voer je stappenplan uit.
6. Beschrijf je **bevindingen** per scenario.
7. Beschrijf de algehele **conclussie**.

Inleveren

Volledig testplan in pdf.

Code

```
import time
import random

# De tijd wanneer het spel start
start_time = time.time()

# Tijdslimiet voor het spel in seconden
time_limit = 20

# Teller voor het aantal goede antwoorden
correct_count = 0

# Bepaal het initiële bereik voor de getallen
number_range = 5

# Bewaar de laatste vraag
last_question = None

while True:
    # Bereken hoeveel tijd er al verstreken is
    elapsed_time = time.time() - start_time

    # Als de tijdslimiet is bereikt, stop dan met het spel
    if elapsed_time > time_limit:
```

```
        print("Tijd is op!")
        print(f"Je hebt {correct_count} vragen goed beantwoord.")
        print(f"Je eindigde op moeilijkheidsgraad {number_range - 4}.")
        break

    # Bereken de resterende tijd en print deze
    remaining_time = time_limit - elapsed_time
    print(f"Resterende tijd: {remaining_time:.2f} seconden")

    # Verhoog het bereik elke keer dat de gebruiker vijf vragen correct beantwoordt
    if correct_count % 3 == 0 and correct_count > 0:
        number_range += 1
        print(f"Moeilijkheidsgraad verhoogd! Je zit nu op moeilijkheidsgraad {number_range - 4}.")

    # Genereer een nieuwe vraag
    num1, num2 = random.randint(1, number_range), random.randint(1, number_range)

    # Controleer of de nieuwe vraag hetzelfde is als de vorige vraag
    # Als dat het geval is, genereer dan een nieuwe vraag
    while (num1, num2) == last_question:
        num1, num2 = random.randint(1, number_range), random.randint(1, number_range)

    # Bewaar de huidige vraag als de laatste vraag
    last_question = (num1, num2)

    correct_answer = num1 * num2

    # Stel de vraag aan de gebruiker
    user_answer = int(input(f"Wat is {num1} * {num2}? "))

    # Controleer het antwoord
    if user_answer == correct_answer:
        print("Correct!")
        correct_count += 1
    else:
        print("Fout! Het juiste antwoord was: ", correct_answer)
```