Projectrapportage  
  
Stationszuil  
  
  
  
  
  
  
  
  
  
Afbeelding met Graphics, Elektrisch blauw, symbool, schermopname

Automatisch gegenereerde beschrijving  
  
  
  
  
  
  
  
  
  
Datum: 2 november 2023  
Versie: 1.0  
Auteur: Aimane Chakiri

Inhoudsopgave  
  
Inleiding  
Doelstelling  
Ontwerp en proces  
 Module 1  
 Module 2  
 Module 3  
Resultaat  
Reflectie  
Referenties  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Inleiding  
  
In deze rapportage wordt de doelstelling, het proces, het resultaat en referenties besproken van het project uit blok 1.  
In dit project wordt er gewerkt aan drie modules: Eén: Het maken van een zuil waarin mensen berichten kunnen achterlaten over de NS. Twee: Een moderatie systeem om de berichten te controleren en als laatste module drie, een GUI scherm waarop allemaal verschillende data op getoond wordt.  
Al deze onderdelen functioneren gezamenlijk en de relevante data wordt opgeslagen in een online database.  
  
Er zal uitgebreid, per module, behandeld worden hoe het proces is verlopen en wat de code per line doet, gepaard met referenties indien nodig.  
  
Aan het einde zal het resultaat uitgebreid besproken worden vanuit het perspectief van een gebruiker. Hoe kan een gebruiker interageren met de gemaakte programma’s? Wat voor input wordt er gevraagd van de gebruiker?  
  
Als laatste zullen er referenties te lezen zijn. Deze zullen in het rapportage staan bij de relevante teksten. Er worden cijfers gebruikt die refereren naar bepaalde bronnen binnen het referentie gedeelte van deze rapportage.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Doelstelling  
  
Het project heeft als hypothetische werkgever de Nederlandse Spoorwegen (NS).  
De NS wil een enquête uitvoeren onder haar reizigers, het lijkt de directie daarom een goed plan dat klanten hun opmerkingen, complimenten, meningen et cetera via een digitaal systeem kunnen invoeren.  
Dit digitale systeem, een computerzuil, wordt de stationszuil genoemd. De stationszuil zal aanwezig zijn op elk station.  
Ook wil de NS deze berichten tonen in stationshallen op schermen, daarom moet er ook een vorm van moderatie zijn. De vrees is dat reizigers de stationszuil als uitlaatklep kunnen gaan gebruiken.  
Via de moderatie kunnen berichten met respectloze uitingen, schuttingtaal en dergelijke eruit gefilterd worden.  
  
De opdracht is verdeeld in drie verschillende modules.  
Module 1 is het creëren van de zuil waarmee reizigers berichten kunnen achterlaten.   
De stationszuil moet gerealiseerd worden in Python. De berichten mogen maximaal 140 karakters zijn.   
Uiteindelijk willen wij de berichten, datum en tijd van het bericht, naam van de reiziger – als de reiziger geen naam invult wordt dit automatisch anoniem en de locatie vanwaar het bericht verstuurd is.  
Deze gegevens mogen opgeslagen worden in een CSV-document of een tekstdocument, maar moeten uiteindelijk staan in een PostgreSQL database. Dit is van belang voor de interactie met module 3.  
  
Module 2 is het programmeren van een moderatie systeem. Een moderator van de NS moet de berichten kunnen controleren, voordat deze op stationshallen op schermen te zien zijn.  
Alleen goedgekeurde berichten worden uiteindelijk getoond op het scherm.  
Naast het feit dat het moderatie systeem berichten goedkeurt of afkeurt moet er ook opgeslagen worden wanneer de berichten beoordeeld zijn, wie de moderator is die de beoordeling heeft gemaakt, de naam van de moderator en ook het email-adres van de moderator.  
Ook moet er een database gemaakt worden. Voordat de database gemaakt kan worden moeten er modellen gemaakt worden, zowel conceptueel, logisch en fysiek. Vervolgens moet de database in Postgresql gerealiseerd worden.  
  
Module 3 is het ontwikkelen van het stationshalscherm. Op dit scherm zijn de laatste 5 goedgekeurde berichten te zien, maar niet alleen de goedgekeurde berichten.   
Ook moeten de beschikbare faciliteiten op het station getoond worden en toegevoegd worden aan de database.  
De database moet ook online staan via een virtual machine in de Azure cloud.   
Ten slotte wordt er op het scherm ook het weerbericht getoond van de locatie waar het scherm zich bevindt. Hiervoor wordt gebruik gemaakt van de OpenWeatherMap API.  
Het scherm moet er goed uitzien, daarom wordt er in deze module gewerkt met een   
Graphical User Interace (GUI) met behulp van Tkinter.  
  
Ook is plannen van belang voor dit project en het proces. Er wordt in sprints gewerkt. Elke sprint wordt er een planning gemaakt. Hiervoor wordt het Teams-planbord gebruikt.   
  
Reflecteren is ook belangrijk, in de overgang van de ene sprint naar de andere sprint wordt er terug gekeken op het proces, wat ging goed of wat ging slecht?   
Stilstaan bij het proces helpt bij het proces en geeft je een moment om te leren en de kans jezelf te verbeteren.   
  
Op het einde wordt er afgesloten met een eindpresentatie aan de docenten. 1  
  
  
  
Ontwerp en proces  
  
Het ontwerp en proces wordt in dit hoofdstuk uiteindelijk per module behandeld, maar eerst wil ik graag een algemeen beeld schetsen voordat we het ontwerp en proces per module af gaan.  
  
Het ontwerp van de modules is per module gedaan in de vorm van de sprints. Ook heb ik met sprints gewerkt omdat er een gebrek aan kennis was ten opzichte van Python, Azure en PostgreSQL.  
Het werken in sprints en het focussen op één module op volgorde van één naar drie is de manier van werken geweest.   
Per module is er gewerkt aan een ontwerp van wat er nodig was en heb ik een simpele planning gemaakt via Teams.   
Module twee moest dus bijvoorbeeld wachten tot dat ik de relevante kennis vergaard had via de lessen en module één moest bijna afgerond zijn voordat ik te werk ging aan module twee.  
  
Dit is hoe ik algemeen te werk ging en vandaar wil ik graag dit hoofdstuk per module bespreken. Ook omdat er tussen de modules veranderingen zijn in het ontwerp en proces. Zo hebben we bijvoorbeeld tijdens één van de project lessen geleerd met Teams te werken. Ik was toen al bijna zo goed als klaar met module één en ben Teams pas gaan gebruiken in module twee.  
  
  
Module één  
  
Module één was al tamelijk snel afgerond na de eerste twee weken. Er is in zowel het ontwerp als proces te zien dat er geen duidelijke structuur te zien is. Zo kun je als je de code van module één vergelijkt met twee en drie zien dat ik steeds meer gestructureerd te werk ben gegaan.  
  
Zo heb ik in module één bijvoorbeeld geen gebruik gemaakt van opmerkingen door ‘#’ te gebruiken.   
Deze heb ik achteraf toegevoegd om terug te kunnen lezen wat bepaalde lines aan code doen.  
Ik ben de hashtags uiteindelijk wel in module twee en drie gaan gebruiken om pseudocode te schrijven. De pseudocode maakt het namelijk overzichtelijk waar ik naartoe wil werken en wat daarvoor nodig is!  
  
Module één was tamelijk simpel en was goed te doen zonder een duidelijk proces of voorkennis, maar wat voor proces ik uit module één heb verkregen is een waardevol leerproces. Module één heeft mij geleerd hoe ik meer gestructureerd te werk kan gaan. Ook heeft het mij geholpen met het beter begrijpen van Python.  
  
Zo heb ik zoals eerder genoemd geleerd pseudocode te gebruiken, ook heb ik geleerd mijn code wat overzichtelijker te schrijven.   
Werken aan module één heeft ook goed geholpen met het ontwikkelen van troubleshooting skills.  
Zo ben ik meer bekwamen geworden met het oplossen van foutcodes waar ik minder goed bekend mee ben en ik ben ook beter geworden in het opzoeken van bijvoorbeeld functies of modules die ik niet ken maar wel nodig heb om uit te voeren wat ik wil.  
Een voorbeeld is het gebruiken van de datetime module, dit hebben we niet behandeld in de lessen tot later in dit blok. Ik was al vol op bezig met module één en wist niet hoe ik de datum en tijd van achtergelaten berichten door reizigers kon opslaan en verwerken. Door het googlen naar wat ik nodig had ben ik op bronnen zoals stackoverflow, W3Schools of Python Docs terecht gekomen.  
  
  
  
  
  
  
  
Het ontwerp van de geschreven code in module één ziet er uit als volgt:  
Afbeelding met tekst, schermopname, software

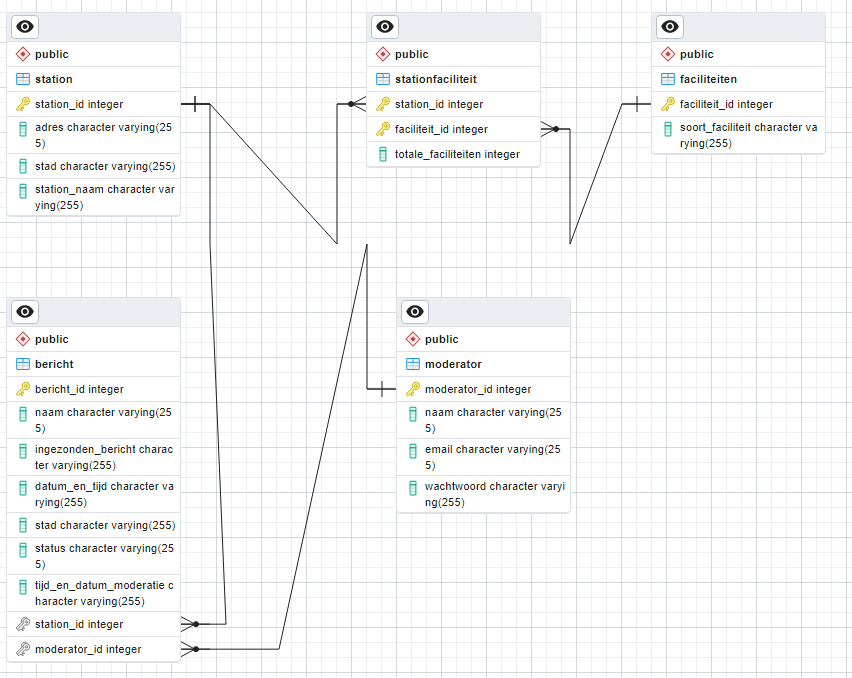
Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving  
  
  
  
  
Zoals eerder genoemd in deze rapportage is er niet gekozen voor een bepaald ontwerp en ben ik maar code gaan schrijven, zonder hier een logische volgorde voor te bedenken.  
  
De gehele code is geschreven in een while loop, zodat het programma constant aan staat en niet na elk ingevoerde bericht handmatig opnieuw opgestart moet worden.  
Er wordt gevraagd om een naam en een bericht, indien er geen naam wordt ingevuld wordt er automatisch ‘Anoniem’ ingevoerd.  
Vervolgens vraagt het programma om een bericht van de gebruiker. Dit bericht wordt gecontroleerd op het aantal karakters door middel van nog een while loop. 2  
Daarna kiest het programma een random stad uit een lijst met steden die ik heb gekozen voor het project. 3  
Een dictonary is aangemaakt om de station IDs mee op te sturen. Deze zijn later in het project toegevoegd nadat er een database met station IDs aangemaakt was. Ook wordt er een tijd variabel aangemaakt om de datum en tijd van het opgestuurde bericht op te slaan. 4 en 5  
Vervolgens wordt alle relevante informatie dat opgeslagen staat in variabelen opgeschreven in zowel in een txt als CSV-document, ook wordt er een string opgeslagen in de documenten, namelijk ‘Niet gecontroleerd’. 6, 7 en 8  
Het txt document was een tijdelijke placeholder en ik heb besloten het te behouden om zo een groot overzicht te hebben van wat ik allemaal ingevoerd heb tijdens het project.  
Vervolgens sluit het programma af met een print statement. ‘Bedankt voor uw bericht! Wij wensen u nog een hele fijne dag!’  
  
Dit is waar het stuk over module één stopt. Er is een programma gecreëerd waarmee gebruikers berichten kunnen achterlaten, deze berichten en bijbehorende extra informatie wordt vervolgens geüpload naar een CSV-document.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Module twee  
  
Voor module twee is er een duidelijker proces. Voor module twee is er gebruik gemaakt van het Microsoft Teams planbord. Tijdens het schrijven van deze documentatie zag het planbord er uit als volgt:  
Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving  
  
Dit waren de drie taken die gedaan moesten worden om module twee afgerond te krijgen. Ik gebruikte het planbord op deze manier om een duidelijk algemeen beeld te hebben waar ik naartoe moest werken. Ik zou het planbord openen en vervolgens beslissen waar ik aan ging werken die dag. Ging ik aan de database werken of het moderatie systeem? Zoals je kunt zien had ik de database eerder klaar dan de moderatie en daarna het CSV document via Python gelinkt aan de database. Ik had deze drie taken ook op die volgorde aangemaakt in mijn Microsoft Teams. Het leek mij het meest logisch om namelijk eerst de database te maken, zonder een database zou het moderatie systeem namelijk nutteloos zijn. Vervolgens heb ik het moderatie systeem gemaakt en in dat systeem wordt het CSV document gelinkt aan de database.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Eerst bespreken we het ontwerp van de database, die ziet er uit als volgt:  
Afbeelding met tekst, Post-it-briefje, schermopname, diagram

Automatisch gegenereerde beschrijving  
Conceptueel datamodel  
Afbeelding met tekst, schermopname, diagram, ontwerp

Automatisch gegenereerde beschrijving  
Logisch datamodel  
  
Afbeelding met tekst, schermopname, diagram, Rechthoek

Automatisch gegenereerde beschrijving  
Fysiek datamodel  
  
Eerst is er gekeken naar de opdracht en is er informatie vergaard, deze informatie is vervolgens via Visual Paradigm in een model verwerkt zoals in de foto’s hier boven te zien is.  
Vervolgens is er in PostgreSQL, via de SQL syntax, een database gemaakt die overeenkomt met het model.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Hier volgt een foto van de ERD volgens PostgreSQL, na het maken van de database in PostgreSQL:  
  
  
Hier uit kunnen we concluderen dat de database overeenkomt met het model. Later in het hoofdstuk resultaat bespreken we de werking van de database.  
Nadat ik de database gemaakt had was ik begonnen aan het moderatie systeem.  
Tijdens het maken van het moderatie systeem merkte ik op dat het database verbonden moest zijn met Python om mijn moderatie systeem te programmeren zoals ik het wil hebben.  
Daarom heb ik na het coderen van een paar lines van het moderatie systeem de database online gezet.  
Hiervoor is een virtual machine van Microsoft Azure gebruikt. Op de virtuele machine draait een Ubunut versie van Linux.  
Deze moest ik updaten, vervolgens moest is PostgreSQL downloaden op de Linux OS. Na het downloaden van PostgreSQL moesten er aanpassingen gemaakt worden in de files van PostgreSQL om connectie te maken met pgAdmin4. 9  
Nadat dit allemaal gedaan was heb ik een nieuwe server aangemaakt. Deze server was nog leeg en de database die ik gemaakt had stond op een localhost.  
Ik heb van de database een backup gemaakt en vervolgens de database op de online server gezet.  
Vervolgens heb ik de nodige informatie in de database met insert commands toegevoegd.  
Denk aan informatie zoals wie de moderators zijn, welke stations er allemaal zijn, wat voor faciliteiten er beschikbaar zijn.  
  
  
  
  
  
  
  
Nadat de database gemaakt was is er gewerkt aan een moderatie systeem.  
Het moderatie systeem ziet er uit als volgt:  
  
Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving  
  
  
Indien nodig is importeer ik boven aan het programma modules zoals psycopg2, hashlib, datetime etcetera. Vervolgens maken we een variabel aan ‘time’ om de datum en tijd van het modereren op te slaan.  
Daarna wordt er een connectie gemaakt met de online database via module psycopg2. Vervolgens wordt er gevraagd naar het email adres en wachtwoord van de moderator. Deze wordt met een query gecontroleerd in de database. Als het email adres en wachtwoord overeen komen in de database krijg je toegang tot het moderatie systeem.  
Het wachtwoord staat gehashed in de database en het wachtwoord dat gevraagd wordt als een input wordt in Python gehashed om beide hashes te vergelijken met elkaar. 10  
Als er toegang verleent wordt aan de moderator is er een menu. Dit menu wordt uitgeprint en vervolgens wordt er naar een keuze gevraagd.  
Keuze één was tamelijk makkelijk te programeren. Door terug te kijken naar de stationszuil was het makkelijk uit te vogelen hoe het CSV-document gelezen kon worden en uitgeprint kon worden.  
  
  
  
Keuze twee is voor het modereren van de berichten in het CSV-document.   
In een for loop worden de berichten berichten van het CSV-document getoond, vervolgens kan de moderator via een input de status ‘Afgekeurd of Goedgekeurd’ toe dienen. Indien dit verkeerd wordt opgeschreven krijgt de moderator een foutmelding en blijft de status ‘Niet gecontroleerd’.  
Indien er een beoordeling is gegeven wordt in de for loop de beoordeelde rij uit het CSV-document aangepast. De status wordt aangepast naar Goedgekeurd of Afgekeurd, er wordt een tijd toegevoegd om aan te tonen wanneer de beoordeling gemaakt is en de moderator ID wordt ingevoerd zodat er terug te zien is welke moderator de beordeeling heeft geplaatst.  
Vervolgens worden alle rijen opnieuw in het CSV-document ingevoerd. Zowel de ‘Goedgekeurde, Afgekeurde’ en ‘Niet gecontroleerde’ berichten. 11  
  
Keuze drie is het opsturen van goedgekeurde berichten naar de PostgreSQL database die online draait op de Azure cloiud.  
Eerst wordt het CSV-document weer geopend en gelezen. Vervolgens lezen we elke rij via een for loop en worden de rijen waar ‘Goedgekeurd’ op de index van status staat opgestuurd naar de database met een query. Dit wordt uitgevoerd via een if statement. Vervolgens committen we de query. Daarna openen we het CSV-document weer opnieuw, maar dit keer in write mode. We maken een lege lijst aan waar we alle berichten zonder status ‘Goedgekeurd’ in kwijt willen. Dat doen we weer met een for loop, rij voor rij leest de for loop het CSV-document en wanneer de juiste index niet gelijk staat aan ‘Goedgekeurd’ wordt die toegevoegd aan de lege lijst.  
  
De lijst wordt dan vervolgens herschreven in het CSV-document. Dit doen we zodat de goedgekeurde berichten die naar de database gestuurd zijn niet opnieuw opgestuurd kunnen worden. We willen de goedgekeurde berichten natuurlijk maar één keer opsturen, niet meerdere keren. Dus nadat de goedgekeurde berichten in de database staan worden ze uit het CSV-document verwijderd, omdat we ze niet meer nodig hebben.  
  
Keuze vier doet eigenlijk het zelfde als keuze drie, behalve het opsturen van berichten naar keuze drie. In keuze vier gebruiken we een for loop om alle rijen in het CSV-document te lezen. Vervolgens checken we de beoordeling status van de rijen. Indien deze niet gelijk staan aan ‘Afgekeurd’ worden ze in een lege lijst geappend. Vervolgens herschrijven we de lege lijst in het CSV document.  
Op deze manier verwijderen we alle afgekeurde berichten uit het CSV-document.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Module drie  
  
De planning van module drie ging als volgt:  
Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving  
  
Deze screenshot is genomen nadat ik klaar was met het maken van de documentatie en presentatie.  
De database was al gelinkt aan de Azure cloud, nu moest ik alleen nog een GUI maken. Na het maken van de GUI kan er gewerkt worden aan deze rapportage.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Het ontwerp van het GUI programma ziet er uit als volgt:  
Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname, software

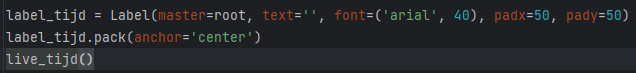
Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname

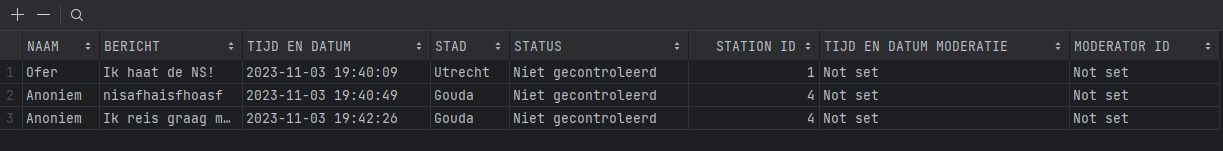
Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname, software, scherm

Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving  
  
Ik heb achteraf nog een live tijd toegevoegd aan het scherm, geïnspireerd door Barrelt. 16  
Afbeelding met tekst, Lettertype, schermopname, nummer

Automatisch gegenereerde beschrijving  
  
  
  
Het ontwerp werkt als volgt.   
Boven aan worden de relevante modules geïmporteerd. Vervolgens wordt er een connectie gemaakt met de PostgreSQL database via psycopg2. Daarna heb ik eerst de GUI vorm gegeven door een root aan te maken en allemaal labels te maken voor het vormgeven van de GUI. Het vormgeven van de GUI is gedaan met het idee dat alle labels gevarieerde teksten tonen, gebaseerd op verschillende data. Zo is het label voor het welkomst bericht en de aangegeven faciliteiten anders gebaseerd op welk station het scherm staat.  
  
Het enige label dat niet varieert is het NS logo, die staat vast. 12  
  
Daarna heb ik allemaal functies geschreven om de labels aan te passen. Deze labels maken gebruik van queries, dictonaries, inputs et cetera.  
  
Eerst vraagt het Python console ‘In welke stad staat het scherm?’ Na het antwoorden van deze vraag wordt het opgeslagen in de variabel ‘gekozen\_stad’.  
Er wordt vervolgens ook gekeken naar het ID nummer van de stad, dit wordt opgeslagen in de variabel ‘station\_id’. De faciliteiten, uitgedrukt in het ID nummer van de faciliteiten, aanwezig op het station van de stad worden opgeslagen in de lijst ‘faciliteiten\_lijst’. Hierna begint alle code die de GUI vormgeeft.  
  
De functie ‘stad\_aanvragen()’ vraagt de steden op uit de database. Vervolgens wordt er gecontroleerd of ‘gekozen\_stad’ in de database staat. Als dit het geval is print het console dat er een stad gekozen wordt ter bevestiging. Vervolgens wordt het welkomst label aangepast gebaseerd op het stations naam van de gekozen stad. Ook wordt er geluid afgespeeld op de achtergrond ter immersie. 13, 14 en 15  
Als de gekozen stad niet in de database staat wordt er opnieuw gevraagd naar een stad.  
  
Er staat een dictionary die nodig is om te bepalen welke faciliteiten op het station staan.  
  
De functie ‘faciliteiten\_label\_aanpassen()’ begint met een string. Deze string zal toegevoegd worden aan het label voor het tonen van de faciliteiten beschikbaar op het station. Voordat de string toegevoegd wordt aan het label moeten de beschikbare faciliteiten aan de string toegevoegd worden.  
Dit doen we via een for loop die loopt in ‘faciliteiten\_lijst’. Indien de ID nummers in de dictonary staan worden de values van de dictonary toegevoegd aan de string.  
Vervolgens wordt er een hele nieuwe string gemaakt die toegewezen wordt aan het label voor het tonen van de faciliteiten.  
  
Als een-na-laatste functie, ‘label\_weernbericht\_aanpassen()’, veranderen we het label dat het weerbericht toont.   
Dit doen we door een API te gebruiken van openweathermap.org. Via json slaan we de gegevens op in de variabel ‘data’. Vervolgens slaan we specifieke informatie van openweathermap.org op in variabelen. Nadat dit allemaal opgeslagen staat in variabelen maken we via een f string daar een gehele string van die toegewezen wordt aan het label dat het weerbericht toont op de GUI.  
  
Als laatste functie, ‘berichten\_displayen()’, voeren we een SQL query uit. Deze query vraagt naar de laatste vijf opgestuurde berichten die in de database staan.  
Vervolgens slaan we de namen, berichten en steden op in aparte lijsten. Op die manier kunnen we f strings maken zoals ‘f'{namen[0]} heeft vanuit {steden[0]} verstuurd: "{berichten[0]}"'.  
Daarna worden de vijf verschillende f strings toegewezen aan labels die worden gebruikt om de berichten te tonen op het ontworpen GUI.  
  
Al deze functies worden tussen de labels door uitgevoerd. Vervolgens is de laatste code een functie die van de GUI een loop maakt.  
In de screenshots ontbreken er nog wat comments hier en daar, die zijn te vinden in de meest recente versie van de GUI op github.  
  
  
  
  
  
  
  
  
  
Resultaat  
  
Het resultaat uit dit project is een stationszuil waarmee berichten opgestuurd kunnen worden naar een CSV document dat vervolgens door een moderator gecontroleerd wordt. De gecontroleerde berichten worden vervolgens verwijderd of naar de online database gestuurd. Vervolgens kan de GUI via de database informatie vergaren en dit tonen op een stationshalscherm.  
  
De stationszuil wordt uitgevoerd in het Python console. (dit kan mogelijk veranderd zijn na het schrijven van deze rapportage als ik tijd heb om extra werk te verrichten.)  
Om het resultaat te tonen maak ik gebruik van een paar foto’s om de werking van de code te tonen.  
Ik zal laten zien dat het invoeren van een naam en bericht opgeslagen wordt in een CSV-document, dat geen naam opgeven automatisch anoniem invoert en dat er controle is op het aantal karakters.  
  
Afbeelding met schermopname, tekst

Automatisch gegenereerde beschrijving  
  
  
Zoals te zien is worden de antwoorden van de stationszuil opgeslagen in het CSV-document en zijn de antwoorden die te lang of nul tekens bevatten. Ook is te zien dat als er geen naam wordt ingediend er ‘Anoniem’ wordt ingevuld.  
  
Dan gaan we nu kijken naar het moderatie systeem. Deze start door te vragen naar een email adres en wachtwoord. Als deze niet kloppen krijg je de zelfde vraag weer te zien. Komt het gene dat ingevuld is overeen met wat er in de database staat dan krijg je toegang tot het moderatie systeem.  
Door een cijfer, één t/m vijf, in te vullen worden er acties uitgevoerd.  
Ik ga deze acties nu één voor één uitvoeren en het resultaat tonen met behulp van afbeeldingen.  
Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving  
  
  
  
  
Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving  
Afbeelding met schermopname

Automatisch gegenereerde beschrijving  
Afbeelding met tekst, schermopname, nummer, Lettertype

Automatisch gegenereerde beschrijving  
  
  
  
  
  
  
  
  
  
  
Zoals te zien is in de afbeeldingen werken alle acties die een moderator kan uitvoeren correct.  
Actie nummer één toont alle berichten die in het CSV-document staan.  
Actie nummer twee maakt het mogelijk voor de moderator om een beoordeling te plaatsen, bij een foutieve beoordeling zoals ik voor heb gedaan blijft de status ongewijzigd. Bij het uitvoeren van actie nummer twee zien we dat alleen de verkeerd beoordeelde bericht nog beoordeeld kan worden.  
Actie nummer drie verstuurt goedgekeurde berichten naar de database toe en verwijdert deze vervolgens uit het CSV-document. Dit is terug te zien in de laatste twee afbeeldingen.  
Actie nummer vier verwijderd alle afgewezen berichten, dit is ook terug te zien in de een-na-laatste afbeelding.  
Vervolgens stopt actie nummer vijf het programma.  
  
Als laatste bekijken we het resultaat van de GUI.  
In het Python console wordt ons gevraagd om een stad naam in te voeren.  
Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijving  
  
Indien je iets invult dat niet in de database staat krijg je een nieuwe kans om een stad in te voeren.  
De input is hoofdletter gevoelig zoals je kunt zien op de afbeelding.  
  
De GUI die er uit komt ziet er uit als volgt.  
Afbeelding met tekst, schermopname, Lettertype, ontwerp

Automatisch gegenereerde beschrijving  
  
Er is een titel, ‘Stationscherm’, het logo is te zien en reizigers worden welkom geheten op het station.  
De bijhorende faciliteiten staan aangegeven, het weer in Utrecht wordt ook getoond en zoals te vergelijken is de afbeelding van de database worden de laatste 5 berichten getoond op de GUI.  
  
Het gehele resultaat is te zien op mijn github repository.  
De link naar mijn github is: <https://github.com/Aimane0>  
De link naar mijn privé github repository is: <https://github.com/Aimane0/StationsZuil>  
Als het klopt heeft Roos toegang tot de repository.  
  
  
  
Reflectie  
  
Ik heb geleerd dat ik geduldig moet werken. Vaak wil ik snel dingen af hebben of te snel aan iets beginnen, voordat ik überhaupt een duidelijk idee heb wat ik wil. Deze ongeduldigheid wordt ook gevoed door stress. Ik wil graag al te perfect werken en moet accepteren dat het maken van fouten okay is. Van fouten kun je leren en dat heb ik ook steeds meer gedaan des te comfortabeler ik mij begon te voelen met het project.   
Mijn ongeduld weerhield mij er van te leren van foutcodes en dergelijke.  
Elke sprint begon ik steeds meer geduld te tonen en mijn werk duidelijker te structureren.  
Zo zou je kunnen zeggen dat het hele project lang ik elke sprint mij hier over ontfermt heb.  
Ook moet ik het geduld en de tijd nemen om de doelstellingen en verwachtingen van het project goed door te nemen.  
Zo zou ik een volgend project een lijstje moeten maken van alle verwachtingen en doelen. Dit scheelt tijd ten opzichte van elke keer opnieuw op canvas het hele project doornemen.  
Meer structuur in hoe ik door het project wil maneuvreren en meer structuur in hoe mijn werk er uit moet zien. Zo is het gebruiken van pseudocode heel fijn voor het overzicht.  
Meer gedetailleerd plannen op Microsoft Teams zou ook kunnen helpen bij het verhelpen van de chaos en ongeduldigheid die ik heb ervaren.  
  
Qua het fysieke werk dat ik heb verricht voor het project voel ik mij significant comfortabeler met het programmeren in Python. Ik heb nu een beter idee hoe ik te werk moet gaan als ik een probleem of opdracht heb.  
Het denk proces tijdens het ontwerpen van mijn programma’s is steeds beter geworden per module, ook is mijn oplossend denkvermogen binnen Python verbeterd.   
Het vormen van een programma en het fixen van fout meldingen is een vaardigheid waar ik per module steeds beter in ben geworden.  
  
Ook ben ik de logica achter het gebruiken van de SQL taal en het maken van een database in PostgreSQL beter gaan begrijpen. Ik vond het analyseren van de opdracht best lastig, maar na de opdracht meerdere keren opnieuw gelezen te hebben heb ik de relevante informatie voor de database uit de opdracht beschrijving weten te verkrijgen.  
  
Als laatste ben ik de logica achter Linux beter gaan begrijpen. Zo was het lastig te wennen aan het feit dat er geen desktop was, maar uiteindelijk was het gebruiken van command lines in het terminal tamelijk makkelijk. Zo klikte het eindelijk dat het terminal eigenlijk gewoon een soort desktop is.  
  
Ik ben heel blij dat het leerproces zeer goed verlopen is, ik heb oprecht belachelijk veel opgepikt van syntax tot logisch na denken, probleem oplossend te werk gaan en analyseren.  
  
Wat zou ik het volgende blok anders willen doen?   
Ik wil geduldiger te werk gaan, het opzoeken van syntaxerrors en het fixen van foutcodes was essentieel in het leren van coderen en programmeren. Soms gaf ik te snel op en gooide ik al mijn werk weg om helemaal opnieuw te beginnen.   
Ook wil ik beter en effectiever gaan plannen door bijvoorbeeld duidelijkere en specifiekere taken voor me zelf te maken en daar ook deadlines aan toe te voegen.  
Ook lijkt het mij verstandig mijn referenties bij te houden vanaf het begin van het blok.  
Ten laatste wil ik mij zelf het volgende blok nog meer uitdagen. Ik heb hier en daar wat extra dingen toegevoegd aan het project, zoals een inlog systeem en een live tijd voor het scherm. Als ik mij hou aan de eerder benoemde punten, beter plannen, meer structuur in mijn werk vormen, dan zou ik meer tijd kunnen hebben om extra werk te verrichten.  
  
   
  
Referenties  
  
1: PROJECT: Casus en Eisen. Hogeschool Utrecht, van <https://canvas.hu.nl/courses/39942/pages/project-casus-en-eisen>  
  
2: Python While Loops. Geraadpleegd 18 September 2023, van <https://www.w3schools.com/python/python_while_loops.asp>   
  
3: Python random choice() function to select a random item from a List and Set. Geraadpleegd 18 September 2023, van <https://pynative.com/python-random-choice/>  
  
4: Python Get Current time. Geraadpleegd 18 September 2023, van <https://www.programiz.com/python-programming/datetime/current-time>  
  
5: Stack overflow. Geraadpleegd 18 September 2023, van <https://stackoverflow.com/questions/7999935/python-datetime-to-string-without-microsecond-component>  
  
6: Python File Write. Geraadpleegd 18 September 2023, van <https://www.w3schools.com/python/python_file_write.asp>  
  
7: csv — CSV File Reading and Writing. Geraadpleegd 18 September 2023, van <https://docs.python.org/3/library/csv.html#module-csv>  
  
8: Stack overflow. Geraadpleegd 18 September 2023, van <https://stackoverflow.com/questions/2918362/writing-string-to-a-file-on-a-new-line-every-time>  
  
9: Installatie en configuratie Postgresql op virtuele machine.docx. Hogeschool Utrecht, van <https://canvas.hu.nl/courses/39942/assignments/261554?wrap=1>  
  
10: Stack overflow. Geraadpleegd 24 Oktober 2023, van <https://stackoverflow.com/questions/5297448/how-to-get-md5-sum-of-a-string-using-python>  
  
11: Bron: ChatGPT. Doel: ChatGPT vragen waarom de index van de rijen in het CSV-document niet door Python geregistreerd werd. Bijdragen: Na het raadplegen van ChatGPT kwam ik er achter dat de lege kolommen niet geregistreerd werden. Ik heb in het programma van de stationszuil in kolommen zoals moderator ID besloten dat er een standaard waarde wordt ingevoerd, namelijk ‘Not set’  
Prompt: ‘Ik krijg als error:  
Traceback (most recent call last):  
 File "C:\Users\Aimane\PycharmProjects\PROJECT\_BLOK1\MODERATIE.py", line 82, in <module>  
 rijen[6] = str(tijd)  
 ~~~~~^^^  
IndexError: list assignment index out of range’  
  
ChatGPT adviseerde mij vervolgens te printen wat de len was van één rij in het CSV-document. Daar kwam acht uit, maar alsnog wilde het Python programma index zes bijvoorbeeld niet registreren.  
  
Vervolgens stelde ik de volgende vraag:  
‘Zou het kunnen liggen aan het feit dat de kolommen die Python niet leest <unset> zijn?’  
  
Daarop heeft ChatGPT geantwoord dat dat mogelijk is waarom de index volgens Python niet bestaat.  
  
12: Stack overflow. Geraadpleegd 4 November 2023, van <https://stackoverflow.com/questions/3177969/how-to-resize-an-image-using-tkinter>  
  
13: Stack overflow. Geraadpleegd 2 November 2023, van <https://stackoverflow.com/questions/62833561/playing-sound-in-background-in-python-while-gui-is-running-with-tkinter>  
  
14: Stack overflow. Geraadpleegd 2 November 2023, van https://stackoverflow.com/questions/42393916/how-can-i-play-multiple-sounds-at-the-same-time-in-pygame  
  
15: Bron: ChatGPT. Doel: ChatGPT vragen hoe ik twee verschillende mp3 bestanden kon afspelen via de mixer, want ik wist hoe ik er één moest afspelen, maar hoe ik twee moest afspelen lukte niet, ook al had ik een stack overflow gevonden (14) waarvan ik dacht dat die zou werken.  
Bijdragen: Na het raadplegen van ChatGPT heeft de AI mij uitgelegd hoe ik te werk moet gaan met channels.  
  
Prompt:  
***Ik:*** ‘Hoi! Het lukt mij helaas niet om twee verschillende MP3 bestanden af te spelen, zou je mij kunnen helpen? Dit is de code die ik op dit moment heb.   
‘’ mixer.init()   
pygame.mixer.set\_num\_channels(2)   
pygame.mixer.Channel(0).play(pygame.mixer.Sound('Omroepen.mp3')) pygame.mixer.Channel(1).play(pygame.mixer.Sound('Lopen.mp3'))   
mixer.music.play(-1)’’  
Wat gaat er hier mis?’  
  
ChatGPT stelde mij toen voor om boven ‘mixer.music.play(-1)’ ‘mixer.music.load('achtergrondmuziek.mp3')’ toe te voegen.   
  
Toen merkte ik aan ChatGPT op dat ik geen derde MP3 bestand wilde afspelen, dat ik die niet had en dat het ging om de twee MP3 bestanden in de code die ik had gedeeld.  
  
Toen legde ChatGPT mij uit dat ik ‘mixer.music.play(-1)’ moest verwijderen. Dat de twee geluidsbestanden dan wel zouden afspelen.  
  
16:plus2net.com. Geraadpleegd 6 November 2023, van <https://www.plus2net.com/python/tkinter-clock.php>  
  
17: NS. <https://www.ns.nl/platform/fundamentals/colours.html>