Testplan

De testcases in het testplan sluiten aan op de user stories en bevatten alle scenario’s

Voor dit onderdeel heb ik gebruik gemaakt van de user stories van de opdracht die ik op mijn stage heb gemaakt. Tijdens mijn stage heb ik gewerkt met het platform Betty Blocks. In het kort: Betty Blocks is een no-code platform, door middel van drag en drop van componenten kan je een applicatie bouwen. Ik heb als opdracht gekregen om een Case Management applicatie te bouwen. Na het bouwen van de applicatie heb ik gebruik gemaakt van End-to-End testen.

Alle functionaliteiten van de userstory: **Als een gebruiker, wil ik een nieuwe caserecord kunnen aanmaken** heb ik getest. Dus dit zijn ook de punten die ik zal toelichten. Bij elke pagina laat ik het gewenste resultaat, het scenario, het werkelijke resultaat en eventueel de punten die moeten worden verbeterd als het werkelijke resultaat niet klopt.

User stories:

* Als een gebruiker, wil ik een nieuwe caserecord kunnen aanmaken.
* Als een gebruiker, wil ik de waardes van een caserecord kunnen updaten.
* Als een gebruiker, wil ik een case type record kunnen verwijderen.
* Als een gebruiker, wil ik een nieuwe case type record kunnen aanmaken.
* Als een gebruiker, wil ik de waardes van een case type record kunnen updaten.
* Als een gebruiker, wil ik een case type record kunnen verwijderen.
* Als een gebruiker, wil ik een nieuwe case step record kunnen aanmaken.
* Als een gebruiker, wil ik de waardes van een case step record kunnen updaten.
* Als een gebruiker, wil ik een case step record kunnen verwijderen.

Voor het testen van mijn applicatie maak ik gebruik van End-to-End testen. End-to-end testen ook wel E2E testen is een test methode dat de gehele applicatie flow doorgaat. Van het begin tot het eind. Met deze test stel je veilig dat alle componenten werken, en de softwareapplicatie functionaliteiten werken in realistische scenario’s. Het is net als unit en functioneel testen in een. Bij E2E testen gaat het dan verder dan alleen op individuele units testen.

Bij een unit test wordt onderzocht of de ontwikkelde software daadwerkelijk doet waarvoor het gemaakt is. Met deze methode kunnen kleine losse stukjes logica (units) onafhankelijk van elkaar grondig getest. Wanneer een aanpassing in de software wordt gedaan, kan een unit test ervoor zorgen dat eventuele ongewenste effecten snel opgespoord kunnen worden. Daarnaast dwingt het je om nog een keer grondig je logica te bekijken voordat je deze oplevert

Functioneel testen is het testen van een product of een onderdeel om vast te stellen of het zal functioneren zoals bedoeld en of het door gebruikers overeenkomstig de bedoelingen van de makers zal worden gehanteerd.

In E2E testen wordt de software getest van de eindgebruikers perspectief, een realistische gebruikers scenario. Zoals in dit geval dat wanneer een gebruiker een component in een ander kan zetten met de verwachting dat dit zal werken.

Ook test ik de user interface, backend functionaliteiten, databases en de netwerkcommunicatie. Het doel van E2E testen is het valideren van het gedrag van de gehele applicatie:

* De functionaliteiten
* Betrouwbaarheid
* Prestatie
* Veiligheid

En waar ik bij E2E testen op doel is het identificeren van defecten en problemen die voorkomen op het moment dat verschillende delen van de applicatie met elkaar communiceren.

De reden dat ik voor E2E testen heb gekozen: Met E2E controleer ik de gehele applicatie op elk vlak. En dat vind ik de meest betrouwbare en efficiënte manier. Omdat je de applicatie op elke plek raakt en identificeert.

Voordelen van End to End Testen:

* Verbeterd de kwaliteit: End to End testen kan helpen bij het verzekeren dat elk deel van de softwareapplicatie correct met elkaar communiceren. En dat de applicatie voldoet aan de genoemde eisen.
* Meer vertrouwen in de software: End to End testen kan helpen bij het toenemen van het vertrouwen dat de software functionaliteiten werken zoals verwacht. En ook geeft de zekerheid aan belanghebbende dat de applicatie betrouwbaar en stabiel is.
* Sneller opmerken van defecten: End to End testen kan helpen bij het vroeg vinden van fouten in de software. Zo kunnen developers voorkomen dat ze in een nog groter en moeilijker probleem komen. Hiermee bespaar je dus ook eventuele kosten.
* Voorkomen van kosten: End to End testen helpt kan helpen bij het voorkomen van kosten. Het identificeert bugs en fouten in een eerder stadium hiermee zou je escalatie van een bug of fout kunnen voorkomen. Na het live zetten van een applicatie.
* Betere afstemming van eisen: End to End testen can helpen met het nakomen van eisen van de applicatie. Wat ervoor zorgt dat belanghebbende tevreden zullen zijn met de applicatie.
* Efficiënte test proces: End to End testen kan helpen met de test process van een gebruikersperspectief. Hiermee hoef je niet elk component individueel te testen, maar test je deze gelijk en ook nog eens of ze goed met elkaar communiceren. Wat het weer makkelijker maakt om fouten te identificeren.

Binnen het platform van Betty Blocks moet je testcode schrijven om End to End te kunnen testen.

Dit is de test code die ik geschreven heb:

A screen shot of a computer program

Description automatically generated with low confidence

Elk component die ik nu wil gaan testen moet ik een attribuut meegeven. Zoals je hier ziet is het meegegeven attribuut “Button-add-case” die is gekoppeld aan de button die een case opslaat.

Om te gaan testen moet ik eerst de terminal openen van het project. Met dit command start ik de test: npx playwright test case-overview --headed

In de code geef ik stap voor stap mee van begin tot eind wat ik wil gaan testen. Wat er in de code staat is:

* Start de applicatie op
* Ga naar de login pagina
* Log in met de meegegeven inloggegevens
* Druk op de submit knop
* Ga naar de case overzicht pagina
* Druk op de knop met het attribuut “Button-open-drawer”
* Vul de input velden in met de attributen: TextField-case-number, TextField-case-title, TextField-case-description, ‘Choose a status’ en ‘Choose a case type’.
* Druk op de knop met het attribuut “Button-cancel-add-case”.
* Druk weer op de knop met het attribuut “Button-open-drawer”
* Vul de input velden in met de attributen: TextField-case-number, TextField-case-title, TextField-case-description, ‘Choose a status’ en ‘Choose a case type’.
* Druk op de knop met het attribuut “Button-add-case”.

In mijn testcode zie je ook telkens waitForTimeout staan. Deze code zorgt ervoor dat de pagina de meegegeven seconden een timeout heeft. Dit doe ik om zo ook echt de gebruikerservaring te testen.

Met deze test doorloop ik de functionaliteiten van elk component, test ik de samenwerking van de componenten en test ik het proces van het gebruikersperspectief.

De pagina die ik getest heb voor deze userstory:

* Case overzicht

De vormconcepten zien er als volgt uit:

Dit is de Case overzicht pagina.

A screenshot of a computer

Description automatically generated with medium confidence

Blauwe button

Dit is de Case overzicht pagina die een formulier laat zien na dat er op de blauwe button met de naam “New” rechtsboven is gedrukt.

A screenshot of a computer

Description automatically generated with medium confidence

#### Scenario’s:

A screenshot of a computer

Description automatically generated with medium confidence

Klik op de blauwe button rechtsboven die als naam “New” heeft.

A blue rectangle with white text

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

Vul het formulier in met de dummy data.

Klik op save na het invullen van het formulier.

#### Gewenste resultaat:

Dit is de pagina van de case overzicht hoe ik hem verwacht.

A screenshot of a computer

Description automatically generated with medium confidence

Na dat ik op de blauwe “New” button heb geklikt wordt het formulier getoond.

A screenshot of a computer

Description automatically generated with medium confidence

Ik verwacht een validatie op elk veld, dus wanneer ik een veld leeg laat zal ik dit moeten zien, een melding dat het niet ingevulde veld verplicht is:

A screenshot of a computer

Description automatically generated with medium confidence

Dit is hoe het formulier eruitziet na het invullen.

A screenshot of a computer

Description automatically generated with medium confidence

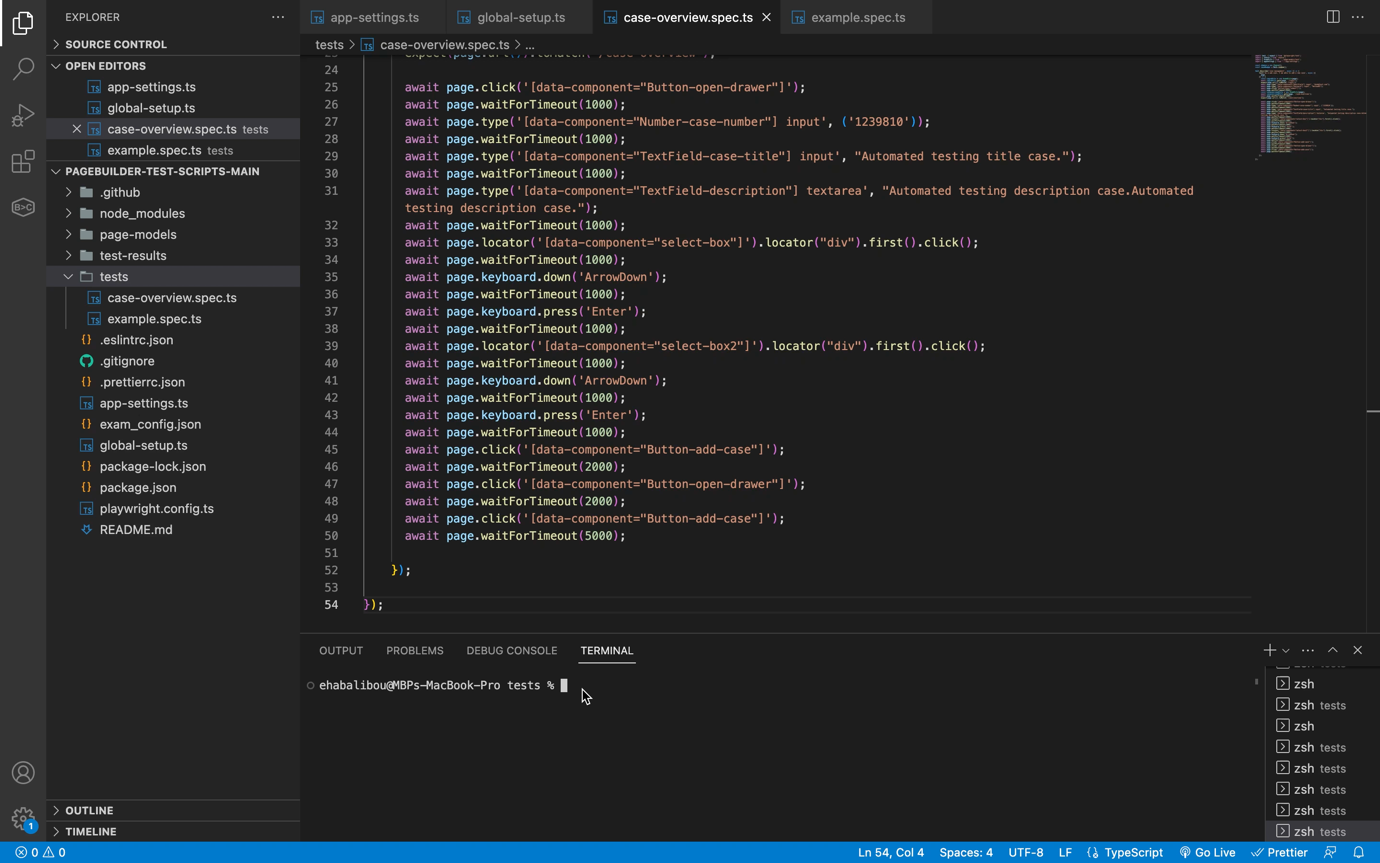
Het gewenste resultaat na dat je op de save button geklikt hebt.

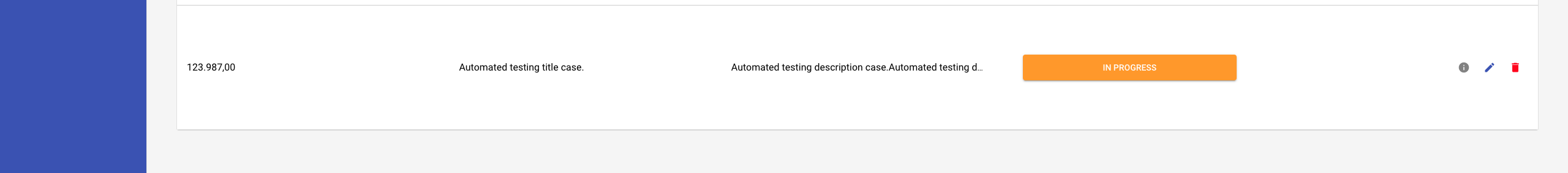
A screenshot of a computer

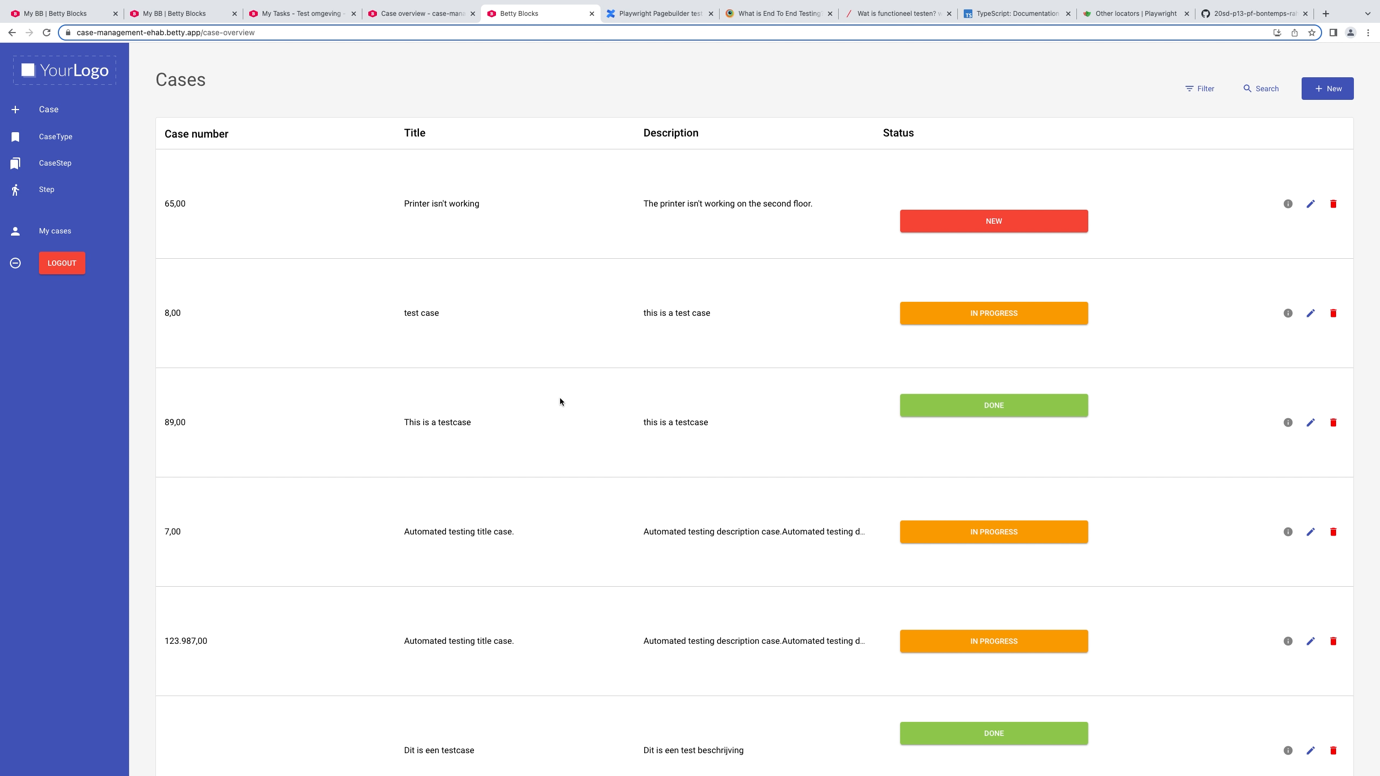
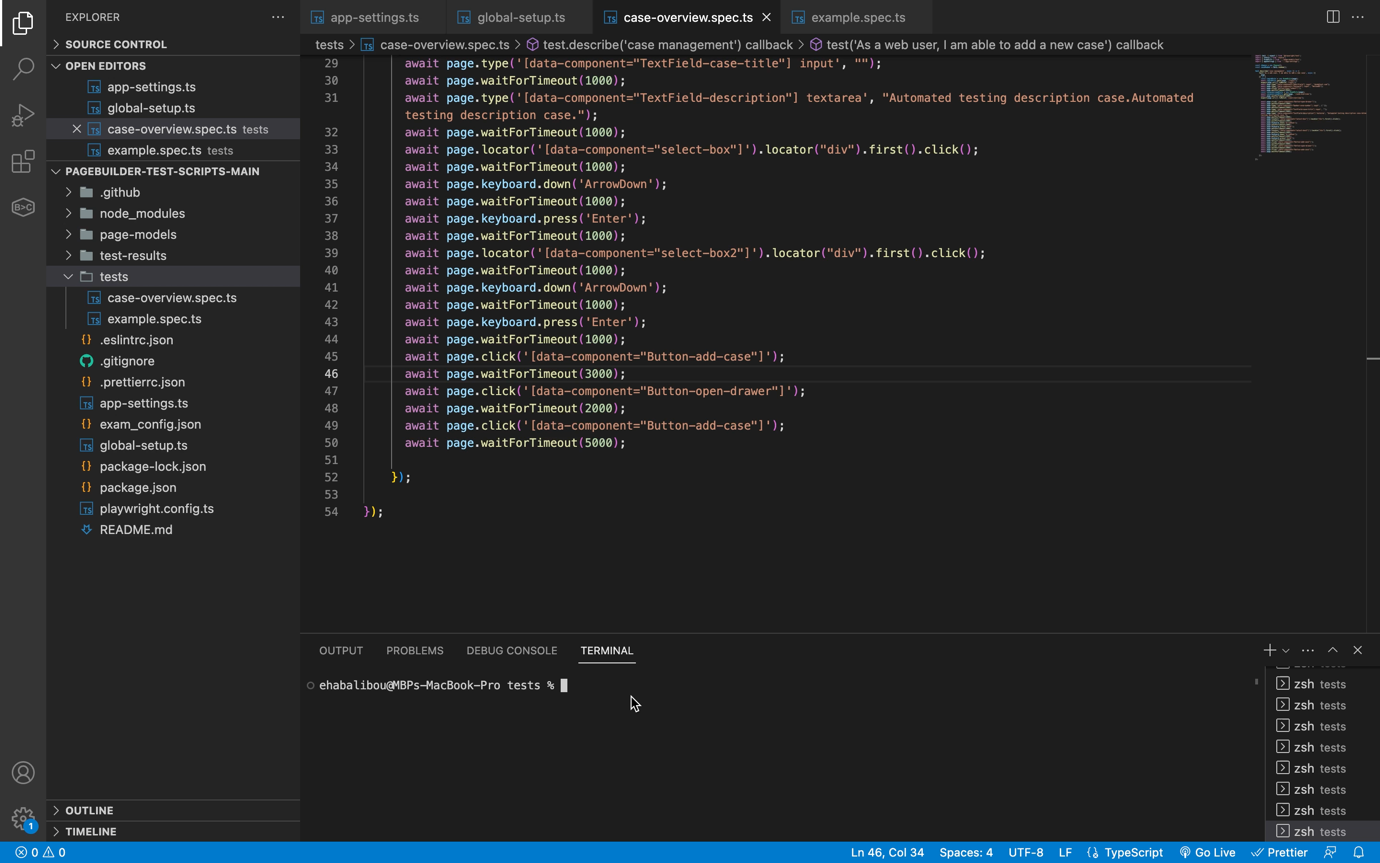
Description automatically generated with medium confidence

Een caserecord is met succes aangemaakt.

Gewenste resultaat van mijn test. De hieronder staande video (2x op klikken) laat zien hoe de test wordt uitgevoerd na dat ik de command heb uitgevoerd:



Het gewenste resultaat: een case record is aangemaakt:



Werkelijke resultaat: Hieronder staande video (2x op klikken) laat zien hoe de applicatie werkelijk werkt:

Werkelijke resultaat van mijn test: Hieronder staande video (2x op klikken) laat zien hoe de test werkelijk werkt:

Hieronder zie je een het testrapport waar alles overzichtelijk in staat:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Story** | **Scenario’s** | **Gewenst Resultaat** | **Resultaat (OK= goed, NOK = niet goed** | **Verbeteren** |
| **Als een gebruiker, wil ik een nieuwe caserecord kunnen aanmaken** | Ga naar de case overzicht pagina. | De case overzicht pagina wordt getoond | OK | n.v.t |
|  | Druk op de knop blauwe knop rechtsboven die als naam New heeft. | Het formulier om een case te kunnen aanmaken wordt getoond. | OK | n.v.t |
|  | Vul het formulier in die te zien is met dummy data.  En laat minimaal een veld leeg. | Het formulier wordt zonder obstakels ingevuld. | OK | n.v.t |
|  | Druk op de knop blauwe knop rechtsonder die als naam Save heeft. | Je krijgt een melding: “Een verplicht veld is niet ingevuld.” | NOK | Zorg ervoor dat er een validatie komt te staan op elk veld. |
|  | Controleer de case overzicht pagina op jouw toegevoegde case. | Er is geen case aangemaakt. | NOK | Zorg ervoor dat er geen case kan worden aangemaakt zonder de verplichte velden. |
|  |  |  |  |  |

Verbetervoorstellen

De verbetervoorstellen die ik nu zal benoemen zijn gebaseerd op het testrapport met de userstory: **Als een gebruiker, wil ik een nieuwe caserecord kunnen aanmaken**.

#### Verbetersvoorstel 1: Zorg ervoor dat er een validatie komt te staan op elk veld.

1. Identificeer alle velden waarop validatie moet worden toegepast in het Betty Blocks-platform.
2. Voor elk veld, configureer de juiste validatieregels op basis van de vereisten, zoals vereist, gegevenstype, patroon, minimale/maximale lengte, enz.
3. Stel duidelijke foutmeldingen in voor elke validatieregel die wordt overtreden.
4. Activeer de validatie op elk veld en test grondig om ervoor te zorgen dat de validatie correct werkt en de juiste foutmeldingen worden weergegeven.
5. Publiceer de bijgewerkte applicatie en voer een systeemtest uit om ervoor te zorgen dat de validatie effectief is en voldoet aan de vereisten.

Verbetervoorstel 2: Zorg ervoor dat er geen case kan worden aangemaakt zonder de verplichte velden.

1. Identificeer de verplichte velden die moeten worden ingevuld om een case aan te maken.
2. Implementeer een validatieregeling op de formulieractie van het aanmaken van een case om te controleren of alle verplichte velden zijn ingevuld.
3. Configureer een foutmelding die wordt weergegeven als een verplicht veld ontbreekt en voorkom dat de case wordt aangemaakt.
4. Test grondig om ervoor te zorgen dat de validatieregeling effectief is en de juiste foutmeldingen worden weergegeven wanneer verplichte velden ontbreken.
5. Publiceer de bijgewerkte applicatie en voer een systeemtest uit om te verifiëren dat de case niet kan worden aangemaakt zonder de verplichte velden in te vullen.

Met deze verbetervoorstellen wordt validatie toegepast op elk veld en wordt voorkomen dat een case wordt aangemaakt zonder de verplichte velden in te vullen.