**Testplan**

Project: Mijn Project

Project team: MinDef

Teamleden: Youri Saman (383661)

Roel Lucassen (386216)

Ruben Hemmes(studentnummer)   
 Twan Beeren (385731)

Alexander Lambooij

Luuk Ebenau

Opdrachtgever: Jean Pierre

Versie: 0.1

Versiedatum: 10 maart 2020

Status: Concept

# Documenthistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Wijzigingen | Auteur | Datum |
| 0.1 | Maken document | Roel Lucassen en Youri Saman | 10 maart 2021 |
| 1.0 | Hoofdstuk teststrategie toegevoegd | Roel Lucassen en Youri Saman | 11 maart 2021 |
|  | Feedback | Twan Mennen | 16-3-2021 |

Inhoud

[Documenthistorie 2](#_Toc66785114)

[Introductie 4](#_Toc66785115)

[Doel 4](#_Toc66785116)

[Omvang 4](#_Toc66785117)

[Uitgangssituatie 5](#_Toc66785118)

[Productrisicoanalyse 6](#_Toc66785119)

[Teststrategie 7](#_Toc66785120)

[Logische testcases 8](#_Toc66785121)

[Fysieke testcases 9](#_Toc66785122)

[Testcoverage 11](#_Toc66785123)

[H8 Unittesten en code coverage 12](#_Toc66785124)

[Statische code analyse 13](#_Toc66785125)

[Conclusie *(alleen van toepassing in testrapport)* 14](#_Toc66785126)

# Introductie

Het testplan is gemaakt om de testaanpak aan teamleden te communiceren. Het omvat de doelstellingen, scope, planning, risico's en aanpak. Dit document zal duidelijk aangeven wat de testresultaten zullen zijn en wat binnen en buiten de scope wordt geacht.

## Doel

MinDef is een verdedigingsapplicatie die gebruikt wordt om soldaten in het veld te helpen op het cybergebied. Deze applicatie is geschreven met ASP.NET Core met het gebruik van meerdere {soort} databases. Het test team is verantwoordelijk voor het testen van de applicatie en het ervoor zorgen dat alles ook getest wordt.

## Omvang

De eerste fase omvat alle ‘must have’ -vereisten. Deze en alle andere vereisten die worden opgenomen, moeten allemaal worden getest. Aan het einde van fase 1 moet een tester in staat zijn om:

1. In te loggen
2. Een account te registreren
3. Een accountcode te creëren
4. Een account te kunnen verwijderen
5. Alle accounts kunnen zien|

Terwijl het team met het product werkt, zullen ze de behoeften voor de tweede fase bepalen.

Het testen van de belasting op applicatie wordt wel gezien als onderdeel van dit project beschouwd, aangezien de hoeveelheid gebruikers onbekend is.

# Uitgangssituatie

Het testen van onze applicatie hebben we onderverdeeld in frontend testen en backend testen. De pipeline die gebruik maakt van CI/CD moet hierbij automatisch de testen uitvoeren bij het pushen van nieuwe code naar de master.

De front-end testen worden gedaan in het Angular framework. Hierbij gaan knoppen testen en kijken of deze de juiste stappen uitvoeren.

Daarnaast gaan we testen maken die in onze backend worden gedaan, waarbij we gebruik gaan maken van integration, controller, service en repository testen. Hierbij wordt gebruik gemaakt van mock-data zodat de daadwerkelijke gegevens in de database niet worden aangetast*.*

# Productrisicoanalyse

Productrisicoanalyse (PRA) wordt bepaald door per testdoel te bepalen: faalkans, schade, risicoklasse en ernstklasse. De risicoklasse is gelijk aan schade \* faalkans. De ernstklasse is gebaseerd op de risicoklasse. Geef aan hoe de ernstklasse wordt bepaald. Bijvoorbeeld:

Schade: 1 = Laag, 2 = Midden, 3 = Hoog

Faalkans: 1 = Laag, 2 = Midden, 3 = Hoog

Risicoklasse = Schade \* Faalkans

Ernstklasse C: risicoklasse ≤ 2

Ernstklasse B: 2 < risicoklasse ≤ 6

Ernstklasse A: risicoklasse > 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Testdoel** | **Schade** | **Faalkans** | **Risicoklasse** | **Ernstklasse** |
| FR-01 | **3** | **2** | **6** | **B** |
| FR-02 | **2** | **1** | **2** | **C** |
| FR-03 | **1** | **2** | **2** | **C** |
| FR-04 | **2** | **3** | **6** | **B** |
| FR-05 | **1** | **2** | **2** | **C** |
| FR-06 | **3** | **3** | **9** | **A** |
| FR-07 | **3** | **2** | **6** | **B** |
| FR-08 | **2** | **2** | **4** | **B** |
| FR-09 | **3** | **2** | **6** | **B** |
| FR-10 | **1** | **2** | **2** | **C** |
| FR-11 | **1** | **1** | **1** | **C** |
| NFR-1 | **2** | **2** | **4** | **B** |
| NFR-2 | **2** | **2** | **4** | **B** |
| NFR-3 | **1** | **1** | **1** | **C** |
| NFR-4 | **2** | **1** | **2** | **C** |

# Teststrategie

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Testdoel** | **PRA** | **UT** | **CT** | **IT** | **ST handm.** | **ST autom.** |
| FR-01 | **B** | □□ | □ | □ | □ |  |
| FR-02 | **C** | □ | □ |  | □ |  |
| FR-03 | **C** | □ | □ |  | □ |  |
| FR-04 | **B** | □□□ | □□ | □□ | □ |  |
| FR-05 | **C** | □ | □ |  | □ |  |
| FR-06 | **A** | □□ | □□□ | □□□ | □□ |  |
| FR-07 | **B** | □□ | □□ | □□ | □ |  |
| FR-08 | **B** | □□ | □ |  | □ |  |
| FR-09 | **B** | □ | □□ |  | □ |  |
| FR-10 | **C** | □ |  |  | □ |  |
| FR-11 | **C** | □ |  |  | □ |  |
| NFR-1 | **B** | □ | □□ | □□ | □ |  |
| NFR-2 | **B** | □□ | □ | □ | □ |  |
| NFR-3 | **C** | □ | □ | □ | □ |  |
| NFR-4 | **C** | □ | □ |  | □ |  |

Toelichting bij bovenstaande tabel:

|  |  |
| --- | --- |
| PRA-RK | Risicoklasse uit de productrisicoanalyse (PRA): risicotabel |
| Toetsen | Toetsing/review van de verschillende tussenproducten zoals requirements, functioneel ontwerp, technisch ontwerp |
| Ontwikkeltest | Unittest en Unitintegratietest |
| ST | Systeemtest |
| IT | Integratie test |
| CT | Component test |
| UT | Unit Test |
| □ | beperkte dynamische test |
| □□ | gemiddelde dynamische test |
| □□□ | zware dynamische test |

# Logische testcases

*In dit hoofdstuk worden de logische testcases beschreven. Een logische testcase is afgeleid van de requirements uit het analyse document. In het algemeen kunnen meerdere logische testcases worden gedefinieerd voor een requirement.*

*Voorbeeld van logische testcases:*

|  |  |
| --- | --- |
| **ID** | **Omschrijving** |
| TC-1 | Inloggegevens laten generen door commandant. |
| TC-2 | Inloggegevens laten genereren door een niet geautoriseerde “gebruiker” |
| TC-3 | Inloggen met juiste gegevens |
| TC-4 | Inloggen met verkeerde gegevens |
| TC-5 | Inloggen terwijl het wachtwoord mist |
| TC-6 | Inloggen terwijl de gebruikersnaam mist |

# Fysieke testcases

*In dit hoofdstuk worden de fysieke testcases beschreven. Een fysieke testcase is afgeleid van de bijbehorende logische testcase. Geef ook het verwachte resultaat. De kolom ‘resultaat’ is bedoeld voor het testrapport. In deze kolom wordt gebruik gemaakt van kleuren om verschillende resultaten aan te duiden. Hieronder een overzicht van de kleuren en hun betekenis:*

|  |  |
| --- | --- |
|  | De test is geslaagd en er is niks op aan te merken. |
|  | De test lijkt goed te gaan, maar er is iets op het behaalde resultaat aan te merken. |
|  | De test is gefaald. |
|  | Deze test is niet van toepassing op de huidige versie van de applicatie en is ook niet uitgevoerd. |
|  | Dit is geen test maar een actie die ondernomen moet worden om daar op volgende tests uit te kunnen voeren (initialisatie actie). |

*Voorbeeld van fysieke testcases:*

|  |  |  |
| --- | --- | --- |
| **ID** | **Omschrijving** | **Resultaat** |
| TC-1 | Maak een commandant aan met:  Gebruikersnaam: "Commandant1",  Wachtwoord: "P@sswW0rd!",  Log in met de gegevens  Genereer gegevens met de commandant voor een gebruiker.  Check of de string, die de methode terugstuurt, gelinked is met een gebruikersaccount  Verwacht resultaat: true |  |
| TC-2 | Maak een gebruiker aan met:  Gebruikersnaam: "gebruiker1",  Wachtwoord: "P@sswW0rd!",  Log in met de gegevens  Genereer gegevens met de gebruiker voor een gebruiker.  Check of de string, die de methode terugstuurt, gelinked is met een gebruikersaccount  Verwacht resultaat: false |  |
| TC-3 | Maak een user aan met:  Gebruikersnaam: “Username1”  Wachtwoord: “P@ssW0rd!”  Voeg deze user toe aan een lijst met users (als mock van een database)  Roep de login methode aan met:  Gebruikersnaam: “Username1”  Wachtwoord: “P@ssW0rd!”  Check wat de login methode terugstuurt.  Verwacht resultaat: true |  |
| TC-4 | Maak een user aan met:  Gebruikersnaam: “Username1”  Wachtwoord: “P@ssW0rd!”  Voeg deze user toe aan een lijst met users (als mock van een database)  Roep de login methode aan met:  Gebruikersnaam: “UsErname1”  Wachtwoord: “P@ssW1rd!”  Check wat de login methode terugstuurt.  Verwacht resultaat: false |  |
| TC-5 | Maak een user aan met:  Gebruikersnaam: “Username1”  Wachtwoord: “P@ssW0rd!”  Voeg deze user toe aan een lijst met users (als mock van een database)  Roep de login methode aan met:  Gebruikersnaam: “Username1”  Wachtwoord: “”  Check wat de login methode terugstuurt.  Verwacht resultaat: false |  |
| TC-6 | Maak een user aan met:  Gebruikersnaam: “Username1”  Wachtwoord: “P@ssW0rd!”  Voeg deze user toe aan een lijst met users (als mock van een database)  Roep de login methode aan met:  Gebruikersnaam: “”  Wachtwoord: “P@ssW0rd!”  Check wat de login methode terugstuurt.  Verwacht resultaat: false |  |

# Testcoverage

*In dit hoofdstuk wordt beschreven welke requirements afgedekt worden met welke testcases. Dit doe je aan de hand van een Requirements Traceabilty Matrix.*

*Voorbeeld van Requirements Traceability Matrix:*

|  |  |  |
| --- | --- | --- |
| **Acties**  **(alleen MUST)** | **TC(‘s)** | **Aantal Test cases** |
| FR-1 | TC1  TC2 | 2 |
| FR-2 | TC3  TC4  TC5  TC6 | 4 |

# H8 Unittesten en code coverage

*In dit hoofdstuk wordt beschreven welke strategie je hebt gehanteerd bij het schrijven van de unittesten. Beschrijf voor welke klassen en methoden je de unittesten gaat schrijven en waarom je voor deze klassen/methoden hebt gekozen. Laat de testresultaten en code coverage zien in het testrapport, bijvoorbeeld door een screenshot te maken. Beschrijf ook je conclusies op basis van de testresultaten en code coverage.*

# Statische code analyse

*In dit hoofdstuk wordt beschreven hoe je statische code analyse gaat onderzoeken. Dit kan met behulp van SonarQube. Laat de resultaten van de analyse zien in het testrapport, bijvoorbeeld door een screenshot te maken. Beschrijf ook je conclusies op basis van de resultaten en geef aan op welke punten je mogelijkheden ziet om de code verder te verbeteren.*

# Conclusie *(alleen van toepassing in testrapport)*

*Beschrijf hier de overall conclusie op basis van de resultaten van de uitvoering van de systeemtest, unittesten en statische code analyse. Geef ook aan of de applicatie kan worden opgeleverd, dat wil zeggen geef aan of de applicatie voldoet aan de requirements (functionele en niet-functionele eisen) zoals beschreven in het analyse document. Als dit niet zo is, geef dan aan wat er nog moet gebeuren voordat de applicatie opgeleverd kan worden.*