# Testing plan Het is nog niet helemaal consistent met use case diagram, waar komen de functies vandaan?

Nu het gehele systeem ontworpen is kunnen we testen ontwerpen die controleren of de implementatie werkt zoals gewenst. We gaan dit doen door vier verschillende testen uit te voeren:

1. Unit tests, waar wordt gecontroleerd of functies werken zoals verwacht door naar de parameters en return waarde te kijken.
2. Integration tests, waar wordt gecontroleerd of de samenwerking tussen functies werkt zoals verwacht.
3. System test, waar wordt gecontroleerd of het systeem voldoet aan de requirements.
4. Acceptance tests, dit zijn vragen voor de gebruikers om er achter of het systeem goed functioneert.

## Unit tests Nog 1 voor de Chauffeur?

Bij unit tests wordt gekeken of functies de verwachte waarden terugsturen. We hebben voor vijf methoden uit de class diagram unit tests gemaakt.

### Gebruiker.SaldoOpwaarderen()

De eerste unit test is voor de functie SaldoOpwaarderen() voor de class Gebruiker. Deze functie heeft de parameters huidig saldo en verhoging. De returnwaarde van de functie is het huidig saldo + de verhoging.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | HuidigSaldo | Verhoging | Verwachting | Output | Pass |
| 1 | 0 | 0,50 | 0,50 |  |  |
| 2 | 0 | 2,00 | 2,00 |  |  |
| 3 | 0 | 10,25 | 10,25 |  |  |
| 4 | 2,00 | 0,50 | 2,50 |  |  |
| 5 | 2,00 | 2,00 | 4,00 |  |  |
| 6 | 2,00 | 10,25 | 12,25 |  |  |
| 7 | 2,00 | 981,75 | 983,75 |  |  |
| 8 | 981,75 | 0 | 981,75 |  |  |
| 9 | 981,75 | 10,25 | 992,00 |  |  |
| 10 | 981,75 | 981,75 | 1963,50 |  |  |

Tabel 1 Unit test SaldoOpwaarderen()

Hierboven in tabel x is de unit test voor de functie SaldoOpwaarderen() te zien. Deze tabel test de uitkomst bij verschillende waarden. Het controleert wat er gebeurt als er met een klein bedrag wordt verhoogd, een groot bedrag, hetzelfde bedrag als het huidig saldo.

### Gebruiker.GegevernsAanpassen()

De functie GegevensAanpassen() van de class gebruiker past gegevens aan op het profiel van de gebruiker. Het geeft alle gegevens van een gebruiker mee en krijgt een nieuw gebruikersprofiel als returnwaarde.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Emailadres | Geslacht | Verwachting | Output | Pass |
| 1 | [abc@gmail.com](mailto:abc@gmail.com) | M | [abc@gmail.com,M](mailto:abc@gmail.com,M) |  |  |
| 2 | [abc@gmail.com](mailto:abc@gmail.com) | V | abc@gmail.com,V |  |  |
| 3 | [1234@gmail.com](mailto:1234@gmail.com) | M | 1234@gmail.com,M |  |  |
| 4 | [1234@gmail.com](mailto:1234@gmail.com) | V | 1234@gmail.com,V |  |  |

Tabel 2 Unit test GegevensAanpassen()

In de unit test in tabel wordt er naar twee verschillende velden gekeken. De daadwerkelijke unit test zal uitgebreider zijn. Het controleert simpelweg of de ingevoerde gegevens hetzelfde zijn als de returnwaardes.

### Meerijder.RitBevestigen() Onduidelijk wanneer? Is het een bevestiging als in reminder voor de rit, of is het een bevestiging als in acceptatie voor het meerijden? Reminder is een uur van tevoren, kan je die een kwartier van tevoren nogsteeds bevestigen dan.. en in het andere geval, is het handig dat mensen een kwartier van tevoren nog kunnen vragen om te carpoolen?

De functie rit bevestigen uit de class Meerijder is een functie dat veel andere functies aanroept. De gebruiker roept de functie aan als hij wilt bevestigen dat hij mee wilt gaan. De meegegeven waarde is de naam, datum en tijd. De returnwaarde is een SUCCES of FAIL. De functie moet een succes geven als de gebruiker op de lijst van geaccepteerde meerijders staat en de vertrektijd nog niet geweest is. Deze informatie staat opgeslagen in de Rit class. Hij zal dus via de class Bevestiging een antwoord moeten krijgen van de class Rit.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # | Naam | Datum + Tijd | Naam op geaccepteerde lijst (Rit class) | Vertrektijd (Rit class) | Verwachting | Output | Pass |
| 1 | Jan123 | 2018:10:17\_  9:27 | Ja | 2018:10:17\_  9:45 | SUCCES |  |  |
| 2 | Piet\_ABC | 2018:10:17\_  9:27 | Nee | 2018:10:17\_  9:45 | FAIL |  |  |
| 3 | Jan123 | 2018:10:17\_  9:46 | Ja | 2018:10:17\_  9:45 | FAIL |  |  |
| 4 | Piet\_ABC | 2018:10:17\_  9:46 | Nee | 2018:10:17\_  9:45 | FAIL |  |  |
| 5 | Jan123 | 2018:10:17\_  9:45 | Ja | 2018:10:17\_  9:45 | FAIL |  |  |

Tabel 3 Unit test RitBevestigen()

In tabel x is de unit test voor Ritbevestigen() te zien. Hij test verschillende waarden. Als de naam niet op de lijst van geaccepteerde meerijders staat wordt hij automatisch afgekeurd. Als de gebruiker de rit te laat bevestigd wordt hij ook afgekeurd. De unit test test ook als beide gegevens fout zijn en als de vertrektijd gelijk is aan de bevestigingstijd.

### Rit.MeerijderBevestigt() maximum meerijders?

De functie meerijderbevestigt() houdt bij hoeveel meerijders er zijn bevestigt bij een bepaalde rit. Dat gebeurd wanneer de naam van de meederijder op de lijst van geaccepteerde meerijders staat en bevestiging verstuurd is voor de vertrektijd van de rit. Als dit zo is wordt er een SUCCES verstuurd naar de bevestiging class en het aantal meerijders met 1 verhoogd.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Naam  (Bevestiging class) | Datum + Tijd  (Bevestiging class) | Naam op geaccepteerde lijst | Vertrektijd | Aantal bevestigde meerijders | Verwachting  Aantal bevestigde meerijders | Output | Pass |
| 1 | Jan123 | 2018:10:17\_  9:27 | Ja | 2018:10:17\_  9:45 | 2 | 3 |  |  |
| 2 | Piet\_ABC | 2018:10:17\_  9:27 | Nee | 2018:10:17\_  9:45 | 2 | 2 |  |  |
| 3 | Jan123 | 2018:10:17\_  9:46 | Ja | 2018:10:17\_  9:45 | 2 | 2 |  |  |
| 4 | Piet\_ABC | 2018:10:17\_  9:46 | Nee | 2018:10:17\_  9:45 | 2 | 2 |  |  |
| 5 | Jan123 | 2018:10:17\_  9:45 | Ja | 2018:10:17\_  9:45 | 0 | 1 |  |  |
| 6 | Jan123 | 2018:10:17\_  9:27 | Ja | 2018:10:17\_  9:45 | 9 | 10 |  |  |
| 7 | Jan123 | 2018:10:17\_  9:27 | Ja | 2018:10:17\_  9:45 | 5 | 6 |  |  |

Tabel 4 Unit test MeerijderBevestigd()

De unit test van de MeerijderBevestigd() functie is te zien in tabel x. Het laat zien dat het aantal bevestigde meerijders met 1 verhoogd wordt als de meerijder op de geaccepteerde lijst staat en op tijd de bevestiging heeft verstuurd.

### Meerijder.Betalen() negatief saldo voorkomen?

De betalen() functie verlaagt het saldo van de gebruiker. Het wordt opgeroepen vanuit de Rit class als de rit bevestigd is. De unit test is te zien in tabel x. Het is een relatief simpele unit test. De prijs van de rit wordt afgetrokken van het saldo van de meerijder.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | HuidigSaldo | Prijs | Verwachting | Output | Pass |
| 1 | -1.50 | 0,50 | -2,00 |  |  |
| 2 | 0,00 | 0,50 | -0,50 |  |  |
| 3 | 1,00 | 1,00 | 0,00 |  |  |
| 4 | 1,00 | 1,50 | -1,50 |  |  |
| 5 | 12,37 | 2,00 | 10,37 |  |  |
| 6 | 25,60 | 3,50 | 22,10 |  |  |

Tabel 5 Unit test Betalen()

## Integration tests 5 toegelicht, de rest ook doen?

Bij de unit tests wordt geverifieerd dat afzonderlijke componenten goed functioneren. Als dit geverifieerd is kunnen integration tests worden toegepast. Deze controleren of de samenhang van componenten goed werkt. Ze controleren bijvoorbeeld of alle nodige data goed wordt uitgewisseld en deze data goed gebruikt wordt in de units. In dit testplan worden vijf integration tests toegelicht.

### Betaling meerijder

Het versturen van de betaling wordt gedaan door de betaling class en wordt afgehandeld door de meerijder class. Bij deze test moet altijd het juiste bedrag worden afgeschreven van het saldo van de meerijder als hij de rit bevestigd. Het doel van deze test is om dit te controleren, dit wordt gedaan door de samenhang tussen alle betrokken componenten te controleren. Het wordt gedaan vanuit een top down method.

De betaling van een meerijder gaat als volgt:

1. De rit class registreert dat de meerijder de rit heeft bevestigd.
2. De rit class maakt een betaling aan.
3. De betaling wordt naar de meerijder verstuurd.
4. De meerijder regelt de betaling waardoor er saldo wordt afgeschreven van de gebruiker.

De registratie van de bevestiging wordt getest zoals aangegeven bij de unit test. Vervolgens wordt er een betaling aangemaakt. Deze betaling moet de juiste meerijder, chauffeur, prijs en RitID hebben. De betaling moet naar de meerijder worden verstuurd, niet de chauffeur. De meerijder moet de prijs en RitID van de rit ontvangen. Deze prijs wordt van het saldo afgetrokken zoals in de unit test is beschreven. Als het gehele systeem werkt wordt dus het juiste bedrag eenmalig worden afgeschreven van de gebruiker.

### Betaling ontvangen chauffeur

Het ontvangen van de betalingen van een chauffeur lijkt erg op het betalen van een meerijder. Bij deze test wordt er gekeken of het juiste bedrag wordt toegevoegd aan het saldo van de gebruiker. Het doel is om te controleren of het juiste bedrag inderdaad wordt toegevoegd aan het saldo van de gebruiker. Dit wordt weer via een top down methode gedaan.

Het ontvangen van een betaling gaat als volgt:

1. De rit class registreert dat de meerijder de rit heeft bevestigd. (?)
2. De rit class maakt een betaling aan.
3. De betaling class verstuurd de betaling naar de chauffeur.
4. Het saldo van de chauffeur wordt verhoogd met de prijs van de rit min 50 cent.

De registratie van de bevestiging wordt getests zoals aangegeven bij de unit test. Vervolgens wordt er een betaling aangemaakt. Deze betaling moet de juiste meerijder, chauffeur, prijs en RitID hebben. Het bericht moet naar de chauffeur worden verstuurd, niet de meerijder. De chauffeur moet de juiste prijs, RitID en meerijder meegestuurd krijgen. Vervolgens zal de chauffeur de prijs min 50 cent bij zijn saldo optellen.

### Rit aanmaken

Als een chauffeur een rit wilt invoeren in het systeem zal hij dit moeten doen via de nieuwe rit knop. Vervolgens vult hij alle gegevens in en drukt op de ‘nieuwe rit aanmaken’ knop. De rit zou dan zichtbaar moeten zijn voor alle ingelogde gebruikers. Het is belangrijk om dit te testen doormiddel van een integration test.

Het doel van deze test is om te controleren dat wanneer een chauffeur een nieuwe rit aanmaakt, de rit zichtbaar is voor alle gebruikers en de data van de rit klopt. Het wordt een test met een top down method.

De chauffeur voert alle gegevens in over de rit. Vervolgens wordt er een RitID aangemaakt en wordt de rit toegevoegd aan de database. Daarna kan de gebruiker de LijstOpvragen() functie oproepen die de lijst met beschikbare ritten update. De nieuwe rit zal nu toegevoegd moeten zijn en alle data over de rit moet overeenkomen met de ingevoerde data van de chauffeur.

### Rit aanvragen

Een ingelogde gebruiker kan een aanvraag indienen, hiervoor moet een melding komen bij de chauffeur en de chauffeur kan hierop reageren. De samenwerking tussen deze componenten moet getest worden met een integration test. Het doel hiervan is om te testen of een meerijder een rit kan aanvragen en of de chauffeur het kan goed- en afkeuren. Dit wordt gedaan door een top down method.

De stappen die het systeem onderneemt om dit te realiseren zijn:

1. Een gebruiker dient een aanvraag in.
2. Het systeem maakt hier een aanvraag object voor aan, deze moet de juiste RitID, Chauffeur en Meerijder hebben.
3. De juiste chauffeur moet een notificatie krijgen dat er een aanvraag is. In deze notificatie moet de juiste RitID en meerijder meegestuurd zijn.
4. De chauffeur kan de aanvraag accepteren of weigeren, dit bericht moet worden ontvangen door de het Rit object
5. Als de chauffeur de aanvraag heeft geaccepteerd moet de meerijder op de lijst van geaccepteerde meerijders, ander niet.

### Rit bevestigen

Voordat een gebruiker mee kan rijden moet hij eerst de rit bevestigen. Dit vraagt om een samenwerking van veel verschillende componenten in het systeem en zal getest moeten worden met een integration test. Het doel van de test is om te verifiëren dat een gebruiker altijd een rit kan bevestigen.

De stappen van het bevestigen van een rit zijn als volgt:

1. De rit verstuurd automatisch een bericht naar de meerijder als het een uur voor de vertrektijd is. In dit bericht wordt het RitID en de naam van de chauffeur meegestuurd.
2. De meerijder moet een melding krijgen van dit bericht.
3. Als de meerijder de rit bevestigd moet dit worden geregistreerd worden in de rit.

Als het gehele bevestigingssysteem werkt staat de meerijder op de lijst van bevestigde meerijders nadat hij de rit heeft bevestigd.

## System tests

Als de afzonderlijke onderdelen en hun relaties getest zijn kan worden getest of de eisen behaald zijn. In dit testplan zullen we een test noemen voor 5 verschillende requirements.

### Requirement #x

De gebruiker kan een account aanmaken. Dit is te testen door de volgende stappen te doorlopen:

1. Start de applicatie.
2. Vul een gebruikersnaam en wachtwoord in.
3. Druk op inloggen, de applicatie geeft een bericht dat de gegevens niet kloppen.
4. Druk op account aanmaken.
5. Vul gegevens in.
6. Druk op bevestigen.
7. Vul opnieuw gebruikersnaam en wachtwoord in.
8. Druk op inloggen.

Als de gebruiker is ingelogd op het nieuwe account is het account succesvol aangemaakt.

### Requirement #x

De gebruiker moet inloggen voordat hij de functionaliteiten van de app kan gebruiken. De test is hiervoor gaat als volgt:

1. Start de applicatie.
2. Druk op alle beschikbare knoppen.
3. Vul foute inloggegevens in.
4. Druk op inloggen, de applicatie geeft een bericht dat de gegevens niet kloppen.
5. Vul juiste inloggegevens in.
6. Druk op inloggen.

Als de gebruiker pas naar het hoofdscherm gaat wanneer de juiste gegevens zijn ingevuld is de test geslaagd.

### Requirement #3

Een chauffeur kan een rit invoeren in de database. Daarbij hoort een op- en uitstapplaats en aantal mogelijke meerijders. Het aantal mogelijke meerijders is later niet meegenomen in het ontwerp omdat het de complexiteit verhoogt, hoeft dus niet getest te worden.

De test voor de rest van deze requirement gaat als volgt:

1. Inloggen.
2. Druk op de knop rit aanmaken.
3. Vul de gegevens in.
4. Zoek de rit op in het hoofdmenu.
5. Controleer of alle ingevoerde gegevens bij de rit staan.

Als alle ingevoerde gegevens online staan is de test geslaagd.

### Requirement #7

Een meerijder kan 1 of meerdere aanvragen indienen voor een rit. Dit is kan getest worden door de volgende stappen te ondernemen:

1. Inloggen.
2. Twee of meer ritten aanmaken.
3. Uitloggen.
4. Inloggen op een ander account.
5. Aanvragen indienen voor twee of meer ritten.
6. Uitloggen.
7. Inloggen op het oorspronkelijke account.
8. Controleren of er voor de verschillende ritten aanvragen zijn ontvangen.

De test is geslaagd als alle aanvragen zijn ontvangen op het oorspronkelijke account.

### Requirement #14

De webapplicatie moet een historie functie hebben waarmee gereden ritten opgeslagen worden en bekeken kunnen worden door de gebruiker. Dit is te testen door:

1. Inloggen.
2. Een aantal ritten aanvragen en bevestigen.
3. Naar het rithistoriescherm gaan.
4. Controleren of alle bevestigde ritten hier aanwezig zijn.

Als alle bevestigde ritten in het rithistoriescherm staan is de test geslaagd.

## Acceptance test

Wat willen we weten van de gebruikers?

* Plaats, om te kijken waar de meeste carpoolers zijn (voor promotie e.d.)
* Welk studiejaar, om te kijken in welk jaar er meer carpoolers zijn (voor promotie e.d.)
* Suggesties voor verbeteringen via een formulier
* Of de rit is bevallen. Zo niet, waarom?
* Hoe kom je bij carpool4uni?
* Service