|  |
| --- |
| TU Delft |
| Carpool4Uni |
| TB241IA, I&C Systeemanalyse |

|  |
| --- |
| Groep 14:  4975243, Floris Kool  4702867, Tommy Oosters  4658787, Quint Sullock Enzlin  7-11-2018 |

# Inhoudsopgave

[1 Inleiding 3](#_Toc529395859)

[2 Requirements analysis 4](#_Toc529395860)

[2.1 Elicitatieproces 4](#_Toc529395861)

[2.2 Inleiding 4](#_Toc529395862)

[2.3 Project Issues 4](#_Toc529395863)

[2.4 Randvoorwaarden 4](#_Toc529395864)

[2.5 Functionele eisen 5](#_Toc529395865)

[2.6 Niet functionele eisen 5](#_Toc529395866)

[3 Use cases 6](#_Toc529395867)

[3.1 Use cases 6](#_Toc529395868)

[3.2 Use case diagram 7](#_Toc529395869)

[4 Interaction design 9](#_Toc529395870)

[4.1 Interaction design onderdelen 9](#_Toc529395871)

[5 Architecture design 20](#_Toc529395872)

[5.1 Swimlane diagram 20](#_Toc529395873)

[5.2 Sequence diagram 25](#_Toc529395874)

[5.3 Class diagram 28](#_Toc529395875)

[6 Database design 29](#_Toc529395876)

[6.1 Entity Relationship Diagram 29](#_Toc529395877)

[6.2 Database diagram 30](#_Toc529395878)

[6.3 SQL queries 31](#_Toc529395879)

[7 Testing plan 33](#_Toc529395880)

[7.1 Unit tests 33](#_Toc529395881)

[7.2 Integration tests 36](#_Toc529395882)

[7.3 System tests 38](#_Toc529395883)

[7.4 Acceptance tests 40](#_Toc529395884)

[Bijlagen 41](#_Toc529395885)

[Bijlage 1 Functionele requirements 41](#_Toc529395886)

[Bijlage 2 Niet functionele requirements 47](#_Toc529395887)

[Bijlage 3 Use cases 48](#_Toc529395888)

[Bijlage 4 Swimlane diagram 54](#_Toc529395889)

[Bijlage 5 Entity Relationship Diagram 59](#_Toc529395890)

[Bijlage 6 Database diagram 60](#_Toc529395891)

# Inleiding

Iedere dag gaan er duizenden studenten, docenten en medewerkers naar universiteiten in Nederland. Ze doen dit doormiddel van het OV, de fiets of met de auto. Het OV is vaak erg druk, de fiets is vermoeiend en het is niet duurzaam als iedereen met de auto zou gaan. Het bedrijf Carpool4Uni zag dit als een kans om de transportatie naar de universiteit te verbeteren door een carpoolwebsite op de markt te brengen. De website zou op een meer duurzame en prettige manier mensen van en naar verschillende universiteiten brengen.

Het bedrijf kan echter zelf geen software ontwikkelen, dit is waar wij als software ontwikkelaars aan bod komen. Met dit verslag willen wij daarom antwoord geven op de probleemstelling: *“Hoe wordt een carpoolwebsite ontwikkeld dat voldoet aan alle eisen van Carpool4Uni?”* Dit wordt gedaan door het gehele software engineeringproces te doorlopen.

Het project begint met het opstellen van eisen in hoofdstuk 2. Vervolgens worden de use cases van deze eisen toegelicht in hoofdstuk 3. Daarna, in hoofdstuk 4 wordt de interface ontworpen. Nadat de interface duidelijk is wordt in hoofdstuk 5 het ontwerp van het gehele systeem in kaart gebracht doormiddel van verschillende diagrammen. Het systeem vereist veel opgeslagen data, daarom wordt in hoofdstuk 6 het ontwerp van de database toegelicht. Als het gehele ontwerp klaar is wordt er een testplan opgesteld in hoofdstuk 7.

# Requirements analysis

Voordat het systeem ontworpen kan worden moet er eerst worden overlegt met de opdrachtgever om er achter te komen wat ze precies willen. Dit hebben wij gedaan en hebben als resultaat hiervan een lijst met eisen. Dit hoofdstuk beschrijft deze eisen en hoe ze tot stand zijn gekomen.

## Elicitatieproces

We hebben vragen voorbereid en een interview gehouden met groep 1. Tijdens het interview kwamen meer vragen naar boven. Op basis van de antwoorden op de vragen hebben we een concept requirements document opgesteld. In eerste instantie hadden wij de rollen cliënt-ontwikkelaars omgedraaid. Om die reden hebben wij dit requirements-document verder laten beoordelen door groep 13, zoals de bedoeling was. Groep 13 hebben bovendien nog een aantal requirements extra aangedragen of wilde bestaande requirements wat omvormen naar hun visie op Carpool4Uni. Na de beoordeling van groep 13 op ons requirements document hebben wij nog een aantal punten aangepast, waarvan wij toch niet helemaal goed hadden begrepen hoe zij het wilden.

## Inleiding

Het systeem zal bestaan uit een webapplicatie dat mensen helpt met het carpoolen. De webapplicatie laat studenten/docenten en medewerkers op een goedkope en meer milieuvriendelijke manier reizen naar de universiteit.

## Project Issues

Het project zal uiterlijk op 8 november afgerond zijn. Wij kunnen hiervoor 8 uur per week per persoon aan werken wat uitkomt op een totaal van 288 uur. We zullen zo veel mogelijk gebruik maken van het materiaal dat aangeleverd wordt bij de module om hiermee tijd te kunnen besparen. We zullen zo’n 1 á 2 keer per week een overlegmoment hebben met de cliënt om bepaalde zaken te overleggen. Verder is er geen budget om overige software/middelen aan te schaffen. Uiteraard wordt de klant nauw betrokken bij het opstellen van de requirements en later in de ontwikkeling bij het testen.

## Randvoorwaarden

De randvoorwaarden voor dit project zijn:

* Het systeem wordt uitgeoefend als een webapplicatie.
* De web-app laat mensen in contact komen om te carpoolen van en naar de universiteit.
* De web-app moet bruikbaar zijn voor alle studenten, docenten en medewerkers van alle Nederlandse universiteiten.

## Functionele eisen

In samenwerking met de opdrachtgever is er een volledige lijst aan functionele eisen opgesteld. De functionele eisen zijn:

1. De gebruikers moet inloggen voordat ze toegang hebben tot de rest van de site.
2. Chauffeurs en meelifters moeten een profiel aanmaken en kunnen daar belangrijke, voor andere gebruikers openbare, gegevens kwijt.
3. Ieder profiel heeft een wallet met een tegoed dat kan worden opgewaardeerd of opgenomen (vanaf 1€) via paypal, IDeal of een bankoverdracht
4. Een chauffeur kan een rit invoeren in de database. Daarbij hoort een op- en uitstapplaats en aantal mogelijke meerijders.
5. De prijs van een rit wordt bepaald door een vast kilometertarief, die gebaseerd is op het type auto van de chauffeur. Bovenop de prijs komt een percentage van het bedrag dat wordt betaald door de meerijder en dat gaat naar Carpool4Uni.
6. Naderhand moeten chauffeur en meerijder de rit bevestigen in de applicatie. Zodra dit gedaan is wordt het geld automatisch afgeschreven van het tegoed van de meerijder en toegevoegd aan het tegoed van de chauffeur.
7. De passagier moet een rit kunnen selecteren uit een lijst van alle ritten naar die universiteit en aan de hand van zijn locatie. De chauffeur kan de passagier accepteren of afwijzen. Een meerijder moet een overzicht kunnen zien van beschikbare ritten.
8. Een meerijder kan 1 of meerdere aanvragen indienen voor een rit.
9. Een chauffeur krijgt een mailtje/sms/app als er een aanvraag wordt ingediend voor zijn rit.
10. Een uur voor de rit krijgen zowel de chauffeur als meerijder een melding/herinnering voor hun rit.
11. Nadat een passagier een rit heeft geselecteerd en de chauffeur de aanvraag heeft geaccepteerd, wordt de chatfunctie actief. De chatfunctie geeft een open communicatie tussen passagier en chauffeur.
12. Gebruikers moeten een beoordeling (rating en plek voor recensies) kunnen krijgen die te zien is op zijn/haar profiel en zich uitsluitend baseert op betrouwbaarheid. Daarbij helpt een optioneel persoonlijk stukje om een indruk te krijgen over de chauffeur of meerijder.
13. De lay-out van de webapplicatie is minimalistisch strak met wit en groene kleuren.
14. De webapplicatie moet een trackfunctie bevatten waarmee de chauffeur te tracken is.
15. De webapplicatie moet een historie functie hebben waarmee gereden ritten opgeslagen worden en bekeken kunnen worden door de gebruiker.

De volledige uitwerking van deze eisen is te vinden in bijlage 1.

## Niet functionele eisen

De volgende niet functionele eisen zijn tot stand gekomen:

1. De eerste vier maanden vindt een maandelijkse reflectie plaats met cliënt en/of gebruikers. Afhankelijk van het succes wordt er nog gekeken naar de frequentie van reflecteren na deze 4 maanden.
2. Het moet een webapplicatie zijn.

De uitwerking van deze eisen is te vinden in bijlage 2.

# Use cases

Nu de eisen duidelijk zijn gemaakt worden er use cases bij gemaakt. Dit zijn scenario’s die aangeven op welke manier verschillende gebruikers het systeem kunnen gebruiken. Doormiddel van de use cases worden de functionaliteiten meer expliciet neergezet en wordt er een eerste stap gedaan naar een ontwerp.

Voor dit project zijn er 9 use cases. Dit hoofdstuk zal ze allemaal kort beschrijven, de volledige uitwerking is te vinden in bijlage 3. Verder wordt een use case diagram weergegeven die de interactie tussen de actoren en de use cases laat zien.

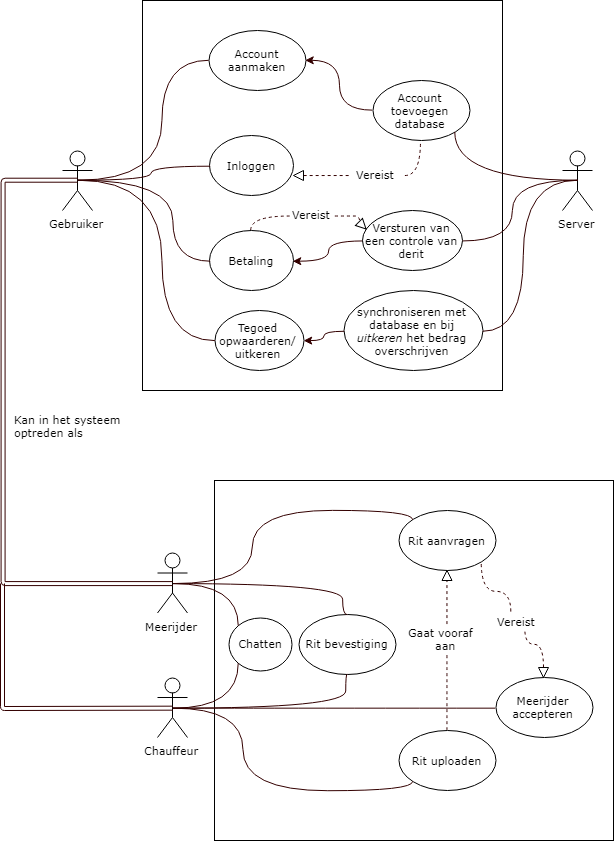
## Use cases

De use cases voor dit project zijn:

1. Account aanmaken. Elke gebruiker moet een account kunnen aanmaken om gebruik te maken van de website.
2. Inloggen. De gebruiker moet ingelogd zijn, omdat alle acties binnen de website verbonden zijn een gebruikersaccount.
3. Rit invoeren. De chauffeur kan een nieuwe rit invoeren in het system. Deze wordt zichtbaar voor alle ingelogde gebruikers.
4. Rit aanvragen. Een meerijder kan een verzoek tot aanvragen indienen bij een rit.
5. Meerijder accepteren. De chauffeur moet de meerijder accepteren voordat een meerijder mee mag rijden.
6. Rit bevestigen. De meerijder en chauffeur moeten de rit bevestigen een uur voor de rit. Zo wordt voorkomen dat er mensen niet komen opdagen.
7. Betaling. De betaling verloopt automatisch na het bevestigen.
8. Tegoed opwaarderen/uikteren. Betalingen gaan via een saldo, de gebruiker moet dit saldo kunnen opwaarderen en opnemen.
9. Chatten. Wanneer een chauffeur een meerijder heeft geaccepteerd moeten de gebruikers kunnen chatten.

## Use case diagram

In het onderstaande afbeelding is de use case diagram te vinden. Op de volgende pagina wordt deze toegelicht.



Figuur 1 Use case diagram

In figuur 1 wordt een afbeelding weergegeven waarbij onderscheid gemaakt kan worden in het bovenste en onderste gedeelte. In het bovenste gedeelte zien we een gebruiker, een rechthoek met daarin tekstballonnen en rechts de server. De tekstballonnen suggereren use cases. In het onderste gedeelte zien we een meerijder, een chauffeur en het systeem met interacties (tekstballonnen). Het onderste gedeelte suggereert dat gebruikers in het systeem kunnen optreden als een chauffeur (rit uploaden), maar ook als meerijder (rit aanvragen). Dit is op deze wijze gevisualiseerd met de reden dat zij wel eenzelfde account hebben. Zo hoeven chauffeurs en meerijders niet meerdere accounts te hebben wanneer ze een keer willen meerijden in plaats van zelf rijden.

De gebruiker (in het algemeen) kan handelingen uitvoeren (use cases), zoals een account aanmaken, inloggen en zijn tegoed opwaarderen of uit laten keren. Er is een verband tussen een account maken en inloggen. Bij het aanmaken van het account wordt deze namelijk opgeslagen op de server. Het is een vereiste dat wanneer iemand wilt inloggen, er wel een account is. Dat wordt tijdens het inloggen gecontroleerd bij de database. Verder vindt er ook een betaling plaats voor de rit. Hiervoor wordt een controle voor de rit verstuurt door het systeem. Chauffeur en mee- moeten deze bevestigen. Dat is een vereiste voordat de betaling daadwerkelijk plaatsvindt. Een gebruiker kan ook zijn tegoed opwaarderen of laten uitkeren. Het systeem moet daarvoor deze wijziging in het tegoed synchroniseren met het systeem en bij een uitkering ook het bedrag overschrijven naar de rekening van deze persoon.

Het onderste gedeelte van de afbeelding laat de interactie zien tussen meerijder en chauffeur. Ze kunnen beiden chatten met elkaar (mits de ritaanvraag is bevestigd). Een meerijder kan een rit aanvragen. Daaraan gaat vooraf dat een chauffeur een rit heeft geüpload. Een chauffeur kan een meerijder accepteren. Daarvoor is vereist dat er een rit is aangevraagd. De rit afwijzen hebben we omwille van het overzicht hier niet in verwerkt. De meerijder en chauffeur behoren de rit te bevestigen achteraf. Bovendien wordt hiermee ook bedoelt dat de chauffeur en meerijder een uur alvorens de rit een bevestiging ontvangen.

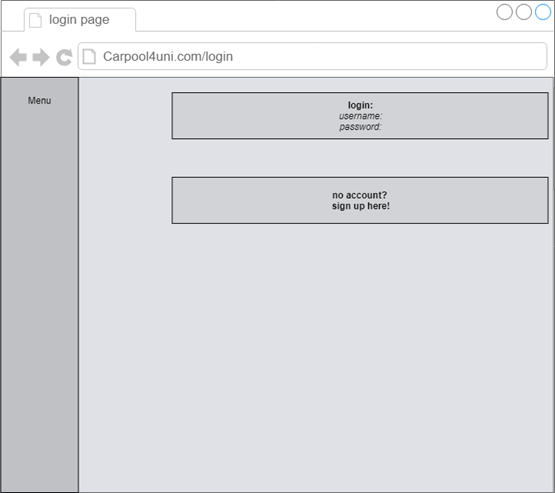
# Interaction design

Na overleg met de opdrachtgever is nu duidelijk aan welke eisen het project moet voldoen en welke stappen de gebruiker moet doen om gebruik te maken van de functionaliteit. In dit hoofdstuk wordt het eerste deel van het ontwerp uitgewerkt. Dit is een interaction design, hierin wordt doormiddel van functionele diagrammen weergegeven hoe de website er uit komt te zien. Het gaat hier voornamelijk om de werking van de website te weergeven, styling maakt nog niet veel uit.

## Interaction design onderdelen

Per onderdeel wordt een figuur gegeven en een beschrijving hoe de pagina gebruikt kan worden.

### Login



Figuur 2 Login

De loginpagina (figuur 2) is de pagina waar een bezoeker van de website als eerst terechtkomt. Hier kan de bezoeker inloggen op zijn account door in de bovenste balk zijn gegevens in te toetsen en dan op de knop te klikken.

Wanneer de bezoeker van de website nog geen account heeft klikt hij op de andere knop, wanneer deze hier op drukt komt deze terecht op de pagina waar een account aangemaakt kan worden.

### Sign up



Figuur 3 Sign up

Figuur 3 geeft de sign up pagina weer. Op deze pagina maakt een nieuwe bezoeker een account aan om gebruik te kunnen maken van de website. Hier moet de bezoeker de volgende gegevens invullen:

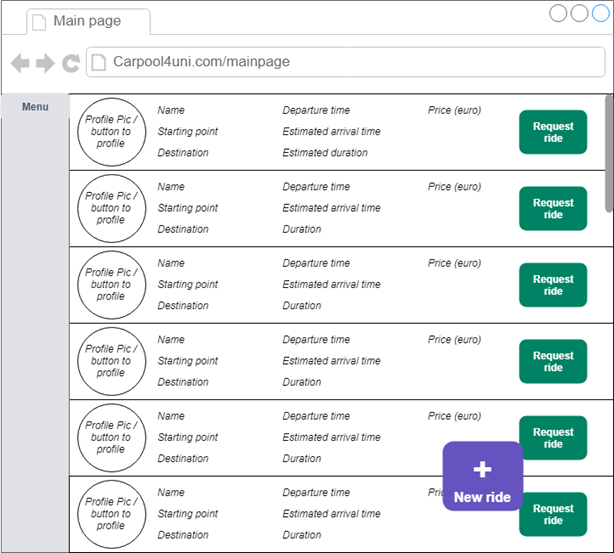
* Gebruikersnaam
* Geslacht
* Leeftijd
* Functie(student/werknemer)
* roken toegestaan(ja/nee)
* bankgegevens
* e-mail
* wachtwoord
* profielfoto

De volgende gegevens zijn eventueel:

* grootte van de auto (klein/middel/groot)
* type van de auto
* extra opmerkingen

Wanneer deze gegevens zijn ingevuld wordt het account aangemaakt door op sign up te klikken. Als er toch geen account aangemaakt dient te worden kan er op elk moment op cancel geklikt worden en dan komt de website terug op de loginpagina.

### Main page



Figuur 4 Main page

De hoofdpagina is waar een ingelogde gebruiker ritten kan bekijken, een nieuwe rit kan aanmaken en een menu kan openen. Het is te zien in figuur 4.

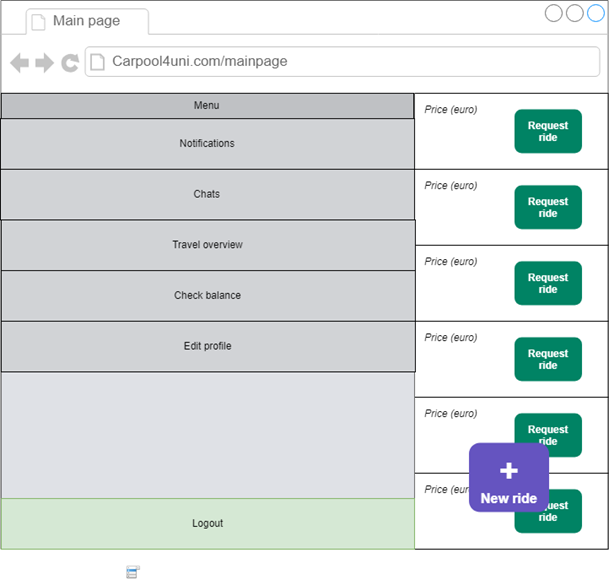
Het menu is een balk aan de linkerkant met een knop links boven. Als er op de knop gedrukt wordt opent het menu.

Het belangrijkste onderdeel van de hoofdpagina is een lijst met beschikbare ritten. Dit is een lijst met een scroll bar met identieke items. Deze items bevatten de volgende informatie:

* Een profielfoto van de chauffeur die dient als knop om naar het profiel van de chauffeur te gaan.
* Naam van de chauffeur.
* Opstappunt
* Bestemming
* Vertrektijd
* Verwachte aankomsttijd
* Verwachte reistijd
* Prijs in euro’s
* Grote knop om de rit aan te vragen

Naast de lijst en de menuknop is er een grote knop met een + icoon om een nieuwe rit aan te maken.

### Menu



Figuur 5 Menu

In het menu (figuur 5) kan de gebruiker schakelen tussen de verschillende pagina’s. Visueel is het menu een balk die vanaf de linker kant een groot deel van het scherm inneemt. Als er op het rechterdeel van het scherm wordt geklikt of op de menu knop verdwijnt het menu.

De knoppen op het menu zijn:

* Hoofdpagina
* Notificaties
* Chats
* Saldo bekijken
* Profiel aanpassen
* Ritgeschiedenis
* Uitloggen

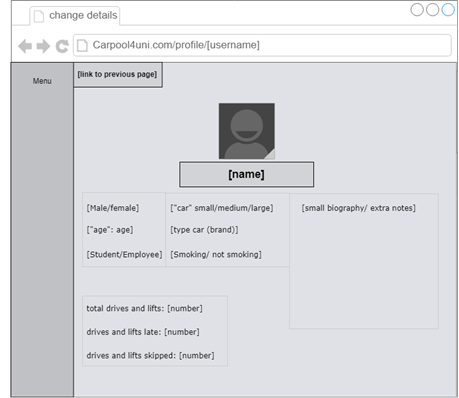
### Change details



Figuur 6 Change details

Op de change details pagina (figuur 6) kan een gebruiker zijn gegevens wijzigen. Door in het vak van het gegeven dat aangepast wil worden een aanpassing te brengen en dan op save changes te klikken, wordt het opgeslagen en aangepast. Als er aanpassingen zijn gemaakt maar je wil dit ongedaan maken klik je op cancel.

### Profile



Figuur 7 Profile

Op de profielpagina, te zien op figuur 7 zijn de voorkeuren van de gebruiker te zien. Verder zie je een foto en wat persoonlijke informatie die prettig is om voor de rit te weten. Er kan een klein persoonlijk stukje op worden geplaatst van max 50 woorden.

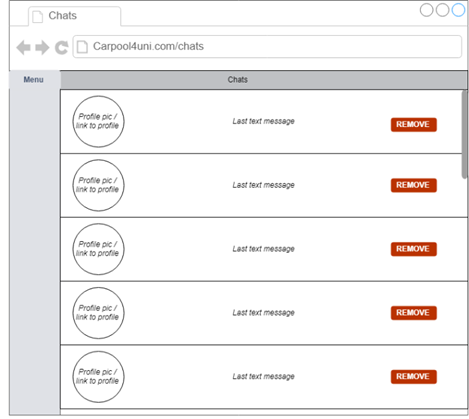
### Balance



Figuur 8 Balance

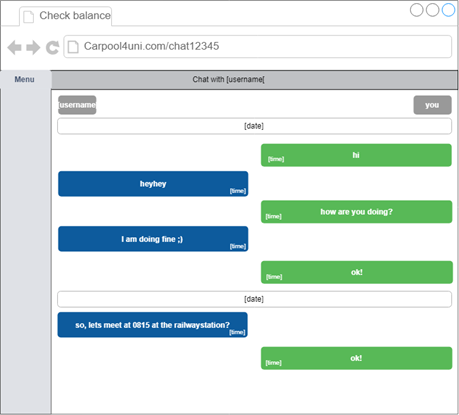
Op het saldo bekijken scherm in figuur 8 staat in een grote bubbel het saldo van de gebruiker. Daaronder twee velden met getallen en twee knoppen. De ene knop laat de gebruiker geld toevoegen aan het saldo, de andere laat de gebruiker geld naar zijn bankrekening storten. De twee velden bepalen hoeveel geld er toegevoegd, afgeschreven wordt.

### Chats



Figuur 9 Chats overview

De chats bestaat uit een overzichtsscherm met alle actieve chats (figuur 9) en een chatpagina tussen 2 gebruikers (figuur 10). Het overzichtsscherm bestaat uit een scrollbare lijst met actieve chats.



Figuur 10 Chat

Bij elke chat kan de gebruiker het laatste chatbericht lezen. De profielfoto van de andere persoon bekijken en naar het profiel van die persoon gaan als hij er op klinkt. Verder zit er een remove knop bij om de chat te verwijderen. Als de gebruiker op het item drukt gaat hij naar de chat pagina.

### Travel overview



Figuur 11 Travel overview

In figuur 11, de pagina travel overview is te zien welke aanvragen er zijn gedaan, zowel aanvragen om mee te rijden Op de pagina travel overview is te zien welke aanvragen er zijn gedaan, zowel aanvragen om mee te rijden (drive) als aanvragen van iemand anders om mee te rijden (lift), welke ritten er actief zijn (geaccepteerd zijn en nog moeten plaatsvinden) en welke ritten er zijn geweest.

Een lift verzoek kan hier geaccepteerd of afgewezen worden. Alle ritten zijn gesorteerd op data (nieuwste bovenaan). De kolommen kunnen worden uitgevouwen en ingeklapt. In elke strook zijn een aantal keywords te vinden. Sommigen daarvan zijn een link:

* De profielfoto is een link naar het profiel van de gebruiker. Op het profiel van de gebruiker zijn de voorkeuren te zien van hem/haar in de auto.
* De ‘view travel’ is een link naar een pop-up met informatie over de rit. In de pop-up staat informatie, zoals: tijd, datum, adres, naam, prijs. Door te klikken op ‘OK’ verdwijnt de pop-up weer. Door te klikken op Cancel ride, wat alleen 48 uur van tevoren kan, verschijnt er een volgende pop-up met de vraag of je het zeker weet.
* Verder kan een rit die bezig is worden bevestigd/beoordeeld door te klikken bij active travels op ‘confirm’. Er verschijnt dan een pop-up met hoe de rit was. Door te klikken op OK, wordt de rit afgesloten. Door te klikken op Not OK, zal er een vervolg pop-up verschijnen. Hierin wordt nogmaals gevraagd of de rit okay was. Zo niet, dan wordt er op ‘complain’ geklikt. Daar kan de gebruiker aangeven wat de klacht is. Als er op ‘Else’ geklikt wordt kan hij/zij een reden invoeren van max 30 woorden. (zie pop-ups: Confirmation, Are you sure?, What happened?)

### Notifications



Figuur 12 Notificaties

Een uur van tevoren verschijnt er een pop-up als reminder dat je een rit hebt. In de pop-up bevindt zich een link om de pop-up te sluiten en een link om de rit te zien. Dat laatste verwijst de gebruiker door naar de pagina ‘travel overview’. Zie figuur 12.

Een lift request is een pop-up die midden in het scherm verschijnt wanneer er een verzoek wordt gedaan om mee te rijden op een bepaalde datum. Er kan gekozen worden om de pop-up te sluiten, in dat geval verschijnt er een ‘1tje’ bij de travel overview, aangezien er 1 request nog niet is beantwoord. Er kan ook gekozen worden voor ‘view request’, in dat geval wordt de persoon doorverwezen naar de pagina ‘travel overview’. Zie figuur 12.

# Architecture design

Het interaction ontwerp is klaar. Er is nu duidelijk hoe de gebruikersinterface eruit komt te zien en wat de gebruikers moeten doen om bepaalde acties te verrichten. In dit hoofdstuk wordt het ontwerp van de website zelf besproken. Dit wordt gedaan door drie verschillende diagrammen te weergeven:

1. Swimlane diagram, deze laat zien welke acties allemaal gedaan moeten worden, hoe deze met elkaar verbonden zijn en welke actor ze uitvoert.
2. Sequence diagram, hierin wordt in een tijdschema weergegeven hoe welke stappen het systeem doorloopt.
3. Class diagram, deze laat zien welke componenten (klassen) in het systeem zitten, welke functies/methoden ze hebben en hoe ze met elkaar verbonden zijn.

Deze diagrammen kunnen worden uitgewerkt in code om daadwerkelijk een website te maken. Dit wordt in dit project niet gedaan, omdat het te veel tijd kost.

## Swimlane diagram

De swimlane diagram laat de acties van alle actoren zien. Omdat het een erg uitgebreide diagram is zal hij per use case toegelicht worden. De volledige diagram is apart meegeleverd en ook te vinden in bijlage 4.

### C:\Users\floris kool\Downloads\Swimlane diagram carpool4uni (1).jpgInloggen en account aanmaken

Figuur 13 Simlane: inloggen & account aanmaken

In figuur 13 zijn de use cases inloggen en account aanmaken weergegeven. De diagram laat zien dat als de website geopend is de gebruiker eerst moet inloggen met correcte gegevens voor hij verder kan gaan.

### C:\Users\floris kool\Downloads\Swimlane diagram carpool4uni (2).jpgHoofdscherm weergeven en nieuwe rit invoeren

Figuur 14 Swimlane diagram: hoofdscherm en nieuwe rit

In figuur 14 is te zien dat het systeem eerst het de lijst met beschikbare ritten opvraagt, daarna het hoofdscherm weergeeft. Daarnaast is de uitwerking van een nieuwe rit invoeren zichtbaar.

### C:\Users\floris kool\Downloads\Swimlane diagram carpool4uni (3).jpgRit bekijken

Figuur 15 Swimlane diagram: rit bekijken

In figuur 15 is te zien wat er gebeurt als de gebruiker een rit wilt bekijken in het hoofdscherm. Hij kan het profiel drukken van de chauffeur om deze te bekijken. Dan heeft hij de optie om te chatten of terug te gaan naar het hoofdscherm. Voordat er gechat kan worden wordt gekeken of er daarvoor toestemming is.

### C:\Users\floris kool\Downloads\Swimlane diagram carpool4uni (4).jpgChatten en menu

Figuur 16 Swimlane diagram: chatten en menu

In het bovenstaande figuur is het menu te zien. In het menu heeft de gebruiker veel keuzes. Eén van die keuzes is chatten. Hier kan de gebruiker een chat selecteren en chatten met een andere gebruiker.

### C:\Users\floris kool\Downloads\Swimlane diagram carpool4uni (6).jpgSaldo opwaarderen

Figuur 17 Swimlane diagram: Saldo opwaarderen

De diagram voor het opwaarderen van het saldo is te vinden in figuur 17. Dit is net zoals chatten een keuze in het menu.

### C:\Users\floris kool\Downloads\Swimlane diagram carpool4uni (7).jpgProfiel aanpassen

Figuur 18 Swimlane diagram profiel aanpassen

Figuur 18 laat zien hoe de gebruiker het profiel kan aanpassen en hoe dit door de website verwerkt wordt.

### C:\Users\floris kool\Downloads\Swimlane diagram carpool4uni (8).jpgNotificaties

Figuur 19 Swimlane diagram: Notificaties

In het bovenstaande figuur, figuur 19 zijn alle notificaties te vinden. Het gaat om twee verschillende soorten: Een rit aanvraag en een bevestiging van een rit. Voor beide scenario’s zijn de mogelijkheden zichtbaar.

## Sequence diagram

In het sequence diagram worden de interacties van use cases tussen de verschillende onderdelen van het systeem gevisualiseerd. De onderdelen van het systeem zijn de actor, de applicatie en de server. De nadruk ligt bij het sequence diagram op een chronologische volgorde van interacties.

De eerste use case in het diagram is de use case van het account aanmaken. Deze werkt als volgt, de gebruiker klikt op de knop account aanmaken. De applicatie stuurt dan een opdracht om gegevens in te vullen. Deze vult de gebruiker in en dan wordt het weer terug naar de applicatie gestuurd. Deze zendt het dan door naar de server om het op te slaan, waarna een bevestiging wordt verstuurd naar de gebruiker dat het account is aangemaakt.

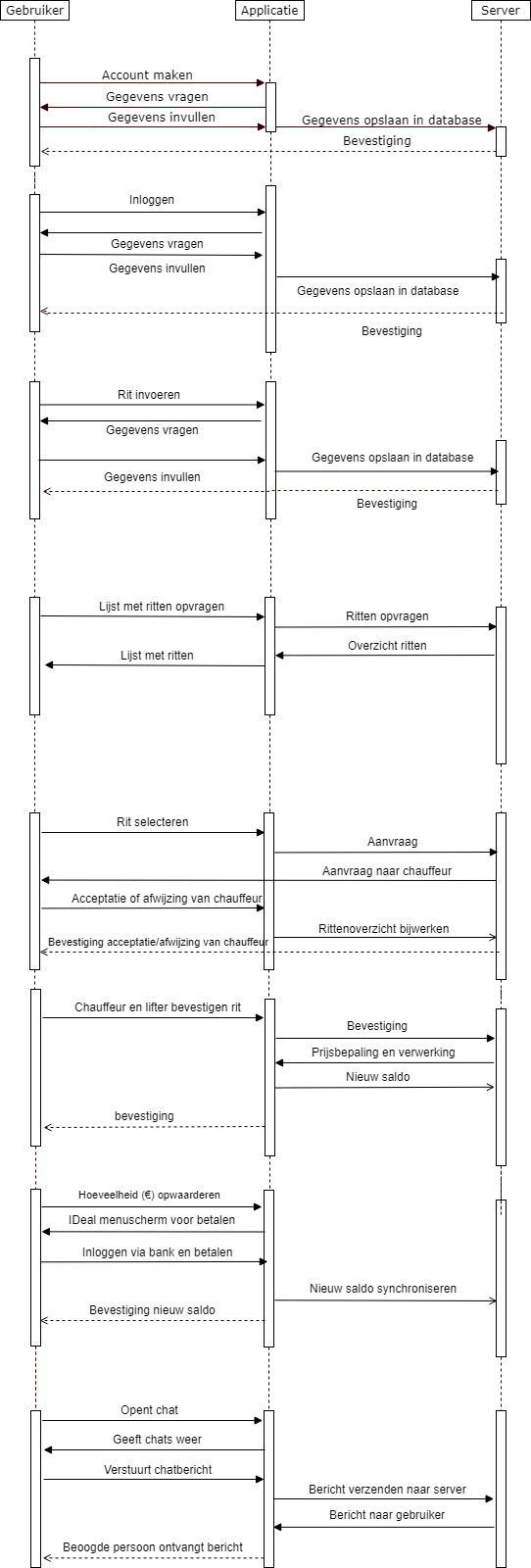
De tweede use case is die van het inloggen, dit werkt bijna hetzelfde als bij het account aanmaken. In dit geval wordt er om andere gegevens gevraagd maar verder is het proces hetzelfde. Ditzelfde geldt voor de derde use case.

De use case ‘ritten opvragen/aanvragen’ ziet er wel anders uit. Hier klikt de gebruiker op de knop lijst met ritten opvragen, dit gaat naar het systeem die vervolgens dit opvraagt uit de database. De database stuurt dan het overzicht van de ritten via de database terug naar de gebruiker.

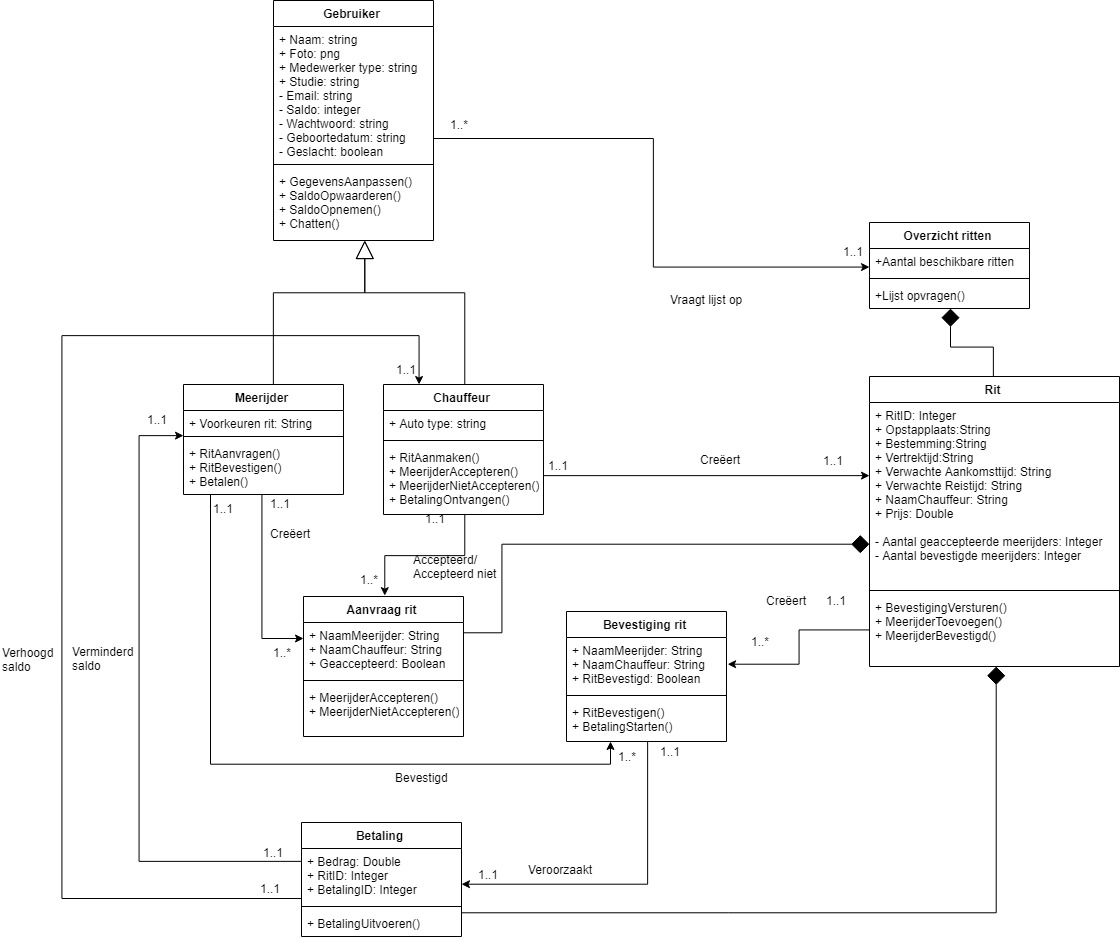
In deze use case kan de gebruiker een rit aanvragen, dit kan door op de knop rit aanvragen te klikken. De aanvraag gaat dan via het systeem naar de database, deze stuurt dit door naar desbetreffende chauffeur. De chauffeur klikt dan vervolgens aan of hij de rit accepteert of afwijst. Dat wordt naar het systeem gestuurd, die past daarna het rittenoverzicht aan en dit wordt in de database gezet. De database geeft daarna een bevestiging van de acceptatie of afwijzing van de chauffeur naar de gebruiker.

Dan komt de use case ‘saldo opwaarderen’. De gebruiker klikt een bedrag aan dat hij wil opwaarderen, dit gaat naar het systeem en het systeem stuurt de gebruiker dan naar het IDeal menuscherm waar hij kan betalen. De gebruiker voert dan zijn gegevens in en betaalt. Het systeem stuurt dit door naar de database om vervolgens het saldo te synchroniseren. Tot slot komt er een bevestiging van het nieuwe saldo naar de gebruiker.

De laatste use case is die van het chatten, de gebruiker klikt op de knop chatten, dit gaat naar het systeem en die zorgt er daarna voor dat de gebruiker de chats kan zien. Als de gebruiker een chatbericht wilt versturen klikt hij op de knop verzenden, het bericht gaat dan via het systeem naar de database. Deze stuurt het bericht vervolgens weer door naar de beoogde ontvanger.



## Class diagram

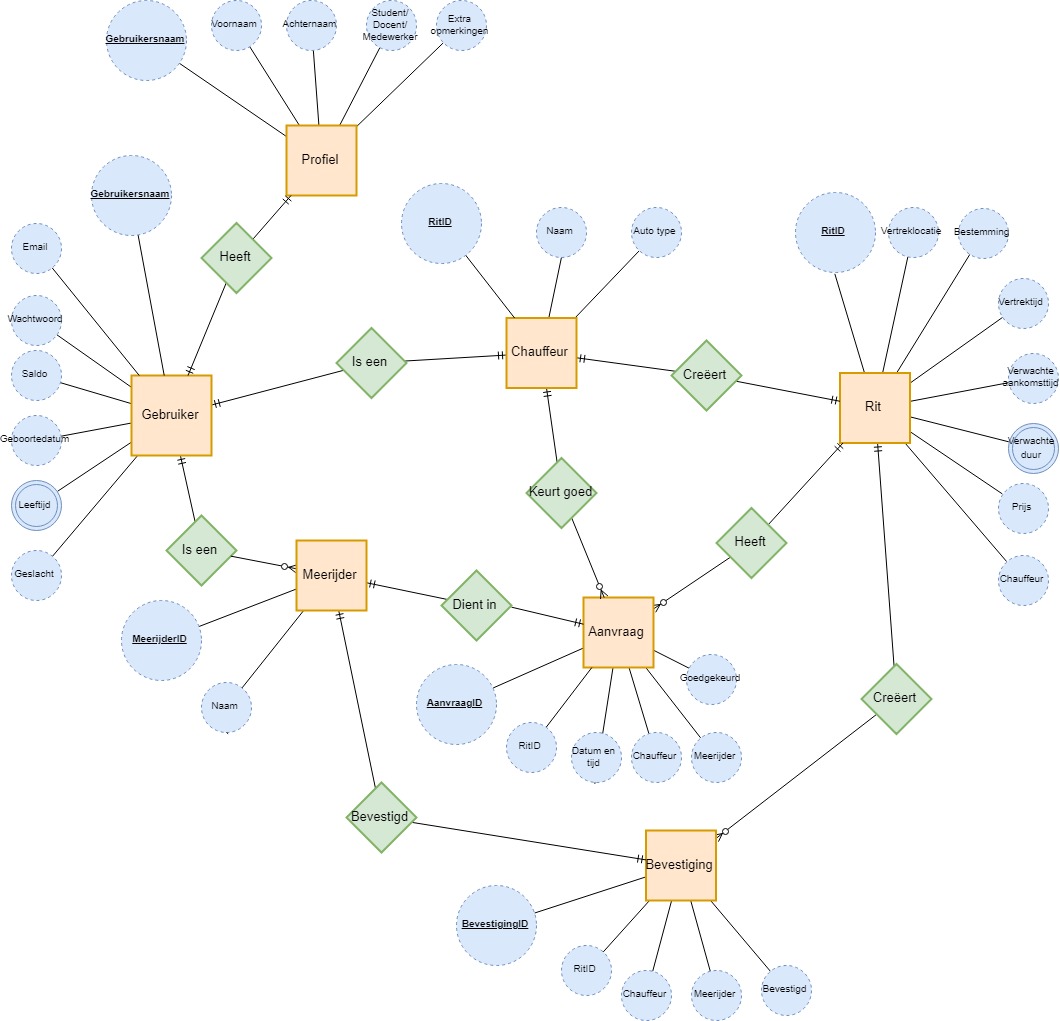
De class diagram is het belangrijkste ontwerpdocument voor de website. Het geeft aan welke variabelen het programma heeft, welke functies en welke klassen. De class diagram voor het carpool4uni project staat in figuur 20 en tussen de bijlagen.

Figuur 20 Class diagram

# Database design

Het ontwerp dat in hoofdstuk 5 heeft gemaakt vereist dat er veel data kan worden opgeslagen. Hiervoor moet een database worden ontworpen. Dat wordt in dit hoofdstuk gedaan door eerst een entity relationship diagram op te stellen die uitlegt welke componenten er in het systeem zitten en hoe deze met elkaar verbonden zijn. Vervolgens wordt er een database ontwerp gemaakt. Om van dit ontwerp gebruik te maken moeten er SQL queries opgesteld worden die data opvragen en toevoegen.

## Entity Relationship Diagram

Een entity relationship diagram (ERD) is een diagram dat aangeeft welke componenten er aanwezig zijn, wat hun attributen zijn en hoe de verschillende componenten data met elkaar uitwisselen. Hieronder in figuur x en in bijlage 5 is de ERD te zien.

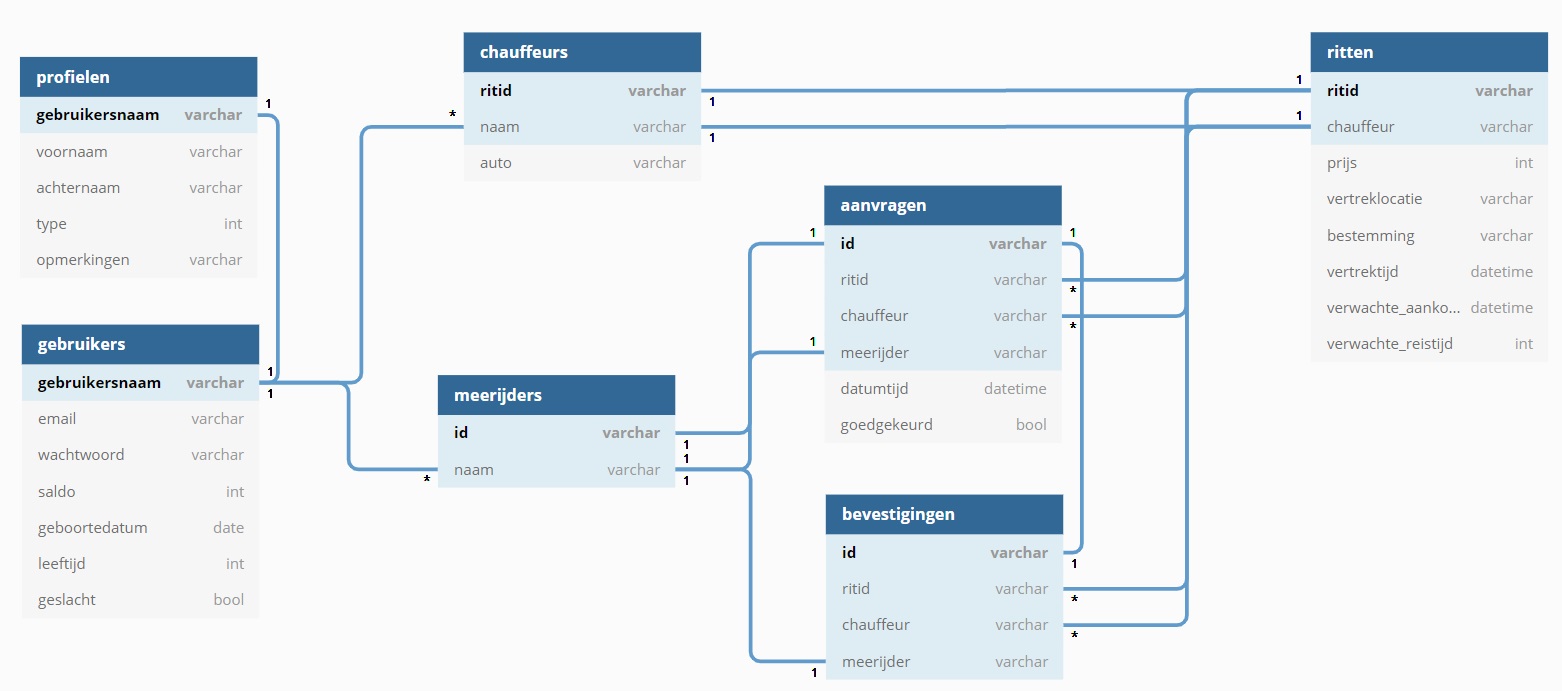
Figuur 21 ERD

De gebruiker is een belangrijke entity. Het bevat alle informatie over de gebruiker zoals naam, email, wachtwoord, etc. De zichtbare eigenschappen van een gebruiker worden opgeslagen op de entiteit profiel. Er zijn twee verschillende soorten gebruikers: Meerijder en Chauffeur. Een gebruiker kan beide zijn. Bijvoorbeeld voor de ene rit een chauffeur en voor twee andere ritten een meerijder.

Een chauffeur kan een rit aanmaken. Als er een rit wordt aangemaakt krijgt deze een uniek ID. Deze rit is gekoppeld aan de chauffeur en heeft veel andere informatie zoals vertrektijd, locatie, prijs, etc. Voor deze rit kunnen meerijders aanvragen indienen. Als de meerijder dat doet ontstaat er een aanvraagID. Dit is een uniek ID dat de naam van de meerijder en het ID van de rit bevat. Een rit kan meerdere aanvragen krijgen. Op de aanvraag entity wordt aangegeven of de chauffeur deze al geaccepteerd heeft.

Als het een uur voor de vertrektijd is zal de rit zelf voor alle geaccepteerde aanvragen een bevestiging sturen. Deze bevestiging heeft dezelfde ID als de aanvraag. Hierin wordt opgeslagen of de meerijder en chauffeur de rit hebben bevestigt.

## Database diagram

De database diagram is te vinden in bijlage 6 en in het onderstaande figuur x.

Figuur 22 Database diagram

De database diagram bestaat uit de 7 entities uit de ERD. De gebruiker bevat alle informatie over de gebruiker en wordt uitgebreid met een profiel. Elke gebruiker heeft 1 profiel en elk profiel 1 gebruiker. De naam van de gebruiker wordt gedeeld met de chauffeurs en meerijders tabellen.

De ritten en chauffeurstabellen hebben dezelfde primary key, namelijk het RitID. De chauffeur bevat wat meer informatie over de chauffeur en de ritten alle informatie over de rit zoals prijs en vertrektijd. Voor elke rit is er maar 1 chauffeur en elke chauffeur in de tabel is er maar 1 rit.

De meerijders, aanvragen en bevestigingen hebben ook dezelfde primary key. Deze key is gekoppeld aan de rit en meerijder. Elke meerijder heeft 1 aanvraag en 1 bevestiging. Een rit heeft wel meerder aanvragen, bevestigingen en meerijders.

## SQL queries

De SQL queries geven weer hoe de benodigde informatie voor een interactie opgehaald wordt uit de database. De interacties die er plaats kunnen vinden worden, zoals eerder al vermeld, use cases genoemd. Totaal zijn er negen use cases waarvoor SQL queries zijn opgesteld.

1. **Account aanmaken. Elke gebruiker moet een account kunnen aanmaken om gebruik te maken van de website.**

INSERT INTO Gebruikers (gebruikersid, gebruikersnaam, wachtwoord, email, geslacht, leeftijd, functie, roken, autoformaat, autotype, bankgegevens, opmerkingen)  
VALUES (xxx123, xxx, xxx, [xxx@xxx.com](mailto:xxx@xxx.com), m/v, dd/mm/yyyy, student/werknemer, yes/no, klein/middel/groot, xxx, xxx, xxx)

Dit is een tabel met alle gebruikers. Er is tevens een functie ‘Deze\_gebruiker’ waarmee de informatie van enkel de ingelogde gebruiker wordt opgehaald.

1. **Profiel aanmaken. Bij het aanmaken van een account er automatisch een profiel aangemaakt waar een deel van de accountgegevens wordt weergegeven.**

INSERT INTO Profiel\_gebruiker (gebruikersid, geslacht, leeftijd, functie, autoformaat, autotype, opmerkingen)  
VALUES ( SELECT gebruikersid, geslacht, leeftijd, functie, autoformaat, autotype, opmerkingen FROM Deze\_Gebruiker )

Met deze code wordt er een nieuwe rij in de tabel ‘gebruikers’ aangemaakt met als kolommen de karakteristieken tussen haakjes. Deze hebben de opeenvolgende waardes die achter ‘VALUES’ staat.

1. **Inloggen. De gebruiker moet ingelogd zijn, omdat alle acties binnen de website verbonden zijn een gebruikersaccount.**

SELECT Gebruikers.EXISTS   
FROM Gebruikers  
WHERE username = xxx@hotmail.com AND password = yyy

Met deze code wordt in de database gezocht naar een account met een bepaalde username en een password. Indien deze bestaat, dan wordt gereturned dat er een account is met deze inloggegevens.

1. **Rit invoeren. De chauffeur kan een nieuwe rit invoeren in het system. Deze wordt zichtbaar voor alle ingelogde gebruikers.**

INSERT INTO ritten (gebruikersid, ritid, chauffeur, prijs, vertreklocatie, bestemming, vertrektijd, verwachte\_aankomsttijd, verwachte\_reistijd, goedgekeurd)  
VALUES (SELECT gebruikersid FROM deze\_gebruiker, xxx, gebruikersnaam, €€€, [locatie], [bestemming], dd/mm/yyyy hh:mm, dd/mm/yyyy hh:mm, dd/mm/yyyy hh:mm, hh:mm, no)

Zoals omschreven bij de eerste use case in deze paragraaf, geeft de functie ‘INSERT INTO’ aan dat er een nieuwe rij met de karakteristieken tussen haakjes als kolommen wordt gecreëerd.

1. **Rit aanvragen. Een meerijder kan een verzoek tot aanvragen indienen bij een rit.**

INSERT INTO aanvragen\_tabel (chauffeurs.gebruikersid, ritid, meerijders.gebruikersid, datum, tijd, goedgekeurd)  
VALUES ( SELECT chauffeurs.gebruikersid, ritid, meerijders.gebruikersid, datum, tijd, goedgekeurd FROM ritten, chauffeurs, meerijders WHERE ritten.gebruikersid = chauffeurs.gebruikersid AND meerijders.gebruikersid = deze\_gebruiker.gebruikersid)

Bij deze querie wordt er een tabel opgevraagd met de karakteristieken ‘ritid’, ‘chauffeur’, ‘meerijder’, ‘datumtijd’ en ‘goedgekeurd’

1. **Meerijder accepteren. De chauffeur moet de meerijder accepteren voordat een meerijder mee mag rijden.**

UPDATE aanvragen\_tabel  
SET goedgekeurd TRUE  
WHERE aanvragen.id = meerijders.id AND meerijder = chauffeur\_bevestigd = true

Wanneer de chauffeur de aanvraag bevestigt, wordt de ‘goedgekeurd’ kolom van de juiste aanvraag (op basis van het id van de meerijder) goedgekeurd of afgewezen.

1. **Rit bevestigen. De meerijder en chauffeur moeten de rit bevestigen een uur voor de rit. Zo wordt voorkomen dat er mensen niet komen opdagen.**

SELECT \*  
FROM bevestigingen  
WHERE bevestigingen.id = meerijders.id

1. **Betaling. De betaling verloopt automatisch na het bevestigen.**

SELECT prijs  
FROM ritten  
WHERE bevestigd\_meerijder = true AND bevestigd\_chauffeur = true AND bevestigingen.ritid = ritten.ritid

Wanneer de pop-up bevestiging wordt bevestigd, zal de prijs worden opgehaald uit de ritten aan de hand van het ritid.

1. **Tegoed opwaarderen. Betalingen gaan via een saldo, de gebruiker moet dit saldo kunnen opwaarderen en opnemen.**

UPDATE gebruikers  
SET saldo saldo + opgewaardeerd  
WHERE gebruikersnaam = [gebruikersnaam]

Zodra het tegoed wordt opgewaardeerd, moet deze worden bijgewerkt in de database. Daarvoor zal van het juiste gebruikersprofiel het saldo moeten worden opgehaald en worden bijgewerkt.

# Testing plan

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

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()

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()

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()

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

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 #2

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 #1

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 #4

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 #8

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 #15

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 tests

Zodra de applicatie in werking treed zullen er vooral in het begin relatief veel opties zijn tot uitbreiding van de applicatie. Die opties verspreiden zich over een vlak dat niet alleen een technisch aspect heeft, maar ook bijvoorbeeld een sociaal en financieel aspect heeft. Dat houdt dus niet alleen technische bugs en efficiëntietoepassingen in, maar ook strategische financiële aspecten en klantbindende aspecten in. Een manier om de applicatie op dit brede vlak te verbeteren, is door het houden van acceptance tests. Deze richten zich op de van de applicatie en doelen op het verkrijgen van informatie die ertoe leiden dat de ontwikkelaars de applicatie effectiever en efficiënter kunnen laten runnen. Dat doen we op de volgende manieren:

1. Het gebruiken van een algoritme waarmee informatie verkregen kan worden over de locaties waar de applicatie relatief meer wordt gebruikt. Daardoor kan strategisch ingezet worden op promotie op drukke gebieden, die dus effectiever zijn. Het doel hiervan is het uitbreiden van het marktaandeel van de applicatie.
2. Het achterhalen van de leeftijd en studiejaren waarin de studenten zich bevinden. Wederom kan dit gebruikt worden om strategisch in te zetten op promotie bij bepaalde categorieën studenten.
3. Een formulier en/of contactgegevens waarmee suggesties kunnen worden verstuurd naar carpool4uni. Deze heeft als doel het uitbreiden van de technische aspecten van de applicatie.
4. Bij het aanmelden een optie toevoegen met de vraag hoe zij bij carpool4uni terecht zijn gekomen. Hieruit blijkt welke promotiemiddelen succesvol zijn.
5. Een rating over de applicatie waaruit blijkt of het resultaat bevalt en/of niet. Wanneer deze niet bevalt moet er dus harder gewerkt worden aan het verkrijgen van informatie waarom de applicatie niet bevalt.

# Bijlagen

## Bijlage 1 Functionele requirements

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 1 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | De gebruikers moet inloggen voordat ze toegang hebben tot de rest van de site. | | | | |
| **Rationale:** | Vrijwel alle acties op de site zijn gekoppeld aan het account van de gebruiker. De gebruiker moet | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | Wanneer de applicatie gestart wordt | | | | |
| **Customer Satisfaction:** | 4 | | **Customer Dissatisfaction:** | 4 | |
| **Priority:** | 4 | | **Conflicts:** | ... | |
| **Supporting Materials:** | Notities gesprek 10/09/2018 | | | | |
| **History:** | Genoteerd 10/09/2018 | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 2 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | Chauffeurs en meelifters moeten een profiel aanmaken en kunnen daar belangrijke, voor andere gebruikers openbare, gegevens kwijt.   * Geslacht \*man/vrouw) * Leeftijd * Foto * Medewerker/Student/Docent * Studie (evt) * Roken (niet) gewaardeerd/toegestaan * Autotype + klein/middel/groot formaat (evt) * Extra opmerking(en) (evt)   Daarbij mieten ze ook een aantal privé gegevens invullen die belangrijk zijn voor hun account:   * Bankgegevens * E-mail (waarmee je ook inlogt) * Een wachtwoord | | | | |
| **Rationale:** | Meerijders en chauffeurs weten wat zij kunnen verwachten en kunnen selecteren op elkaar. | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | Meerijders kunnen een hun keuze baseren op een selectie van voorkeuren, bijvoorbeeld carpoolers filteren op geslacht, roken, functie of autotype. Chauffeurs kunnen hun meerijders accepteren of afwijzen op basis van hun profiel, zodat het voor hun prettig blijft om te carpoolen | | | | |
| **Customer Satisfaction:** | 4 | | **Customer Dissatisfaction:** | 5 | |
| **Priority:** | 5 | | **Conflicts:** |  | |
| **Supporting Materials:** | notities interview 10-9 | | | | |
| **History:** | genoteerd op 11-9 | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 3 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | Ieder profiel heeft een wallet met een tegoed dat kan worden opgewaardeerd of uitgecasht (vanaf 1€) via paypal, IDeal of een bankoverdracht | | | | |
| **Rationale:** | Hiermee kan betaald worden en betalingen worden ontvangen. | | | | |
| **Originator:** | ... | | | | |
| **Fit Criterion:** | Dit moet gebeuren wanneer een gebruiker een account aanmaakt. | | | | |
| **Customer Satisfaction:** | 4 | | **Customer Dissatisfaction:** | 4 | |
| **Priority:** | 4 | | **Conflicts:** | ... | |
| **Supporting Materials:** | Notities gesprek 10/09/2018 | | | | |
| **History:** | Genoteerd 10/09/2018 | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 4 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | Een chauffeur kan een rit invoeren in de database. Daarbij hoort een op- en uitstapplaats en aantal mogelijke meerijders. | | | | |
| **Rationale:** | De chauffeur moet kunnen aangeven dat hij een rit van/naar de universiteit gaat maken. | | | | |
| **Originator:** | Groep 13 | | | | |
| **Fit Criterion:** | Als een ingelogde chauffeur een nieuwe rit kan toevoegen aan de database. | | | | |
| **Customer Satisfaction:** | 4 | | **Customer Dissatisfaction:** | 5 | |
| **Priority:** | 5 | | **Conflicts:** | ... | |
| **Supporting Materials:** | Notities gesprek 10/09/2018 | | | | |
| **History:** | Genoteerd 10/09/2018 | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 5 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | De prijs van een rit wordt bepaald door een vast kilometertarief, die gebaseerd is op het type auto van de chauffeur. Bovenop de prijs komt een percentage van het bedrag dat wordt betaald door de meerijder en dat gaat naar carpool4uni | | | | |
| **Rationale:** | De chauffeur kan zo niet te weinig vergoeding krijgen voor de rit. | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | Dit gebeurt wanneer de chauffeur een rit aanbiedt | | | | |
| **Customer Satisfaction:** | 4 | | **Customer Dissatisfaction:** | 5 | |
| **Priority:** | 5 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 6 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | Naderhand moeten chauffeur en meerijder de rit bevestigen in de applicatie. Zodra dit gedaan is wordt het geld automatisch afgeschreven van het tegoed van de meerijder en toegevoegd aan het tegoed van de chauffeur | | | | |
| **Rationale:** | chauffeurs behoren een compensatie te krijgen voor hun gemaakte kosten | | | | |
| **Originator:** | Ontwikkelaars | | | | |
| **Fit Criterion:** | De betaling verloopt soepel, zonder storingen en zonder fraude. | | | | |
| **Customer Satisfaction:** | . | | **Customer Dissatisfaction:** | . | |
| **Priority:** | 1 | | **Conflicts:** | ... | |
| **Supporting Materials:** | betalingssysteem | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 7 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | De passagier moet een rit kunnen selecteren uit een lijst van alle ritten naar die universiteit en aan de hand van zijn locatie. De chauffeur kan de passagier accepteren of afwijzen. Een meerijder moet een overzicht kunnen zien van beschikbare ritten. | | | | |
| **Rationale:** | Functie rit-verificatie | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | Beschikbare ritten zijn zichtbaar en selecteerbaar. Chauffeur ontvangt aanvraag. Chauffeur krijgt de keuze accepteren of afwijzen. Passagier krijgt de mogelijkheid de rit te bevestigen of er vanaf te zien. | | | | |
| **Customer Satisfaction:** | 4 | | **Customer Dissatisfaction:** | 5 | |
| **Priority:** | 5 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 8 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | Een meerijder kan 1 of meerdere aanvragen indienen voor een rit. | | | | |
| **Rationale:** | De meerijder kan zo een rit selecteren. Omdat een aanvraag ook afgewezen kan worden kan een meerijder voor meerdere ritten een aanvraag indienen. Zodra de meerijder ergens wordt geaccepteerd, vervallen automatisch alle andere aanvragen. | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | ... | | | | |
| **Customer Satisfaction:** | 3 | | **Customer Dissatisfaction:** | 3 | |
| **Priority:** | 3 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 9 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | Een chauffeur krijgt een mailtje/sms/app als er een aanvraag wordt ingediend voor zijn rit. | | | | |
| **Rationale:** | Zo weet de chauffeur dat er iemand mee wilt rijden en kan hij snel reageren. | | | | |
| **Originator:** | Ontwikkelaars | | | | |
| **Fit Criterion:** | Als er een push notification wordt gegeven als er een aanvraag wordt ingediend voor de rit van een chauffeur. | | | | |
| **Customer Satisfaction:** | 3 | | **Customer Dissatisfaction:** | 4 | |
| **Priority:** | 2 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 10 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | Een uur voor de rit krijgen zowel de chauffeur als meerijder een melding/herinnering voor hun rit. | | | | |
| **Rationale:** | Er worden geen lifters vergeten (aangezien een rit ook ver van tevoren afgesproken kan worden). | | | | |
| **Originator:** | Ontwikkelaars | | | | |
| **Fit Criterion:** | Als er zo min mogelijk lifters vergeten worden | | | | |
| **Customer Satisfaction:** | . | | **Customer Dissatisfaction:** | . | |
| **Priority:** | . | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 11 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | Nadat een passagier een rit heeft geselecteerd en de chauffeur de aanvraag heeft geaccepteerd, wordt de chatfunctie actief. De chatfunctie geeft een open communicatie tussen passagier en chauffeur. | | | | |
| **Rationale:** | Beveiliging van contactgegevens totdat de chauffeur toestemming heeft verleent. Chatfunctie om afspraken over de rit te maken. | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | Er kunnen berichten verstuurt worden. De berichten die verstuurt zijn worden ontvangen door de andere partij. Chatfunctie wordt pas actief na acceptatie van de chauffeur. | | | | |
| **Customer Satisfaction:** | 4 | | **Customer Dissatisfaction:** | 2 | |
| **Priority:** | 3 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 12 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | App-gebruikers moeten een beoordeling (rating en plek voor recensies) kunnen krijgen die te zien is op zijn/haar profiel en zich uitsluitend baseert op betrouwbaarheid. Daarbij helpt een optioneel persoonlijk stukje om een indruk te krijgen over de chauffeur of meerijder. | | | | |
| **Rationale:** | Hierdoor kunnen gebruikers andere gebruikers beoordelen op de betrouwbaarheid van de rit (komt diegene wel opdagen?). | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | Gebruikers kunnen mensen die relatief vaak te laat komen of niet komen opdagen herkennen en zij kunnen een inschatting maken over de persoon met behulp van het korte persoonlijke stukje. | | | | |
| **Customer Satisfaction:** | 4 | | **Customer Dissatisfaction:** | 5 | |
| **Priority:** | 4 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 13 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | De lay-out van de webapplicatie is minimalistisch strak met wit en groene kleuren. | | | | |
| **Rationale:** | Uitstraling interface | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | Minimalistisch strak Groene kleur Witte kleur | | | | |
| **Customer Satisfaction:** | 3 | | **Customer Dissatisfaction:** | 3 | |
| **Priority:** | 3 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 14 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | De webapplicatie moet een trackfunctie bevatten waarmee de chauffeur te tracken is. | | | | |
| **Rationale:** | Trackfunctie voor passagier. | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | Zodra de chauffeur vertrekt moet de passagier de locatie van de chauffeur kunnen volgen via de webapplicatie. | | | | |
| **Customer Satisfaction:** | 4 | | **Customer Dissatisfaction:** | 2 | |
| **Priority:** | 2 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 15 | **Requirement type:** | F | **Event/Use Case #:** |  |
| **Description:** | De webapplicatie moet een historie functie hebben waarmee gereden ritten opgeslagen worden en bekeken kunnen worden door de gebruiker. | | | | |
| **Rationale:** | Historie functie voor gereden ritten. | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | Gereden ritten worden opgeslagen. Gereden ritten kunnen ingezien worden. | | | | |
| **Customer Satisfaction:** | 3 | | **Customer Dissatisfaction:** | 2 | |
| **Priority:** | 2 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

## Bijlage 2 Niet functionele requirements

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 16 | **Requirement type:** | NF | **Event/Use Case #:** |  |
| **Description:** | De eerste vier maanden vindt een maandelijkse reflectie plaats met cliënt en/of gebruikers. Afhankelijk van het succes wordt er nog gekeken naar de frequentie van reflecteren na deze 4 maanden. | | | | |
| **Rationale:** | Betrekking op onderhouds - en kwaliteitsplan. Naarmate er meer reflecties zijn geweest, zal dit steeds minder vaak hoeven. | | | | |
| **Originator:** | Cliënt & Ontwikkelaars | | | | |
| **Fit Criterion:** | Positieve reflecties | | | | |
| **Customer Satisfaction:** | 3 | | **Customer Dissatisfaction:** | 3 | |
| **Priority:** | 3 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement #:** | 17 | **Requirement type:** | NF | **Event/Use Case #:** |  |
| **Description:** | Het moet een webapplicatie zijn. | | | | |
| **Rationale:** | Deze eis legt grenzen aan de compatibiliteit van de softwareoplossing. | | | | |
| **Originator:** | Cliënt | | | | |
| **Fit Criterion:** | De functies werken op een website. | | | | |
| **Customer Satisfaction:** | 1 | | **Customer Dissatisfaction:** | 5 | |
| **Priority:** | 5 | | **Conflicts:** | ... | |
| **Supporting Materials:** | ... | | | | |
| **History:** | ... | | | | |

## Bijlage 3 Use cases

|  |  |
| --- | --- |
| **Name and identifier** | Account aanmaken, AccAanm |
| **Objective** | De gebruiker moet een account kunnen aanmaken. Bij het aanmaken voert hij informatie over zichzelf in zoals Naam, Geboortedatum, Type auto (indien van toepassing) etc. Nadat een account is aangemaakt moeten de gegevens bewaard worden, zodat de gebruiker kan inloggen. |
| **Priority** | 1 |
| **Source** | Overleg 11/9 |
| **Actors** | Gebruiker, zowel meerijder als chauffeur |
| **Basic flow** | De gebruiker drukt op de knop account aanmaken. Vult vervolgens alle gevraagde informatie in en drukt op de knop bevestigen. |
| **Alternative flow 1** | De gebruiker drukt op de knop account aanmaken. Vult vervolgens een aantal gegevens in en drukt op de knop bevestigen. Het systeem moet dan beoordelen of hij alle verplichte velden heeft ingevuld. Zo niet krijgt de gebruiker hier bericht van en kan hij opnieuw zijn gegevens invullen. |
| **Alternative flow 2** | De gebruiker drukt op de knop account aanmaken. Vult vervolgens een aantal gegevens in en drukt op de knop annuleren. Het systeem zal dan de ingevulde gegevens moeten verwijderen en de gebruiker terug moeten sturen naar het vorige scherm. |
| **Exceptions** | De gebruiker drukt op de knop account aanmaken, vult een aantal velden in en sluit de app af. De app moet dan niet de gegevens bewaren, de gebruiker zal opnieuw gegevens moeten invullen. |
| **Includes** | ... |
| **Extends** | ... |
| **Generalizes** | ... |
| **Preconditions** | Als de app is opgestart en de gebruiker niet is ingelogd kan de gebruiker kiezen of hij een account wilt aanmaken of wilt inloggen. Daarnaast zal de knop account aanmaken beschikbaar zijn in het menu. |
| **Postconditions** | Na het aanmaken van een account wordt de gebruiker uitgelogd als hij al is ingelogd en gaat de gebruiker naar het inlogscherm. |
| **Notes/Issues** | ... |

|  |  |
| --- | --- |
| **Name and identifier** | Inloggen, Inl |
| **Objective** | De gebruiker kan hier zijn gebruikersnaam en wachtwoord invoeren. Nadat de gebruiker is ingelogd kan hij gebruik maken van de functionaliteit van de app. |
| **Priority** | 1 |
| **Source** | Overleg 11/9 |
| **Actors** | Gebruiker, zowel meerijder als chauffeur. |
| **Basic flow** | De gebruiker vult zijn gebruikersnaam en wachtwoord in en drukt op de knop inloggen. |
| **Alternative flow 1** | De gebruiker vult een verkeerde combinatie van gebruikersnaam en wachtwoord in en drukt op de knop inloggen. Het systeem zal dan een melding geven waarna de gebruiker opnieuw kan inloggen. |
| **Alternative flow 2** | ... |
| **Exceptions** | ... |
| **Includes** | ... |
| **Extends** | ... |
| **Generalizes** | ... |
| **Preconditions** | Als de app wordt opgestart, als er een nieuw account is aangemaakt en als de gebruiker is uitgelogd. |
| **Postconditions** | Als de gebruiker is ingelogd gaat de app naar het hoofdscherm waar de gebruiker ritten kan invoeren of aanvragen. |
| **Notes/Issues** | ... |

|  |  |
| --- | --- |
| **Name and identifier** | Rit invoeren, Rit Inv |
| **Objective** | Als een chauffeur van of naar de universiteit wilt rijden kan hij dit laten registreren in onze database zodat er mensen mee kunnen rijden. |
| **Priority** | 1 |
| **Source** | Overleg 11/9 |
| **Actors** | Chauffeur |
| **Basic flow** | De chauffeur drukt op de knop nieuwe rit. Vervolgens kan hij informatie invoeren over de rit zoals opstapplaats, uitstapplaats, type auto, of er gerookt mag worden en eventuele overige opmerkingen die hij heeft. Na het invoeren drukt hij op bevestigen, waarna de rit wordt opgeslagen in de database. |
| **Alternative flow 1** | De chauffeur drukt op de knop nieuwe rit. Vervolgens kan hij informatie invoeren over de rit zoals opstapplaats, uitstapplaats, type auto, of er gerookt mag worden en eventuele overige opmerkingen die hij heeft. Na het invoeren drukt hij op annuleren, er wordt nu niks van de rit opgeslagen en de chauffeur heeft de mogelijkheid het opnieuw in te voeren. |
| **Alternative flow 2** | ... |
| **Exceptions** | ... |
| **Includes** | ... |
| **Extends** | ... |
| **Generalizes** | ... |
| **Preconditions** | Als de gebruiker is ingelogd en op het hoofdscherm op nieuwe rit drukt. |
| **Postconditions** | Na het invoeren van de rit gaat de app terug naar het hoofdscherm. De chauffeur kan nu notificaties ontvangen over de rit. |
| **Notes/Issues** | ... |

|  |  |
| --- | --- |
| **Name and identifier** | Rit aanvragen, Rit Aanv |
| **Objective** | Als een persoon wilt meerijden van/naar de universiteit kan hij een overzicht bekijken van beschikbare ritten en daarop reageren. |
| **Priority** | 1 |
| **Source** | Overleg 11/9 |
| **Actors** | Meerijder |
| **Basic flow** | Op het hoofdscherm kan de gebruiker een overzicht vinden van alle beschikbare ritten. Hij kan een rit selecteren om |
| **Alternative flow 1** | De chauffeur drukt op de knop nieuwe rit. Vervolgens kan hij informatie invoeren over de rit zoals opstapplaats, uitstapplaats, type auto, of er gerookt mag worden en eventuele overige opmerkingen die hij heeft. Na het invoeren drukt hij op annuleren, er wordt nu niks van de rit opgeslagen en de chauffeur heeft de mogelijkheid het opnieuw in te voeren. |
| **Alternative flow 2** | ... |
| **Exceptions** | ... |
| **Includes** | ... |
| **Extends** | ... |
| **Generalizes** | ... |
| **Preconditions** | Als de gebruiker is ingelogd en op het hoofdscherm op nieuwe rit drukt. |
| **Postconditions** | Na het invoeren van de rit gaat de app terug naar het hoofdscherm. De chauffeur kan nu notificaties ontvangen over de rit. |
| **Notes/Issues** | ... |

|  |  |
| --- | --- |
| **Name and identifier** | Meerijder accepteren, MeR Acp |
| **Objective** | Als een meerijder een rit aanvraagt moet deze geaccepteerd worden door de chauffeur. De chauffeur kan het profiel van de meerijder bekijken en dan een beslissing maken. |
| **Priority** | 1 |
| **Source** | Overleg 11/9 |
| **Actors** | chauffeur |
| **Basic flow** | De chauffeur krijgt een push notificatie waarin staat dat iemand wilt meerijden. De chauffeur opent het bericht waarna hij het profiel van de meerijder te zien krijgt. Hij accepteert de meerijder, dit wordt opgeslagen in de database. De applicatie gaat weer naar het hoofdscherm. |
| **Alternative flow 1** | De chauffeur krijgt een push notificatie waarin staat dat iemand wilt meerijden. De chauffeur opent het bericht waarna hij het profiel van de meerijder te zien krijgt. Hij accepteert de meerijder niet, ook dit wordt opgeslagen in de database. De applicatie gaat weer naar het hoofdscherm. |
| **Alternative flow 2** | ... |
| **Exceptions** | ... |
| **Includes** | ... |
| **Extends** | ... |
| **Generalizes** | ... |
| **Preconditions** | Als de chauffeur een rit heeft ingevoerd en een meerijder de rit heeft aangevraagd. |
| **Postconditions** | De app gaat weer naar het hoofdscherm. Als de aanvraag geaccepteerd is kan de meerijder vanaf een uur voor de rit bevestigen. |
| **Notes/Issues** | ... |

|  |  |
| --- | --- |
| **Name and identifier** | Rit bevestigen, Rit Bev |
| **Objective** | Na de rit is geaccepteerd moet deze bevestigd worden, hierna wordt de betaling automatisch voldaan. |
| **Priority** | 1 |
| **Source** | overleg 11/9 |
| **Actors** | Meerijder, chauffeur |
| **Basic flow** | De meerijder en chauffeur krijgen een push notificatie dat ze de rit kunnen bevestigen. Ze bevestigen hem hem waardoor de betaling geregeld kan worden. |
| **Alternative flow 1** | De meerijder krijgt en push notificatie en negeert hem. De meerijder hoeft nu niet te betalen en er gebeuren verder geen acties. |
| **Alternative flow 2** | De meerijder krijgt een push notificatie dat hij zijn rit kan accepteren. Hij accepteert hem niet, waardoor geen verdere actie gebeurt. |
| **Exceptions** | ... |
| **Includes** | ... |
| **Extends** | ... |
| **Generalizes** | ... |
| **Preconditions** | Als de meerijder een rit heeft aangevraagd, de chauffeur de aanvraag heeft geaccepteerd en het een uur voor de geplande vertrektijd is. De gebruiker moet voldoende saldo hebben om de rit te kunnen betalen |
| **Postconditions** | Als de meerijder het heeft bevestigt gaat de app verder naar de betaling, anders naar het hoofdscherm. |
| **Notes/Issues** | ... |

|  |  |
| --- | --- |
| **Name and identifier** | Betaling, Bet |
| **Objective** | De betaling vindt plaats na de rit, wanneer chauffeur en mee-rijder hebben bevestigd. Dat gebeurd door een automatische prijsbepaling, die zich baseert op de voorspelde route en een kilometervergoeding (afspreeklocatie tot eindbestemming). |
| **Priority** | 1 |
| **Source** | overleg 11/9 |
| **Actors** | meerijder |
| **Basic flow** | De gebruiker krijgt te zien hoe veel saldo hij had, hoe veel de rit koste en hoe veel hij na de rit heeft. |
| **Alternative flow 1** | ... |
| **Alternative flow 2** | ... |
| **Exceptions** | ... |
| **Includes** | ... |
| **Extends** | ... |
| **Generalizes** | ... |
| **Preconditions** | De meerijder moet de rit geaccepteerd hebben, wat betekend dat hij voldoende saldo heeft |
| **Postconditions** | Na de betaling gaat de app naar het hoofdscherm. |
| **Notes/Issues** | ... |

|  |  |
| --- | --- |
| **Name and identifier** | Tegoed opwaarderen/uikteren, Teg Opw/Uitk |
| **Objective** | Betaling gaat via een saldo. Dit kan worden opgewaardeerd met IDeal. Automatisch opwaarderen kan ook. |
| **Priority** | 1 |
| **Source** | overleg 11/9 |
| **Actors** | alle gebruikers |
| **Basic flow** | De gebruiker geeft aan hoe veel hij wilt opwaarderen, selecteert een bank en gaat naar de betaling. Hij doet de betaling en gaat terug naar het hoofdscherm. |
| **Alternative flow 1** | De gebruiker geeft aan hoe veel hij wilt opwaarderen, selecteert een bank en gaat naar de betaling. Hij annuleert de betaling. Hierna gaat de app weer terug naar het hoofdscherm. |
| **Alternative flow 2** | De gebruiker geeft niet aan hoe veel hij wilt opwaarderen. Hij krijgt hiervoor een bericht en kan het daarna opnieuw proberen. |
| **Exceptions** | De gebruiker blijft langer dan 5 minuten op het ideal scherm, nu wordt de betaling automatisch geannuleerd. |
| **Includes** | ... |
| **Extends** | ... |
| **Generalizes** | ... |
| **Preconditions** | Als de gebruiker te weinig saldo heeft voor het bevestigen voor een rit of als hij op saldo opwaarderen drukt in het menu. |
| **Postconditions** | De app gaat terug naar het hoofdscherm. |
| **Notes/Issues** | ... |

|  |  |
| --- | --- |
| **Name and identifier** | Chatten, Chat |
| **Objective** | Als een meerijder een aanvraag heeft gedaan en de chauffeur deze heeft geaccepteerd kunnen ze met elkaar chatten via een chatfunctie op de webapplicatie. Zo kunnen ze vragen stellen over de rit. |
| **Priority** | 1 |
| **Source** | overleg 11/9 |
| **Actors** | alle gebruikers |
| **Basic flow** | Een van de gebruikers drukt op het profiel van de andere en drukt op chatten. Kan hier berichten versturen en ontvangen. 1 uur na de geplande vertrektijd van de rit kunnen de gebruikers niet meer chatten. |
| **Alternative flow 1** | ... |
| **Alternative flow 2** | ... |
| **Exceptions** | ... |
| **Includes** | ... |
| **Extends** | ... |
| **Generalizes** | ... |
| **Preconditions** | De meerijder een aanvraag heeft gedaan en de chauffeur deze heeft geaccepteerd. |
| **Postconditions** | De app gaat terug naar het hoofdscherm |
| **Notes/Issues** | ... |

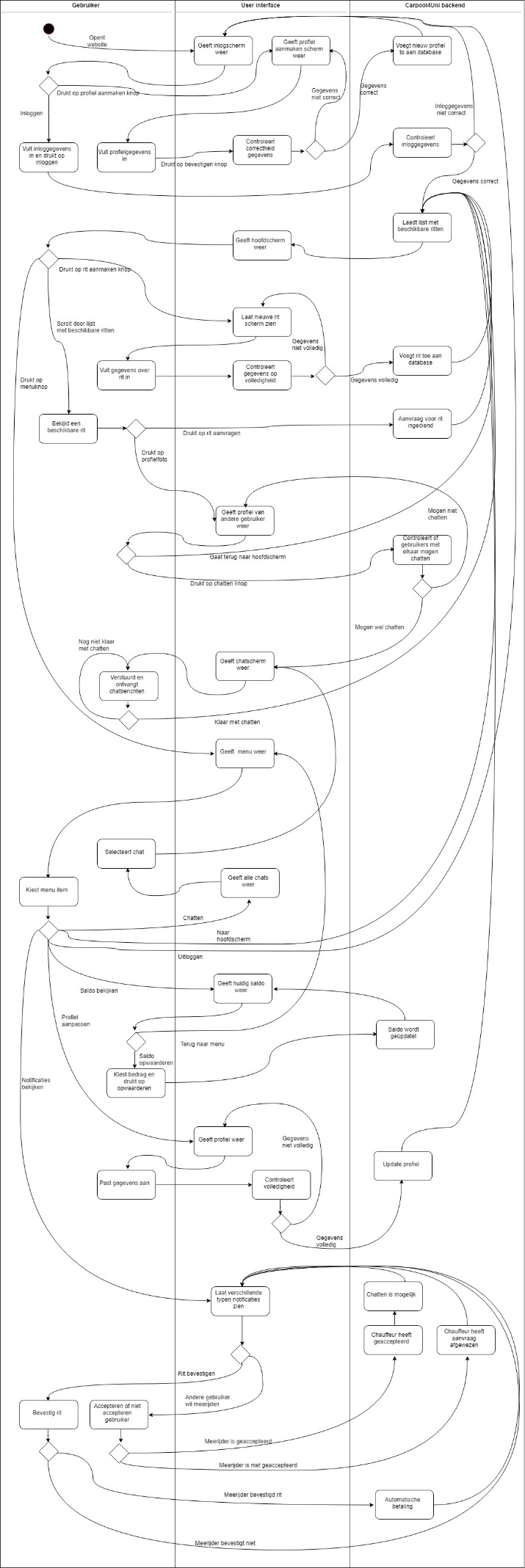
## Bijlage 4 Swimlane diagram

De volledige niet onderbroken diagram is te vinden in het zip bestand.











## Bijlage 5 Entity Relationship Diagram

## Bijlage 6 Database diagram

