Scriptie

Afstuderen Adwise

Naam:	Kenrick Dietrich
Studentnummer:	452401
Hogeschool:	Saxion
Opleiding:	HBO-ICT
Uitstroomprofiel:	Software engineering
Afstudeerbegeleider:	Peter Ebben
Afstudeerbedrijf:	Adwise B.V
Bedrijfsbegeleider:	Ken Sleebos
Datum:	13 juni 2021

Versiebeheer

Auteur	Datum	Versie	Beschrijving
Kenrick Dietrich	06-04-2021	0.1	Tussentijdse oplevering
Kenrick Dietrich	10-05-2021	0.2	Tussentijdse oplevering
Kenrick Dietrich	30-05-2021	0.3	Concept Scriptie
Kenrick Dietrich	13-06-2021	1.0	Definitieve Scriptie

Samenvatting

Voor het afstuderen van de opleiding HBO-ICT van Saxion te Enschede is een afstudeeropdracht uitgevoerd binnen Adwise. De aanleiding voor de opdracht is een probleem met de onboarding van nieuwe medewerkers. Onboarding is een proces waarbij een nieuwe medewerker de nodige kennis, vaardigheden en gedrag verwerft om sneller productief te zijn binnen het bedrijf. Tijdens de onboarding van nieuwe werknemers bij Adwise ervaren collega's dat zij elkaar steeds minder goed leren kennen door de groei van het bedrijf.

Voor de afstudeeropdracht is onderzoek gedaan met als hoofdvraag: "Hoe kan de onboarding van een nieuwe medewerker worden verbeterd door het leren kennen van collega's makkelijker te maken met behulp van technologie?" Hierbij is onderzocht hoe mensen leren, welke bestaande oplossingen zijn ontwikkeld, welke technologie helpt collega's elkaar beter leren kennen, hoe wordt een host met meerdere clients verbonden, hoe kan de applicatie white label ontwikkeld worden en hoe kan Google Cast worden geïntegreerd om gebruik te maken van de tv's op het kantoor?

Aan de hand van dit onderzoek is een applicatie ontwikkeld waarmee werknemers door middel van de spellen quiz en recall, collega's kunnen leren kennen. De quiz vorm kan met een verbinding over sockets worden gespeeld met meerdere medewerkers. De socket verbinding is ontwikkeld met Socket.io. Voor de applicatie is een front-end met Vue.js en een back-end met Node.js, Express en Socket.io ontwikkeld. De front-end is ontwikkeld als een Progressive Web App. Hiermee kan de app worden geïnstalleerd en daarmee een native ervaring bieden op de telefoon en goed functioneren op een laptop. De front-end is ook white-label ontwikkeld. Waardoor het uiterlijk makkelijk veranderd kan worden. Wanneer Adwise een klant heeft waarbij soortgelijke problemen zijn kan het uiterlijk van de applicatie snel worden veranderd om aan te sluiten aan de huisstijl van de klant.

Inhoud

1- Inleiding

Voor het afstuderen van de opleiding HBO-ICT heb ik de mogelijkheid gekregen om voor Adwise intern onderzoek te doen waaruit een applicatie ontwikkeld zal worden. De aanleiding hiervoor is een probleem in de onboarding waarbij nieuwe werknemers elkaar minder goed leren kennen. Onboarding is een proces waarbij een nieuwe medewerker de nodige kennis , vaardigheden en gedrag verwerft om sneller productief te zijn binnen het bedrijf. Om dit probleem op te lossen moet worden onderzocht hoe collega's elkaar beter leren kennen om het kennismakingsproces en communicatie te verbeteren. Aansluitend op dit onderzoek zal een passend stuk software ontwikkeld worden die een mogelijke oplossing voor het huidige probleem zal bieden. Voor deze software moet onderzoek worden gedaan naar mogelijke nieuwe bruikbare technieken, zoals front-end frameworks, PWA ondersteuning, White-label integratie en Google Cast.

Dit document beschrijft het afstudeertraject. In deze scriptie is beschreven: het probleem, de opdracht, het onderzoek, het resultaat, het proces tot dit resultaat en een reflectie op dit resultaat.

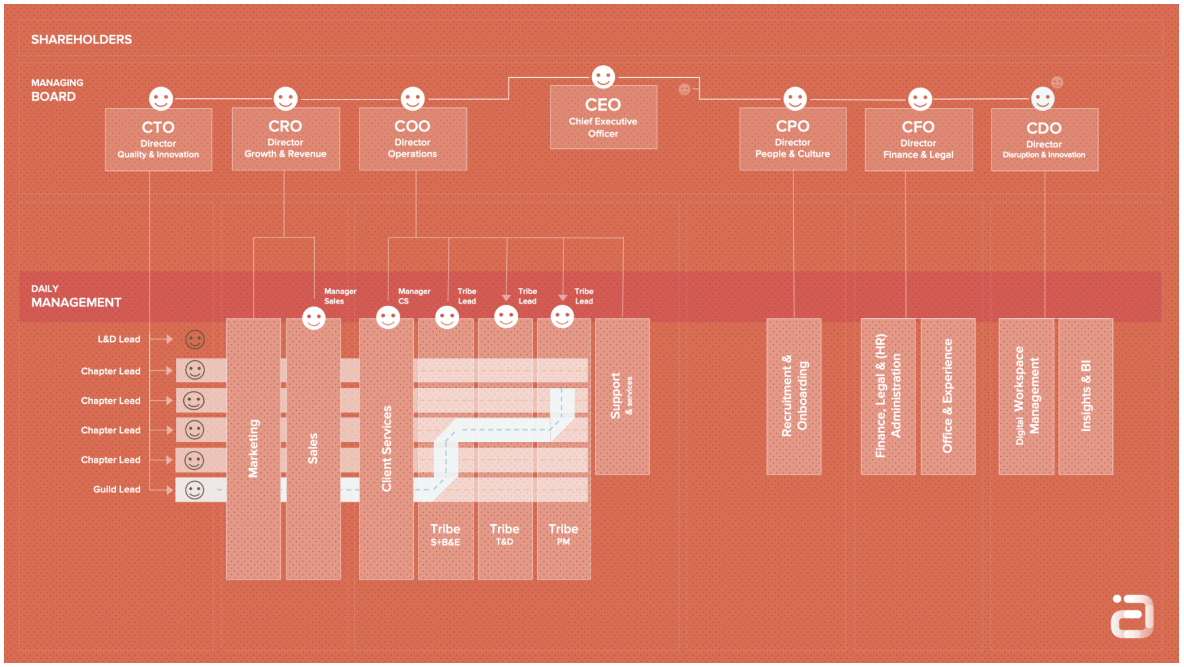
2 - Probleemstelling en opdracht

2.1 - Aanleiding

Tijdens de onboarding van nieuwe werknemers bij Adwise ervaren collega's dat zij elkaar steeds minder goed leren kennen. Elke Tiehuis, Talent Acquisition Manager, is verantwoordelijk voor de acquisitie en integratie van werknemers bij Adwise. Elke merkt dat het moeilijk leren kennen van nieuwe collega's op het moment een tweetal oorzaken heeft. Adwise is als gezond bedrijf aan het groeien, dit zorgt voor een steeds grotere groep collega's om te leren kennen en de huidige omstandigheden door COVID dragen bij aan het verminderd contact op kantoor. Deze twee oorzaken versterken elkaar en zorgen voor een moeilijkere onboarding voor nieuwe medewerkers. De beoogde oplossing is tijdens de pandemie van belang maar ook in de toekomst door de groei van het bedrijf. Adwise wil in de toekomst meer gaan thuiswerken. Hierdoor is voor de lange termijn een oplossing nodig. Elke Tiehuis is opdrachtgever voor dit project.

2.2 - Het bedrijf

Adwise is een full-service digital agency, dit houdt in dat zij voor haar klanten innovatieve en creatieve oplossingen ontwikkeld over het gehele spectrum van IT. Deze oplossingen kunnen onder andere zijn: websites, webshops, search engine optimisation, design, strategie, branding en marketing. Adwise heeft 106 werknemers waarvan 25 developers.



Figuur 2.2-1 : Organisatiestructuur Adwise

Adwise hanteert als bedrijfsstructuur een variant van het Spotify model. Adwise bestaat uit verschillende multidisciplinaire teams die squads worden genoemd. Elke squad heeft medewerkers met verschillende expertises om zo een autonoom team te vormen. Medewerkers met dezelfde expertises uit de verschillende squads vormen samen een overkoepelende chapter. Chapters zijn de verbindende factor binnen Adwise tussen

individuen en squads. Medewerkers binnen een chapter focussen zich op het innoveren en het delen van informatie binnen de organisatie. Een goede communicatie wordt bevorderd door het goed leren van collega's en is hiervoor van belang om de overdracht van informatie goed te laten verlopen. Meerdere squads die zich op dezelfde afdeling bevinden vormen samen een tribe. De tribe bepaalt de algemene richting van de squads.

2.3 - Probleemanalyse

Tijdens de onboarding van een nieuwe werknemer bij Adwise kan normaliter de werknemer zich voorstellen door een voorstelronde op het kantoor langs andere medewerkers te maken. Naast deze voorstelronde is het fysiek aanwezig zijn op kantoor een grote rol in het leren kennen van collega's en het bedrijf. Door veel mensen te kennen binnen het bedrijf blijft de communicatie laagdrempelig, dit is een aspect van Adwise waar veel waarde aan wordt gehecht. In de middag is er bijvoorbeeld een gezamenlijke lunch waarbij collega's met elkaar kunnen zitten en eten. Met deze communicatie op kantoor leert de medewerker gaandeweg het bedrijf en de collega's kennen. Met de huidige omstandigheden door COVID, en het meer thuiswerken vanuit het bedrijf, is het minder gebruikelijk voor medewerkers om op kantoor te werken. De communicatie verloopt hierdoor uitsluitend via videogesprekken, mail en berichten. Samen met de groter wordende groep collega's en moeilijkere vormen van communicatie verminderd de mogelijkheid om collega's makkelijk te leren kennen.

2.4 - Doelstelling

Het uiteindelijke doel van het project is om medewerkers een plek te geven om collega's beter te leren kennen en om informatie over collega's te kunnen opzoeken. Om deze plek te creëren moet een applicatie worden ontwikkeld die op verschillende apparaten gebruikt kan worden. Hiermee kan de medewerker waar en wanneer hij wilt de applicatie gebruiken om collega's te leren kennen. De applicatie dient als spelvorm te fungeren met gamification, dit stimuleert de gebruiker om collega's beter te leren kennen. Gamification is een term waarbij een applicatie wordt ontworpen waarbij spelvorm wordt ingezet om gebruikers te motiveren en meer interesse te wekken. Gamification kan in de applicatie worden toegepast door spelelementen te implementeren om zo collega's te stimuleren om elkaar beter te leren kennen. Deze spelelementen zullen bestaan uit een score die per spel aan de hand van de prestaties wordt toegekend. Deze score kan worden vergeleken met andere collega's op het scoreboard om te zien wie het beste is in het kennen van collega's.
De applicatie moet white-label ontwikkeld worden hiermee kan de applicatie makkelijk van uiterlijk worden veranderd, mocht in de toekomst een klant van Adwise kampen met soortgelijke problemen kan de app makkelijk worden aangepast en ingezet bij de betreffende klant.

Vanuit de opdracht moet de applicatie beschikken over een singleplayer zowel als een multiplayer spelmodus om gamification te implementeren. De singleplayer modus moet bestaan uit een quiz game met eventueel meer games, de multiplayer modus moet bestaan uit een speler die als host een kamer kan aanmaken waar andere spelers mee kunnen doen. Dit multiplayer aspect voegt toe aan de gamification van de applicatie waarbij gebruik wordt gemaakt van competitie tussen twee of meer spelers. Voor de multiplayer moet worden onderzocht welke spelvorm hiervoor geschikt is.

De applicatie dient een koppeling met Google Workplace te hebben, hierdoor kan een medewerker met zijn Adwise account inloggen op de applicatie en de score van de collega worden opgeslagen.
Het kantoor van Adwise beschikt over meerdere tv's waar chromecasts gebruikt kunnen worden. De applicatie dient informatie van het spel te kunnen casten naar een tv. Hiermee kan in de toekomst op het kantoor de multiplayer worden gespeeld met meerdere mensen rond een tv.

3 - Onderzoek

Vanuit de probleemstelling en beoogde oplossing zijn een aantal onderzoeksvragen naar boven gekomen. Gedurende het afstudeertraject is hier uitvoerig onderzoek naar gedaan. De hoofdvraag kan worden beantwoord met behulp van het onderzoeksresultaat van de deelvragen.

3.1 - Onderzoeksvragen

3.1.1 - Hoofdvraag

Hoe kan de onboarding van een nieuwe medewerker worden verbeterd door het leren kennen van collega's makkelijker te maken met behulp van technologie?

3.1.2 - Deelvragen

Vanuit de opdracht, probleemstelling en hoofdvraag komen een aantal deelonderzoeksvragen naar voren:

- Hoe leren mensen elkaar beter kennen?
- Welke bestaande oplossingen zijn al ontwikkeld?
- Hoe kijken Adwisers tegen het leren kennen van collega's aan?
- Welke technologie helpt collega's elkaar beter kennen?
- Hoe kan Google Workspace worden geïntegreerd om werknemer accounts te gebruiken voor login?
- Hoe kunnen game resultaten gekoppeld worden aan Google Workspace accounts?
- Hoe wordt een host met meerdere clients verbonden?
    - Welke spelvorm sluit hierbij aan?
- Hoe kan de applicatie white label ontwikkeld worden?
- Hoe kan Google Cast worden geïntegreerd om gebruik te maken van de tv's op kantoor?

3.2 - Onderzoeksopzet

Een groot deel van de te onderzoeken vragen zijn door middel van deskresearch onderzocht.

Om uit te zoeken of soortgelijke oplossingen bestaan is onderzoek gedaan naar bestaande oplossingen die hetzelfde doel hebben als de beoogde oplossing. Hierbij is gekeken wat deze soortgelijke oplossingen bieden en of deze ideeën toegepast kunnen worden in het eindproduct.

Voor het onderzoek naar wat Adwisers vinden van de huidige omstandigheden en de manier waarop collega's elkaar leren kennen, is een kwantitatief enquête onderzoek binnen Adwise gedaan. Hierbij is een enquête opgesteld waarbij vragen over de huidige situatie worden gesteld en wat zij een goede spelvorm vinden voor het leren kennen van collega's.

Om te achterhalen welke technologie het beste geschikt is om een applicatie te ontwikkelen, is kwalitatief onderzoek uitgevoerd. Hiervoor wordt gebruik gemaakt van deskresearch. Voor het deskresearch van deze vraag worden de 3 grootste front-end frameworks met PWA support bestudeerd. De frameworks zullen aan de hand van Google Trends, Github sterren, NPM downloads en Stackoverflow vragen worden bepaalt welke de grootste en populairste front-end frameworks zijn. Het framework moet ondersteuning hebben voor PWA doordat een PWA op desktop als een normale web app kan functioneren en op de telefoon een native ervaring kan aanbieden. Hiermee biedt de applicatie ondersteuning voor verschillende apparaten. Per framework zal een klein prototype worden ontwikkeld. Hiermee kunnen de frameworks worden vergeleken op basis van leercurve en PWA ondersteuning. Uit het deskresearch en de prototypes zullen blijken welk framework het beste geschikt is om de applicatie mee te ontwikkelen.

Voor het onderzoek naar de Google Cast integratie zal door middel van inlezen en implementeren van de Google Cast documentatie, de cast functionaliteiten worden geintegreerd (Google Cast, z.d).

De integratie met Google Workspace accounts zal op een soortgelijke manier moeten worden onderzocht als de Google Cast integratie. Beide technologieën hebben minder impact op de ontwerpkeuzes van de applicatiearchitectuur en zullen daarom uitsluitend worden onderzocht door middel van deskresearch(Google Identity, z.d).

De implementatie van een host en clients zal moeten worden onderzocht door middel van het opzetten van een klein prototype, waarbij een host met meerdere clients kan verbinden. Dit prototype zal worden ontwikkeld met het nader te onderzoeken socket framework. Dit prototype heeft als doel om in een afgebakende ontwikkelomgeving kennis te vergaren over de socket technologie en de implementatie van het framework om deze kennis vervolgens toe te passen en te implementeren in de applicatie.

Om te ondervinden hoe de applicatie white-label kan worden ontwikkeld moet door middel van deskresearch worden onderzocht hoe de applicatie gemakkelijk van uiterlijk kan wisselen. Uit onderzoek moet blijken welke technologieën hiervoor geschikt zijn en hoe deze technologieën kunnen worden toegepast met het gekozen front-end framework.

3.3 - Hoe leren mensen elkaar beter kennen?

Uit onderzoek blijkt dat mensen informatie onthouden door middel van associaties te vormen met herinneringen en gebeurtenissen. Wanneer informatie een associatie krijgt wordt het opgeslagen in het langetermijngeheugen. Informatie die aan meerdere herinneringen geassocieerd wordt is makkelijker terug te halen door de betreffende persoon. Wanneer veel informatie dezelfde associatie heeft is dit moeilijker om terug te halen. Om een gebruiker effectief te laten leren is het van belang dat met de betreffende informatie duidelijke associaties kunnen worden gelegd. Door deze duidelijke associaties wordt de informatie sneller opgeslagen in het declaratieve geheugen, dit is een vorm van lange termijn geheugen waar kennis wordt opgeslagen die bewust kan worden opgeroepen of beleeft. Naast het opslaan in het lange termijn geheugen door middel van associaties is het van belang om te kijken naar hoe de informatie gebruikt moet worden. Door informatie aan te leren op dezelfde manier als dat het gebruikt moet worden is het makkelijker voor de persoon om de informatie te reproduceren. Wanneer een antwoord zelf gereproduceerd moet worden is het goed om een activiteiten te oefenen die dit stimuleert. Wanneer een persoon alleen een antwoord moet herkennen uit een set van mogelijke antwoorden, is een multiple choice test adequaat. Het reproduceren van informatie kan beter gestimuleerd worden met open vragen. De manier van leren moet het uiteindelijke gebruik van de informatie simuleren (J. Dirksen, 2011).

Uit het bovenstaande onderzoek kan worden geconcludeerd dat een oefenvorm moet worden gebruikt waarmee de gebruiker makkelijk associaties kan maken met de op te nemen informatie. Deze oefenvorm moet aansluiten bij het praktisch gebruik en moet gebruik maken van het declaratief geheugen om zo gebruikers bewust informatie aan te leren. Om te bepalen wat een geschikte oefenvorm is, zijn criteria opgesteld. De oefenvorm moet:

- Associaties maken met informatie
- Gebruik maken van declaratief geheugen
- Ontwikkelbaar met collega informatie
- Met meerdere spelers speelbaar zijn

Door middel van deskresearch is gezocht naar mogelijke oefenvormen. Hierbij zijn vier gekozen om te vergelijken aan de hand van de criteria. Deze zijn gekozen doordat het bekende spellen zijn. Deze oefenvormen zijn:

- Quiz
- Recall
- Memory
- Kwartet

In de tabel hieronder staan de oefenvormen en de criteria in een overzicht. Een groene achtergrond betekent dat de oefenvorm voldoet, een oranje achtergrond dat het voldoet maar niet gewenst is en een rode achtergrond voldoet het spel niet.

	Quiz	Recall	Memory	Kwartet
Associaties maken	Ja	Ja	Nee	Nee
Gebruik declaratief geheugen	Ja	Ja	Nee	Ja
Ontwikkelbaar met collega informatie	Ja	Ja	Ja	Nee
Meerdere spelers	1 - ∞	1 - ∞	1 - 2	1 - 2

Figuur 3.3-1 : Criteria tabel oefenvormen

Uit bovenstaande tabel is te concluderen dat zowel quiz als recall aan de hand van de criteria een geschikte oefenvorm is voor de applicatie. Bij zowel quiz als recall kunnen associaties gemaakt worden met de informatie uit de vraag en de antwoorden. Voor beide vormen wordt gebruik gemaakt van het declaratief geheugen doordat het antwoord op de vraag bewust wordt opgehaald uit het geheugen. De vragen voor beide vormen kan op basis van de collega informatie worden gegenereerd.

Memory en kwartet zijn spellen waarbij het lastig is om associaties te maken aan nieuwe informatie. Bij memory is het doel van het spel om twee dezelfde figuren bij elkaar te zoeken. Hierdoor wordt geen nieuwe informatie geassocieerd aan andere informatie. Memory maakt ook niet gebruik van het declaratief geheugen. De speler wordt aangemoedigd om met het korte termijn geheugen de locatie van het paar te onthouden, de informatie is minder relevant zolang het gekozen paar dezelfde informatie bevat (Wikipedia, 2019).

Kwartet gebruikt een viervoud van informatie die tot dezelfde groep horen. Hierbij moet de speler de associaties van de informatie die tot dezelfde groep behoort al kennen en niet leren. Het implementeren van het spel is met de collega informatie lastig doordat informatie tot een categorie moet behoren. Hierbij is het gebruikelijk dat er van 8 tot 20 kwartetten per spel bevatten. De collega informatie bevat categorieën zoals functietitel, squad, tribe, verjaardag en hobby's. Deze categorieën zijn een beperkt aantal maar per categorie zijn er veel mogelijkheden door de grote hoeveelheid collega's. Dit zou leiden tot een spel met zes categorieën op basis van collega informatie met ruim 100 mogelijkheden om tot een viertal te komen (Wikipedia, 2021a).

Quiz wordt gespeeld door een vraag en vier antwoorden te laten zien. De gebruiker moet het juiste antwoord selecteren. Deze vraag is aan de hand van informatie over de collega gegenereerd.

Recall wordt gespeeld door de speler een aantal woorden of informatie voor een bepaalde tijd te laten zien. De speler moet vervolgens binnen de tijd raden wat het onderwerp is van de informatie. Dit kan worden toegepast door informatie over een collega te tonen en te vragen welke collega bij de informatie hoort.

Beide spelvormen kunnen alleen gespeeld worden maar ook multiplayer door te wachten totdat elke speler heeft geantwoord of een timer laten aflopen om de spelers binnen de tijd te laten antwoorden. Per ronde kunnen spelers een poging doen om het antwoord juist in te vullen. Wanneer een speler goed antwoord krijgt hij een aantal punten. De speler met de meeste punten of goed geraden antwoorden wint.

3.4 - Welke bestaande oplossingen zijn al ontwikkeld?

Om een goede oplossing te ontwikkelen voor het leren kennen van collega's kan worden gekeken naar oplossingen die al bestaan. Hierdoor is gezocht naar applicaties die leren bevorderen en gebruik stimuleren met gamification. Voor het onderzoek is gekozen om drie applicaties toe te lichten om binnen de scope van het project te blijven. De drie gekozen applicaties verschillen in doel maar beschikken over functionaliteiten die interessant zijn en een mogelijke inspiratiebron voor de applicatie.

3.4.1 - Knowingo

Knowingo is een LMS of Learning Management System, gericht op bedrijven om medewerkers te helpen met het leren van belangrijke informatie binnen het bedrijf. Dit kan informatie over de onboarding zijn of bedrijfsinformatie. Door de mogelijkheid om eigen vragen te maken voor de app kan de inhoud worden aangepast aan de eisen van het bedrijf. De gebruiker heeft een overzicht van alle mogelijke vragenlijsten met de voortgang en kan via het overzicht kiezen om aan een nieuwe vragenlijst te beginnen of een andere vragenlijst te hervatten. Wanneer de gebruiker vragen goed beantwoord ontvangt hij punten die aan het gebruikersaccount worden toegevoegd. Met genoeg punten gaat een gebruiker een niveau omhoog. Knowingo biedt naast de app een webportaal aan waarin alle informatie en prestaties van de medewerkers in een overzicht worden getoont (Knowingo B.V., z.d.).

Knowingo is in het verleden gebruikt binnen Adwise maar is na een korte periode mee gestopt. De reden voor het stoppen met de applicatie is aan Elke Tiehuis gevraagd die aangaf dat het een mooie simpele app was maar waar uiteindelijk door een kosten kwestie mee is gestopt. Volgens Elke Tiehuis leek de applicatie voor de gebruikers op Kahoot, dit is een online website waar vragenlijsten kunnen worden gemaakt, waarbij verschillende gebruikers met hun telefoon kunnen meedoen en vragen kunnen beantwoorden.

Knowingo was een mogelijke oplossing door vragenlijsten te maken over collega's om elkaar beter te leren kennen. Naast de kosten van de applicatie moeten de vragen bijgehouden worden om relevant te blijven. Wanneer collega's beginnen bij Adwise of Adwise verlaten moeten de vragenlijsten handmatig worden aangepast.

Knowingo heeft zoals eerder beschreven, een webportaal. Dit is een handige toevoeging om inzicht te geven over de prestaties van de medewerkers. Knowingo kan naast vragen over onboarding ook gebruikt worden voor bedrijf specifieke vragen zoals de geschiedenis of visie van het bedrijf. Deze veelzijdigheid kan worden gezien als een voordeel maar ook als een nadeel door overbodige functionaliteiten.

3.4.2 - Icogny

Icogny is een party app die op zowel Android als iOS kan worden gebruikt. De app wordt gespeeld met een groep van 3 tot oneindig. Een speler kan een groep aanmaken waar andere spelers aan mee kunnen doen. Gedurende het spel krijgen de spelers vragen in beeld die alle spelers individueel moeten beantwoorden. Wanneer iedereen heeft geantwoord kunnen de spelers gokken wat andere spelers hebben geantwoord. Het spel bevat een aantal standaard vragen en kunnen nieuwe aangepaste vragen worden toegevoegd. Het nadeel van Icogny is dat de applicatie zich richt op een informeel gezelschap. De standaard vragen zoals “Wie schiet zijn vriend in het been voor een miljoen euro?” zijn niet toepasselijk voor een meer formele werksfeer binnen Adwise, daarnaast is het doel van het spel om iemands gedrag te voorspellen en niet om nieuwe informatie over een medespeler te leren. (Incogny - Know your friends, z.d.).

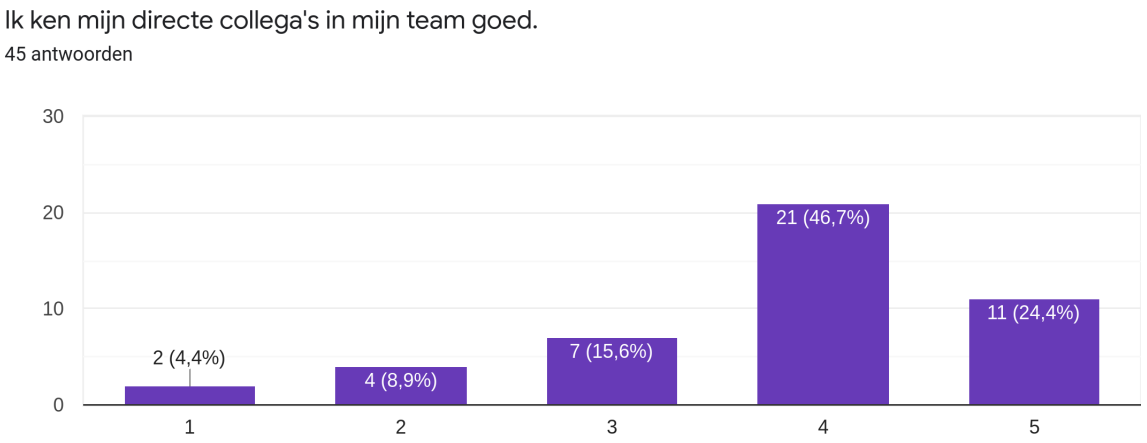
De multiplayer functionaliteiten waarbij een speler een groep kan aanmaken waarbij andere spelers aan mee kunnen doen, kan een inspiratiebron zijn voor de multiplayer modus van de te ontwikkelen applicatie.

3.4.3 - Lead

Lead is een Slack of Teams bot die collega’s met elkaar kan verbinden. Het kan virtuele lunchrooms of coffeecorners organiseren. De bot maakt een ruimte aan die medewerkers kunnen joinen, die gebruikt kan worden om bij te praten of te overleggen. Dit is een oplossing die de drempel van communicatie verlaagt (*Lead - Helps connect teams*, z.d.).

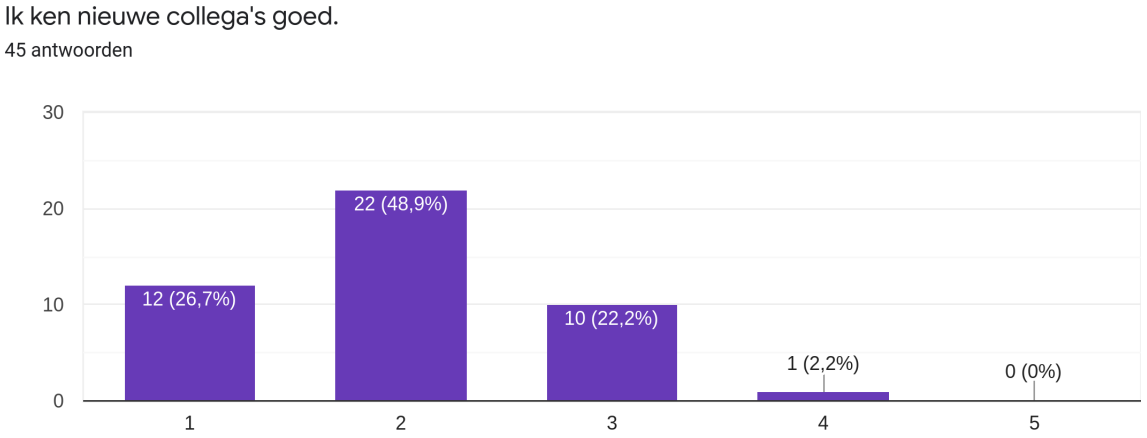
3.5 - Hoe kijken Adwisers tegen collega's leren kennen aan?

Om te onderzoeken wat Adwisers over het leren kennen van collega's vinden, is een kwantitatief enquête onderzoek gedaan. De enquête is gemaakt met Google forms en verstuurd via de mail binnen Adwise. De enquête is na een eerste versie gedeeld voor overleg met Elke Tiehuis. De enquête is 45 keer ingevuld. Alle enquêteresultaten zijn terug te vinden in bijlage (Bijlage 11.2).



Figuur 3.5-1 : Enquête resultaat: Ik ken mijn directe collega's in mijn team goed.

Uit onderzoek is gebleken dat Adwisers vinden hun directe collega's goed te kennen. Een grote meerderheid geeft aan eens te zijn met de stelling: Ik ken mijn directe collega's in mijn team goed.



Figuur 3.5-1 : Enquête resultaat: Ik ken nieuwe collega's goed.

Nieuwe collega's leren kennen blijkt een struikelblok te zijn, 48% geeft aan niet eens te zijn met de stelling "Ik ken nieuwe collega's goed".

Ondanks het moeilijker leren kennen van nieuwe collega's blijkt uit de enquête dat Adwisers collega's leren kennen erg belangrijk vinden, naast het belangrijk vinden van collega's leren kennen geeft 93% aan collega's beter te willen leren kennen, 69% geeft aan meer te willen doen om collega's te leren kennen maar dat zij dit als lastig ervaren door de huidige thuiswerk situatie.

Elke had als toevoeging op de enquête een drietal vragen over welke informatie collega's van elkaar geïnteresseerd in zin.

Deze vragen zijn:

- Ik ben geïnteresseerd in het werkverleden van mijn (nieuwe) collega's.
- In ben geïnteresseerd in het gezinsleven van mijn (nieuwe) collega's.
- Ik ben geïnteresseerd in de hobby's/vrijetijdsbesteding van mijn (nieuwe) collega's.

Uit de drie vragen werkverleden, gezinsleven en vrijetijdsbestedingen bleek vrijetijdsbestedingen de meeste interesse te wekken. 35% van de werknemers gaf aan hier zeer mee eens te zijn en 62% mee eens. Het werkverleden en het gezinsleven gaven allebei een soortgelijk resultaat rond de 23% gaf aan hier zeer mee eens en 62% gaf aan hier mee eens te zijn. Voor alle drie vragen gaven medewerkers aan interesse in te hebben.

Uit de enquête komt ook naar voren dat medewerkers digitaal leren makkelijk ervaren en dat de telefoon en laptop veel worden gebruikt. In tegenstelling tot de tablet, dit apparaat wordt door medewerkers weinig gebruikt, 68% geeft aan de tablet niet dagelijks te gebruiken.

3.5.1 - Conclusie

Medewerkers die al een langere tijd bij Adwise werken kennen elkaar goed. Nieuwe en huidige collega's leren elkaar minder goed kennen door de groei van het bedrijf en de huidige thuiswerk situatie. De onderzoeksresultaten bevestigen de ingeving van Elke Tiehuis over het leren kennen van nieuwe collega's vanuit de organisatie. Medewerkers geven aan dat zij collega's leren kennen belangrijk vinden voor de samenwerking en dat zij graag nieuwe collega's beter willen leren kennen. Uit dit onderzoek kan worden geconcludeerd dat het probleem, het leren kennen van nieuwe collega's vanuit de organisatie wordt gedragen

3.6 - Welke technologie helpt collega's elkaar beter leren kennen?

Vanuit de opdracht moet een applicatie worden ontwikkeld voor het probleem binnen Adwise. Het enquête onderzoek gaf aan dat medewerkers het meest hun telefoon en laptop gebruiken. De applicatie moet hierdoor responsive zijn en goed in gebruik voor zowel laptop als telefoon. Om de applicatie voor mobiel en desktop vriendelijk te ontwikkelen is gekozen om een PWA te ontwikkelen. Een PWA kan op desktop als een normale web app functioneren en op de telefoon een native ervaring aanbieden. Hierdoor kan met een codebase, de applicatie op desktop als mobiel gebruikt worden.

Een PWA is een Progressive Web App, een PWA wordt ontwikkeld met behulp van moderne webtechnologieën waaronder HTML, CSS en Javascript. Een PWA is een webapplicatie waarbij een "service worker" en web manifest is toegevoegd, de service worker voegt functionaliteiten toe die zorgt dat de applicatie kan fungeren als een soort native app zoals het installeren van de applicatie voor offline gebruik en push notificaties. De web manifest is een JSON bestand waarin alle metadata van de app in is beschreven. Deze metadata wordt gebruikt wanneer de app voor offline gebruik wordt gedownload. In de metadata staat welk icoon gebruikt moet worden, de titel en eventuele snelkoppelingen die gebruikt kunnen worden met bijvoorbeeld het starten een spel vanuit het hoofdscherm van de telefoon door het app icoon lang ingedrukt te houden en de snelkoppeling aan te klikken.

Een PWA beschikt over een aantal functionaliteiten die het onderscheid van een responsive webapplicatie. Deze functionaliteiten zijn:

- Offline mogelijkheden
- Native telefoon functionaliteiten
- Automatische updates

3.6.1 - Welk framework

Verschillende moderne frameworks hebben tegenwoordig PWA ondersteuning, om te onderzoeken welk PWA framework het beste geschikt is om de applicatie mee te ontwikkelen, is onderzoek gedaan naar de drie grootste front-end frameworks met PWA ondersteuning. Dit is gedaan door middel van deskresearch. Hierbij zijn een aantal criteria opgesteld om te bepalen wat het meest geschikt framework is. Deze criteria zijn:

- PWA support
- Grote community (Github sterren, NPM downloads, Stackoverflow vragen)
- Kleine bundel grootte
- Leercurve
- Voorkeur vanuit Adwise

Hierbij is het belangrijkst dat het framework als PWA ontwikkeld kan worden. Een grote community is belangrijk zodat het framework een grotere kans heeft op ondersteuning van verschillende tools. Daarnaast zorgt een grotere community voor meer vragen en antwoorden van voorgaande developers. Hiermee kan het complexe probleem al een keer opgelost zijn. Een kleine bundel grootte is van belang zodat de applicatie snel laadt. De

leercurve is van minder belang maar kan in de beginfase van de ontwikkeling, het proces versnellen of vertragen door de aanwezige of ontbrekende ervaring. De voorkeur vanuit Adwise is van belang door het mogelijk door ontwikkelen in de toekomst.

Om te bepalen welke front-end frameworks het grootst zijn is met behulp van Google Trends en Github gekeken naar de activiteit rondom het framework. In de tabel hieronder is een kort overzicht van de criteria en de drie frameworks getoond. Een groene achtergrond betekent dat het framework voldoet, een oranje achtergrond dat het voldoet maar niet gewenst is en een rode achtergrond voldoet het framework niet.

	Angular	React	Vue
PWA support	Ja	Ja	Ja
Github sterren	71.800	165.000	181.000
NPM downloads (wekelijks)	531.724	10.615.795	2.351.345
Stackoverflow vragen	262,181	309,360	78,900
Bundle grootte	250 kB	45 kB	46 kB
Leercurve	Moeilijk	Gemiddeld	Makkelijk
Voorkeur Adwise	Nee	Nee	Ja

Figuur 3.6-1 : Criteria tabel frameworks

Tijdens het deskresearch is gebleken dat aan de hand van de criteria zowel Angular, React als Vue geschikte frameworks zijn die gebruikt kunnen worden voor het ontwikkelen van de applicatie. Alle drie de frameworks hebben PWA support, kunnen white label ontwikkeld worden, kunnen gebruik maken van JavaScript SDK's, libraries zoals Google Cast, Workspace en sockets. Daarnaast hebben de frameworks een grote community met developers voor vragen en tools.

Angular verschilt het meest van React en Vue. De bundel is groter dan React en Vue en de leercurve is steiler. Uit eigen ervaring is tijdens het ontwikkelen van het prototype gebleken dat de folder structuur vrij onoverzichtelijk en rommelig is. Hierdoor zal de applicatie niet ontwikkeld worden met Angular.

React en Vue liggen erg dichtbij elkaar. React heeft een grotere community maar de leercurve ligt hoger. Om tot een conclusie te komen is een poll gehouden binnen Adwise om de developers te vragen wat gebruikt wordt binnen Adwise en waar de voorkeur ligt. Uit deze poll blijkt (Bijlage 11.1) dat Vue een duidelijke voorkeur heeft. Vue is hierdoor gebruikt om de

applicatie te ontwikkelen. Hierdoor kan de applicatie in de toekomst verder worder ontwikkeld door de voorkeur en kennis van Vue Binnen Adwise. Dit sluit aan bij mijn persoonlijke voorkeur voor Vue.

3.7 - Hoe wordt een host met meerdere clients verbonden

Om de applicatie van multiplayer functionaliteiten te voorzien moeten verschillende clients met een hostserver verbonden worden. Deze multiplayer functionaliteiten eisen een voortdurende verbinding tussen de spelers om zo elkaars acties en berichten in real time te kunnen zien. Hierdoor zijn Websockets voor een webapplicatie noodzakelijk.

Websocket is een netwerkprotocol dat bidirectionele communicatie biedt over een enkele TCP verbinding tussen clients en een server. TCP staat voor Transmission Control Protocol en zorgt voor een betrouwbare verbinding. TCP heeft als kenmerk dat het gegevens in een datastroom kan versturen waarbij de garantie wordt geleverd dat de gegevens aankomen zoals ze verstuurd werden. Websockets over TCP biedt een betrouwbare en constante verbinding tussen clients en maakt het mogelijk om real time berichten en updates te sturen (Wikipedia, 2020).

Het Websocket protocol wordt ondersteund door de meeste populaire browsers zoals Chrome, Edge, Firefox, Safari en Opera.

Websocket maakt gebruik van een server die de verbinding tussen clients opzet. Deze server zorgt voor een handshake tussen clients en zet de verbinding op. De server houdt ook bij wanneer een cliënt disconnect. Deze bibliotheken bouwen verder op het Websocket protocol en voegen extra functionaliteiten toe om het makkelijker te maken voor de ontwikkelaar. Drie populaire Websocket bibliotheken zijn:

- Socket.io
- ws
- Sockjs

Deze bibliotheken zijn gekozen als grootste aan de hand van de Github sterren en NPM downloads. Hierbij zijn een aantal criteria opgesteld om te bepalen wat het meest geschikte socket framework is. Deze criteria zijn:

- Grote community (Github sterren, NPM downloads, Stackoverflow vragen)
- Extra functionaliteiten
- Leercurve
- Front-end integratie
- Eerste release
- Huidige versie

Een grote community zorgt voor meer vragen en antwoorden van voorgaande developers. Hierdoor kan een complex probleem al eerder zijn opgelost. Extra functionaliteiten kunnen zorgen voor een snellere ontwikkeling doordat een eigen implementatie hierdoor niet nodig is. De integratie met de front-end is belangrijk. Wanneer de applicatie een Vue package heeft kan dit de ontwikkeling bevorderen. Met de eerste release zowel als de huidige versie kan worden gekeken of het framework niet net nieuw is en nog steeds wordt onderhouden.

Wanneer een framework net nieuw is kan het functionaliteit ontbreken. Wordt een framework niet meer ondersteund, dan kan dit toekomstige problemen voor de applicatie veroorzaken zoals bugs of beveiligingsproblemen die niet meer worden opgelost. Hieronder is een kort overzicht weergeven van de drie grootste Websocket bibliotheken.

	Socket.io	ws	Sockjs
Github sterren	53.000	16.000	9.500
NPM downloads (Wekelijks)	8,928,488	38,314,606	8,615,689
Stackoverflow vragen	18.500+	206	619
Extra functionaliteiten	Fallback, broadcasting, reconnecting, rooms	Geen	Geen
Leercurve	Makkelijk	Gemiddeld	Makkelijk
Front-end integratie	Vue cliënt	Geen	Javascript client
Eerste release	v0.1, 2010	v0.4.3, 2013	v0.0.1, 2011
Huidige versie	4.1.2	7.4.6	0.3.21

Figuur 3.7-1 : Criteria tabel socket frameworks

Uit deze tabel is te concluderen dat Socket.io met 53.000 Github sterren en 18.500 Stackoverflow vragen een veel grotere community heeft dan ws en Sockjs. De wekelijkse NPM downloads geven een ander beeld over het gebruik van de bibliotheek echter zijn deze cijfers niet een correcte weerspiegeling van het gebruik van de bibliotheken. Socket.io maakt gebruik van Engine.io, dit is een implementatie van het Websocket protocol die als dependency ws gebruikt. Wanneer Socket.io word geinstalleerd word ook ws als dependency gebruikt, dit zorgt voor een vertekend beeld bij de NPM downloads.

Ws is een barebones implementatie van het Websocket protocol waarop Socket.io verder bouwt en extra functionaliteiten toevoegt zoals fallback opties wanneer de browser van de gebruiker geen Websockets ondersteund, broadcasting naar alle clients en wanneer een cliënt de verbinding verbreekt, probeert Socket.io automatisch de verbinding te herstellen (Socket.io, 2021).

Ws kan als zelfstandige bibliotheek worden gebruikt maar zal extra functionaliteiten ontbreken die zelf ontwikkeld moeten worden (Ws, z.d.).

Sockjs maakt gebruik van de Faye Websocket implementatie en is erg vergelijkbaar met Socket.io. Faye Websocket is een library die is ontstaan uit het Faye project. Faye is publish and subscribe berichten applicatie die gebruikt kan worden om via een server berichten te sturen (Sockjs, z.d.).

Door de grotere community, uitgebreide functionaliteiten en betere front-end integratie is gekozen om het Websocket protocol te implementeren met het gebruik van Socket.io.

3.8 - Hoe kan de applicatie white-label ontwikkeld worden

De applicatie heeft als primaire doel, collega's elkaar beter te laten leren kennen binnen Adwise. De opdracht vanuit Adwise is de applicatie ontwikkelen voor intern gebruik. Echter door white-label support te integreren in de applicatie kan Adwise in het vervolg de applicatie aan klanten aanbieden die hetzelfde probleem ervaren binnen het bedrijf. Wanneer Adwise een klant vindt voor de applicatie kan snel een versie worden ontwikkeld met minimale configuratie om aan te sluiten aan de huisstijl van de klant.

Het uiterlijk van een applicatie wordt gedefinieerd in een CSS bestand. CSS staat voor Cascading Style Sheets. Dit CSS bestand kan in de front-end worden gedefinieerd of vanuit de back end worden opgevraagd door middel van een request.

Een front-end white-label oplossing bestaat uit een project waarbij verschillende CSS bestanden zijn gedefinieerd. Wanneer de applicatie tot een productie versie wordt gebouwd wordt aan de hand van "environment" variabelen het juiste CSS bestand gebruikt. Environment variabelen zijn variabelen die bij het bouw command worden meegegeven of uit een los .env bestand komen. Een environment bestand wordt alleen lokaal bewaard omdat er vaak andere variabelen instaan die geheim zijn en niet online of een versiebeheer platform moeten worden gedeeld.

Een back end white-label oplossing kan door middel van het dynamisch inladen van styling. Wanneer de applicatie wordt geladen, wordt een verzoek vanuit de front-end naar de back end gedaan om de CSS dynamisch in te laden. De geladen CSS wordt gelinkt aan de HTML en wordt vervolgens getoond. Kan de applicatie de styling niet laden dan laadt de applicatie niet of valt de applicatie terug een default styling.

Naast de applicatie styling is het mogelijk om andere onderdelen te white-labelen. Een PWA applicatie kan worden geïnstalleerd. Hoe het app icoon, naam en thema kleur er uit ziet wanneer de applicatie is geïnstalleerd staat in een "manifest.json" bestand. Deze app informatie kan worden aangepast door verschillende "manifest.json" bestanden te genereren. Dit kan zowel in de front-end tijdens het bouwproces of wanneer de applicatie wordt ingeladen, vanuit de back end worden opgehaald.

Voor deze opdracht is gekozen om de white-label integratie te implementeren in de front-end. Hiermee is de juiste styling en het manifest bestand al bij de productie versie gebundeld. Hiermee wordt overbodige standaard styling voorkomen. Daarnaast is een front-end implementatie minder gevoelig voor het uitvallen van de back-end of slechte verbinding waardoor de styling niet zou kunnen laden.

3.9 - Hoe kan Google Cast worden geïntegreerd om gebruik te maken van de tv's op kantoor

Voor het delen van multiplayer informatie op een tv via Google Cast moet de applicatie beschikken over Cast integratie. Hiermee kan de vraag van een multiplayer spel worden getoond op de tv waarbij elke speler antwoorden op zijn of haar apparaat te zien krijgt. Met deze functionaliteit kan in de toekomst het spel op het kantoor met collega's rond een tv worden gespeeld.

Google Cast is een protocol dat door Google is ontwikkeld voor het afspelen van audio en video op beeldscherm met het gebruik van een Chromecast. Naast het afspelen van audio en video kan Google Cast worden gebruik voor het besturen van de media met het gebruik van een mobiel apparaat, pc of smart speaker. Het protocol is gelanceerd op 24 juli 2013. De SDK is op 3 februari 2014 uitgebracht. Met deze SDK kunnen developers het Cast protocol integreren in applicaties van derden.

Google Cast apparaten kunnen op twee manieren inhoud streamen. De eerste maakt gebruik van mobiele en web-apps die het protocol ondersteunen om media vanuit de app te streamen, de tweede maakt het spiegelen van de webbrowser Google Chrome op een pc mogelijk, evenals het scherm delen via de Google Home app op een Android telefoon. In beide gevallen wordt het afspelen gestart via de "cast"-knop op het apparaat.

De primaire methode voor het afspelen van media op het apparaat is via mobiele en web-apps met Cast integratie. De apps regelen zelf het afspelen en volume. De Google Cast streamt zelf de media van internet in een lokale versie van de Chrome browser. Hierdoor kan het apparaat die de media "cast" andere applicaties gebruiken (Wikipedia, z.d.).

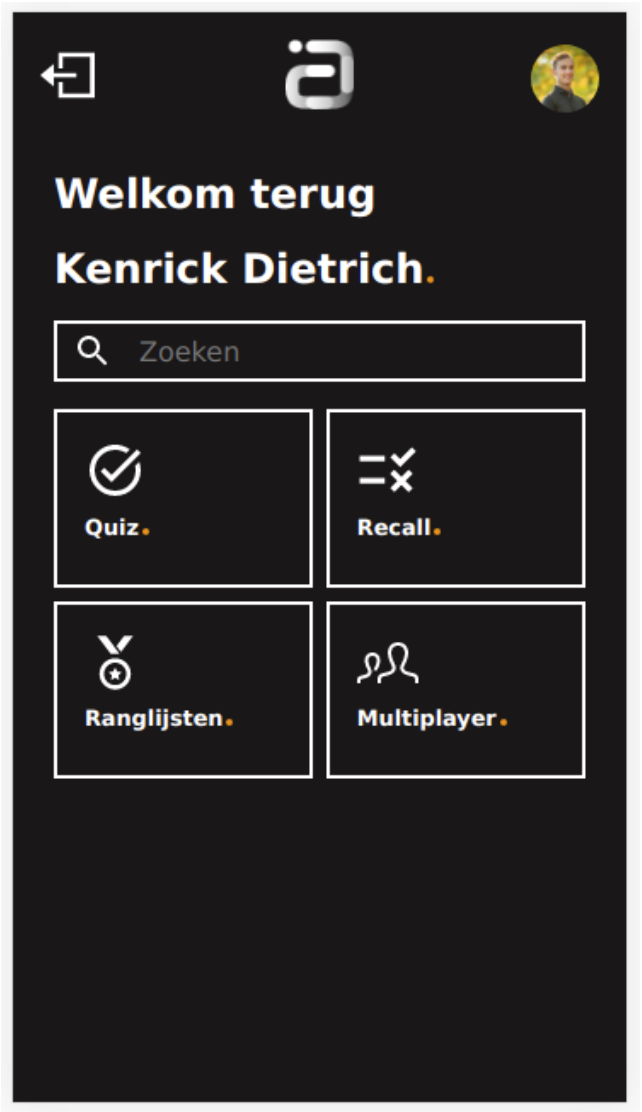
Het delen van het scherm via Chrome op de PC of Google Home op de telefoon is een mogelijke oplossing maar niet ideaal. Hiermee zal een speler zijn scherm moeten delen en daarmee ook zijn of haar antwoord laten zien.

Google Cast in de applicatie integreren is gebruiksvriendelijker doordat de gebruiker geen extra software nodig heeft om het spel te kunnen casten. Daarnaast wordt het scherm dan niet gedeeld maar alleen de vraag. Hierdoor kunnen andere spelers niet zien wat de "host" heeft geantwoord.

Voor het integreren van Google Cast moet een Web Receiver worden geschreven die de content ontvangt en toont in Chrome op de Google Cast. De standaard "Styled Media Web Reciever" ondersteund alleen audio en video content. Om andere inhoud te kunnen streamen moet een Custom Web Receiver worden ontwikkeld. Deze Custom Web Receiver kan HTML ontvangen waardoor de applicatie kan worden getoond (Web Receiver Overview | Cast | z.d.).

4 - Resultaat

Aan het begin van de ontwikkeling van de applicatie zijn requirements en userstories opgesteld (Dietrich, 2021). De applicatie is aan de hand van deze requirements ontwikkeld. In dit hoofdstuk wordt het resultaat van de ontwikkeling beschreven. Hierbij wordt beschreven hoe de applicatie is vormgegeven en welke functionaliteiten zijn geïmplementeerd.



Figuur 4-1: Homepagina van applicatie

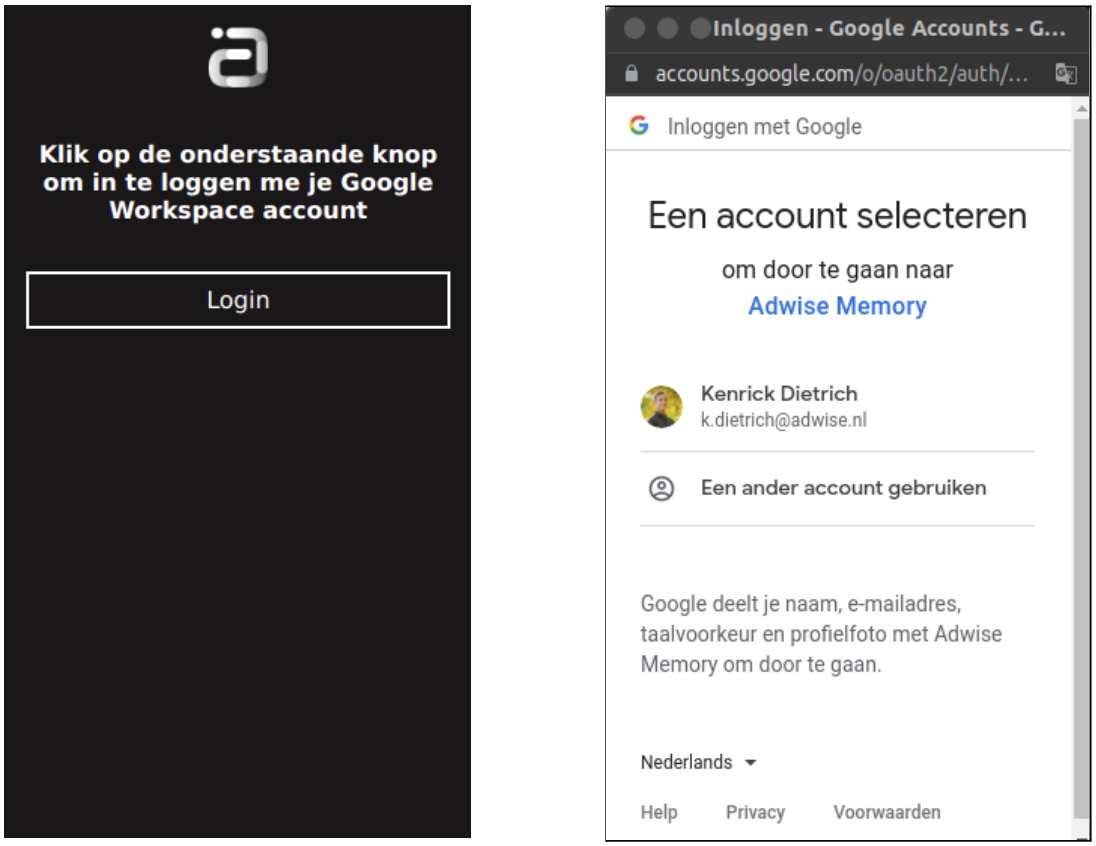
Hierboven is de hoofdpagina van de applicatie weergegeven. Op deze hoofdpagina heeft de gebruiker meerdere mogelijkheden zoals het zoeken van collega's, de verschillende spelvormen spelen, het profiel bekijken en de ranglijsten bekijken.

4.1 - Functionaliteiten

De applicatie beschikt over een aantal functionaliteiten. Deze functionaliteiten zijn:

- Inloggen met Google Workspace account
- Wisselen van styling met white-label support
- Quiz en recall spelen
- Highscores zien
- Collega's zoeken
- Multiplayer quiz spelen
- PWA functies

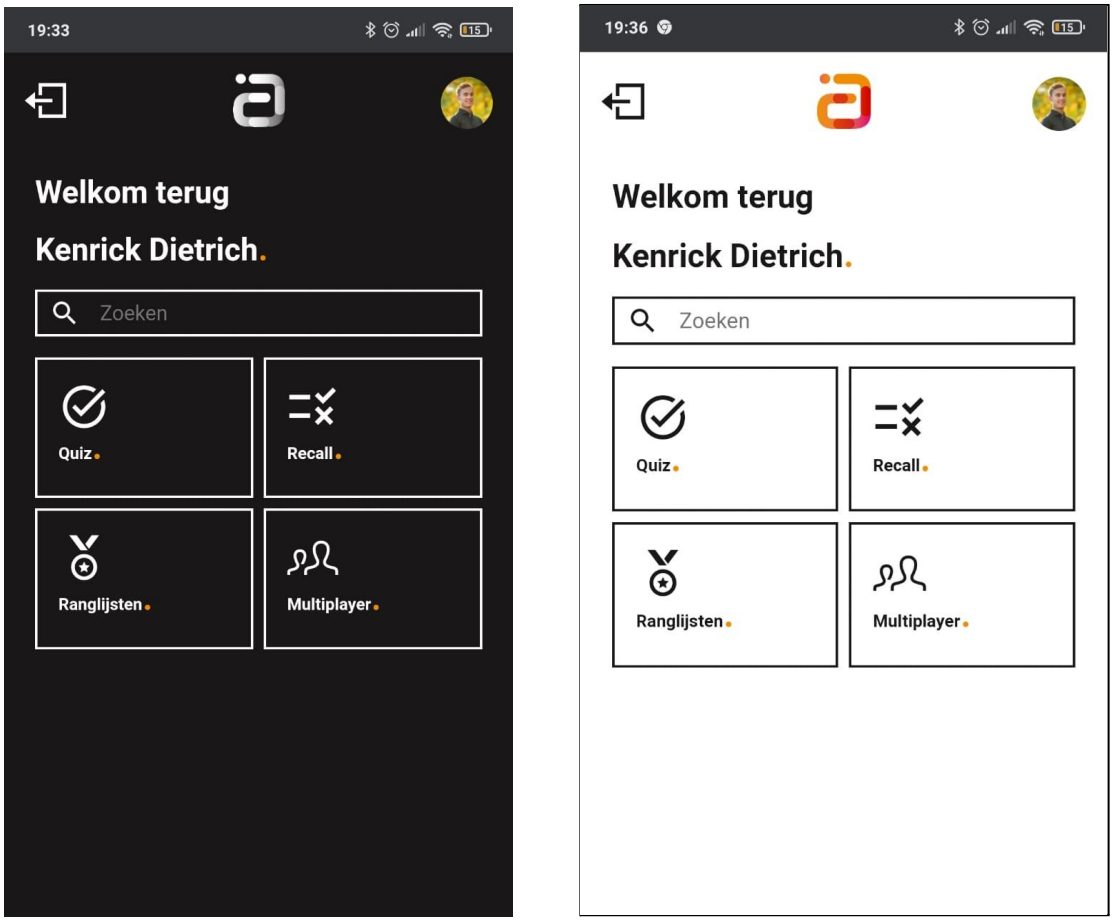
4.2 - Inloggen met Google Workspace account



Figuur 4.2-1 : Login pagina met Google OAuth2 consent scherm

Een medewerker kan inloggen door op de login knop te klikken, vervolgens opent een authenticatie scherm van Google waarbij de gebruiker zijn Adwise account kan selecteren en na een succesvolle authenticatie wordt doorgestuurd naar het hoofdscherm. Gebruikers kunnen alleen inloggen met een account die gekoppeld is aan het Adwise Google Workspace domein.

4.3 - Wisselen van styling met white-label support



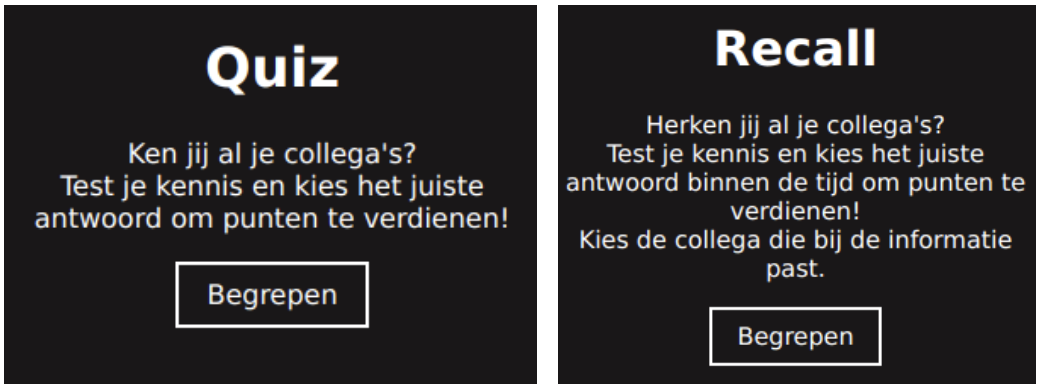
Figuur 4.3-1 : Applicatie in twee verschillende white-label versies

De applicatie kan makkelijk van styling worden veranderd door middel van een app brand te specificeren tijdens het bouwen van de applicatie. Naast het wisselen van styling kunnen plaatjes en logo's van de applicatie worden veranderd.

Figuur 4.3-2 : Logo's en app namen in verschillende white-label versies

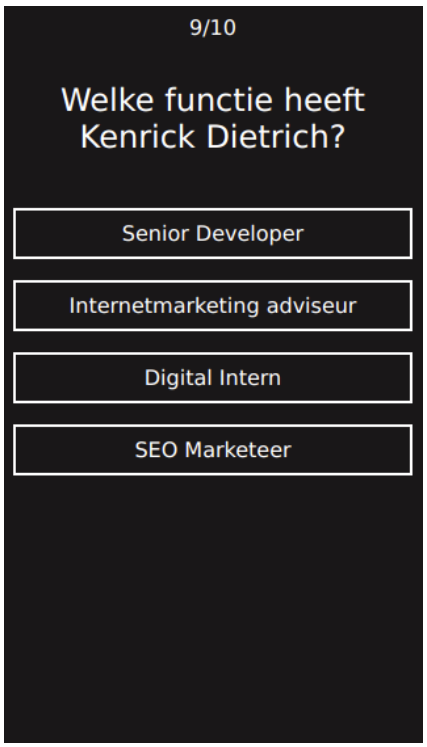
4.4 - Quiz en recall spelen

Medewerkers kunnen vanuit het hoofdscherm Quiz of Recall aanklikken een van de twee spelvormen te spelen. De gebruiker ziet een korte uitleg en kan vervolgens beginnen met spelen. Wanneer de gebruiker tien vragen heeft beantwoord krijgt de speler een einde spel bericht waar de hoeveelheid goede antwoorden te zien is en de opties om terug of opnieuw te spelen.



Figuur 4.4-1 : Uitleg Quiz en Recall

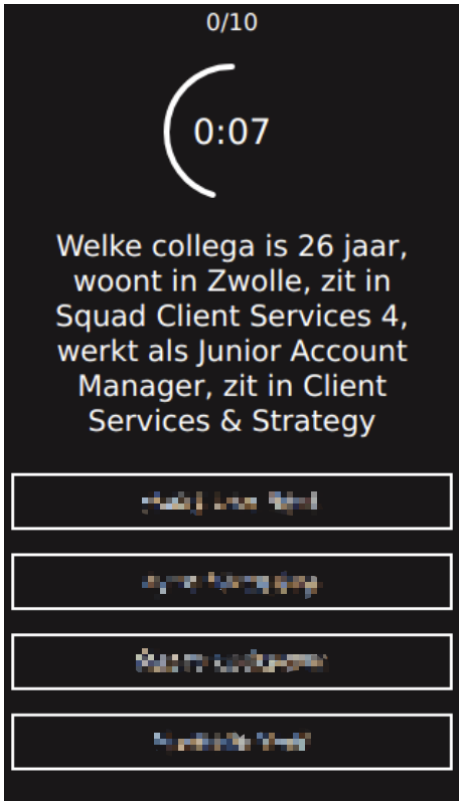
Figuur 4.4-2 : Einde spel bericht



Figuur 4.4-2 : Quiz spel

Bij het quiz spel ziet de gebruiker bovenaan het scherm de hoeveelheid vragen die hij al heeft beantwoord. Hieronder een willekeurig gegenereerde vraag op basis van de data uit de Adwise API, met als laatst vier antwoorden die ook op basis van werknemer informatie zijn gegenereerd.

Selecteert de gebruiker een antwoord dan veranderen de knoppen van kleur aan de hand van het goede antwoord. Het goede antwoord groen en de andere antwoorden rood. Na drie seconden wordt een nieuwe vraag getoond.

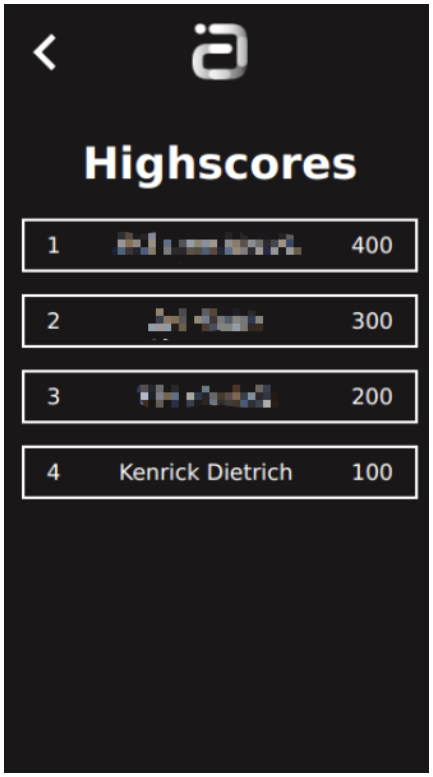


Bij het recall spel krijgt de gebruiker een statement met informatie te zien over een collega. De speler moet binnen de tijd achterhalen over welke collega de statement gaat.

Selecteert de gebruiker een antwoord dan veranderen de knoppen op dezelfde wijze van kleur als bij de quiz. Wanneer de timer op nul staat wordt het goede antwoord getoond en verschijnt na drie seconden een nieuwe statement.

Figuur 4.4-3 : Recall spel

4.5 - Highscores inzien



Figuur 4.5-1 : Highscore spel

Medewerkers kunnen highscores van de vijftien collega's met de meeste punten verdient. Wanneer een collega nog geen score heeft wordt hij niet getoond in de lijst.

4.6 - Collega's zoeken

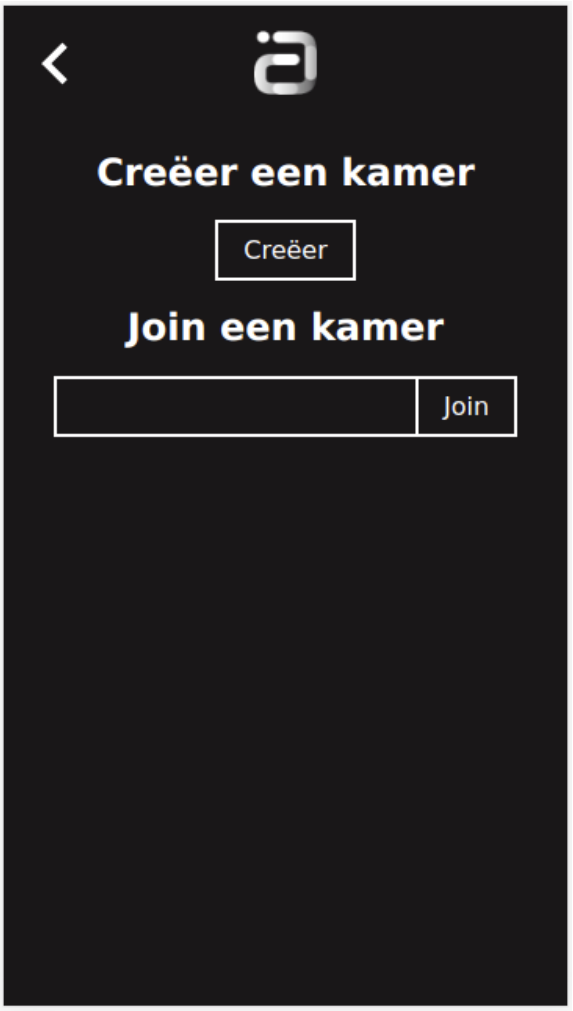
De applicatie biedt gebruikers de mogelijkheid om collega's te zoeken. Op het hoofdscherm kan de zoekbalk worden gebruikt om te zoeken op medewerkers. Wanneer op een zoekresultaat wordt geklikt navigeert de applicatie naar de profielpagina van de betreffende collega. Hier wordt extra informatie over de collega getoond.

Figuur 4.6-1 : Zoekfunctie hoofdscherm

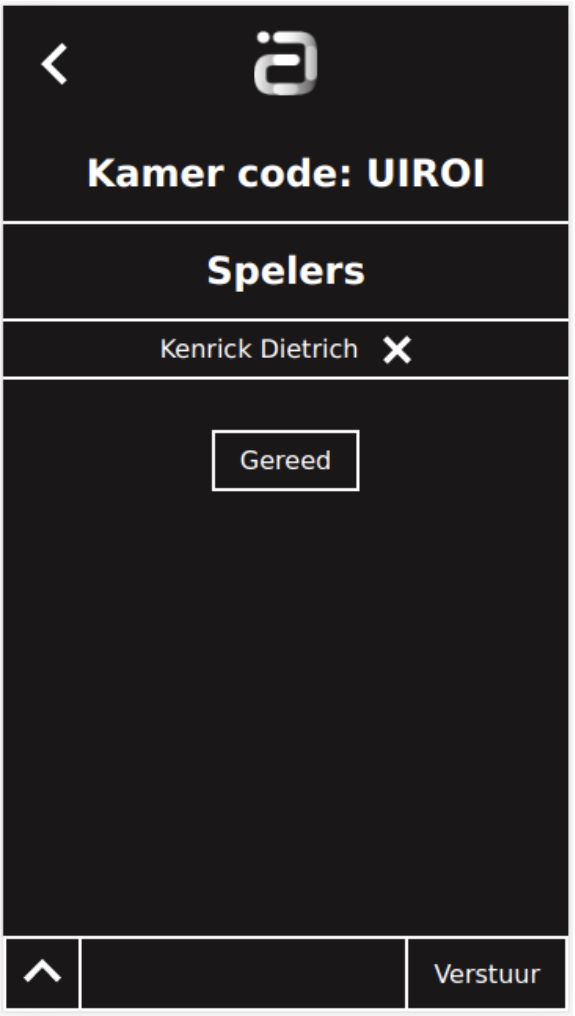
Figuur 4.6-2: Collega/profielpagina

4.7 - Multiplayer quiz spelen

De applicatie biedt de mogelijkheid om met meerdere mensen tegen elkaar de quiz te spelen. Vanuit het multiplayer beginscherm kan een kamer worden gemaakt of aan een kamer worden deelgenomen. Vervolgens wordt de gebruiker in een nieuwe kamer of bestaande kamer toegevoegd. In de kamer kan via de chatbox berichten worden gestuurd en aangegeven of je klaar bent om te beginnen met de quiz. De code moet gedeeld worden buiten de applicatie om.



Figuur 4.7-1 : Kamer deelnemen of aanmaken

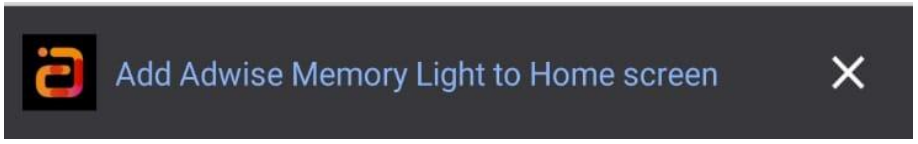


Figuur 4.7-2: Kamerscherm met alle spelers, code en chatbox

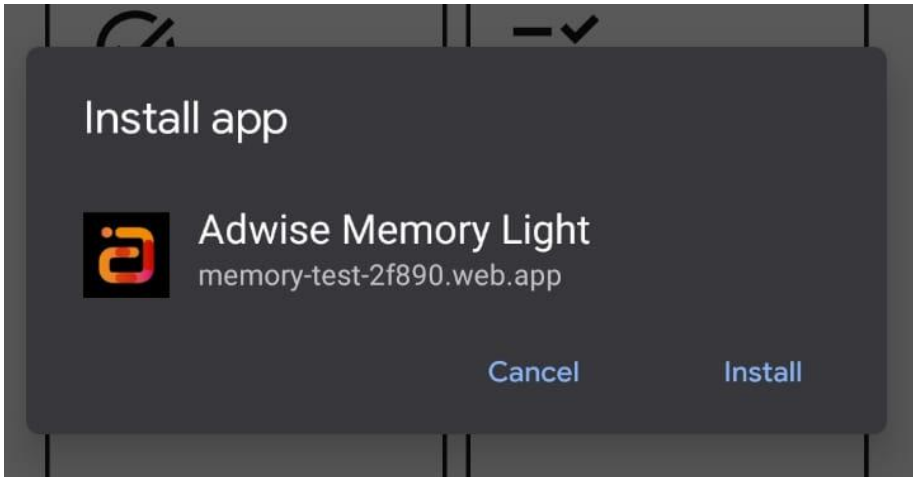
Wanneer alle spelers klaar zijn om te beginnen wordt de quiz gestart. Elke speler krijgt dezelfde vraag te zien waarbij de gebruiker het juiste antwoord moet selecteren net zoals figuur 4.4-2 . Wanneer alle spelers hebben geantwoord krijgen de spelers te zien of ze het goede antwoord hebben geselecteerd. Wanneer tien vragen zijn beantwoord keren de spelers weer terug naar de kamer waaruit een nieuw spel kan worden gestart of terug kan naar het hoofdscherm.

4.8 - PWA functies

Wanneer de gebruiker op de telefoon de applicatie via de browser bezoekt, heeft hij de mogelijkheid om de applicatie toe te voegen aan het hoofdscherm. De applicatie wordt vervolgens geïnstalleerd en kan als een native app worden gebruikt.



Figuur 4.8-1 : Installatie balk



Figuur 4.8-2 : Installatie popup

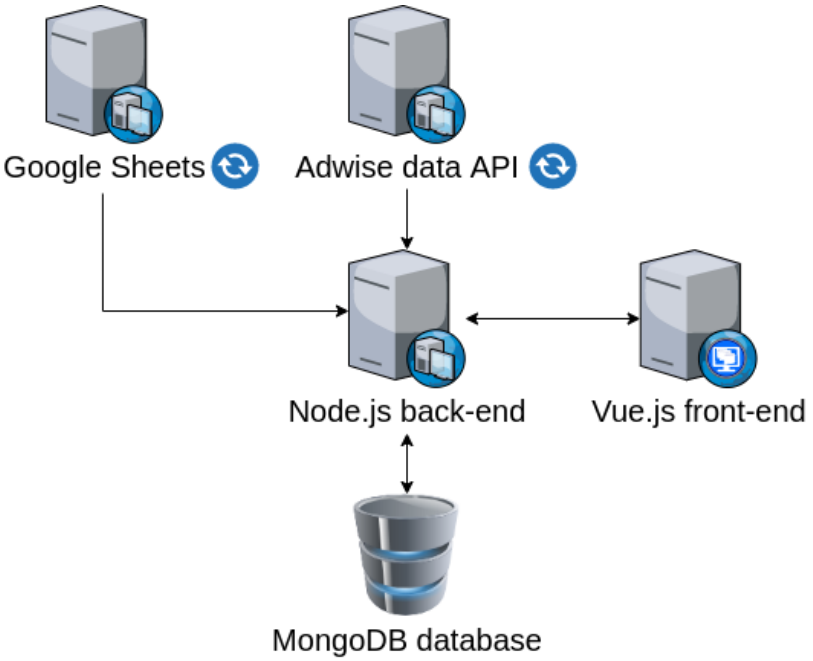
Figuur 4.8-3 : App icoon van geïnstalleerde applicatie

Wanneer de applicatie opstart met internetverbinding worden collega informatie, vragen en highscores opgehaald vanuit de back-end die vervolgens worden gecached. Wanneer de applicatie offline is wordt vanuit de cache de collega informatie, vragen en highscores geladen. Wanneer de internetverbinding is hersteld wordt data zoals vragen voor de quiz en recall per request opgehaald.

5 - Implementatie

5.1 - Architectuur

Voor de afstudeeropdracht zijn twee applicaties ontwikkeld, als onderdeel van een MEVN stack. MongoDB, Express.js, Vue.js en Node.js. Naast MEVN is er ook een MEAN en MERN stack, waarbij de back-end frameworks hetzelfde blijven maar het front-end framework veranderd. MEAN is met Angular en MERN is met React. Deze development stacks worden veel gebruikt voor full stack applicaties (Zahabia, 2021). Hieronder is een globaal overzicht weergeven van de verbinding tussen de applicaties en de API's. De back-end gebruikt data uit de Adwise API en combineert deze met extra collega informatie uit een Google Sheets bestand. Deze data wordt in de database opgeslagen. De data wordt vanuit de database in de back-end verwerkt tot vragen die naar de front-end worden gestuurd.



Figuur 5.1-1: Architectuur applicaties

5.1.1 - Back-end

De back-end server die voor de sockets is ontwikkeld is uitgebreid om efficiënt de data uit de Adwise API op te halen, te verwerken en in een database op te slaan. De data wordt vanuit de database verwerkt tot vragen en gestuurd naar de front-end. De data verwerken is in eerste instantie geïmplementeerd in de front-end, dit zorgde voor teveel verzoeken naar de Adwise API. Wanneer een enkele vraag voor de quiz of recall gegenereerd moest worden werd de volledige dataset doorlopen en bewerkt tot een vraag. Dit resulteerde in veel computatie in de browser waardoor de applicatie langzaam werd en elk request voor een vraag lang duurde. Naast de Adwise API is de mogelijkheid geïmplementeerd om een Google Sheets bestand te maken en extra informatie aan collega's te kunnen toevoegen. Google Sheets zorgt voor een snelle integratie met de back-end en een goede user interface voor een

gebruiker voor het toevoegen van extra informatie. Hierdoor is het mogelijk om snel deze functionaliteit te implementeren.

Wanneer de back-end opstart wordt data vanuit de Adwise API gesynchroniseerd en geupdate in de database. Hierna wordt een request naar het Google Sheets bestand gedaan die een bestand met extra collega informatie als CSV terugstuurd. Deze CSV data wordt omgezet naar JSON en vergelijke met de data in de database. Wanneer een id van de database overeenkomt met een id uit het Google Sheets bestand wordt de extra informatie aan de collega in de database toegevoegd.

De server wordt gestart vanuit het entry point “app.js”. Dit bestand start een express server waarbij de alle loaders uit de “loaders” map worden geïmporteerd en ingeladen. Deze loaders bestanden bevatten de configuratie voor Express, MongoDB, Mongoose en Socket.io.

Dependencies

Door middel van deskresearch is onderzocht hoe een Node.js back-end gestructureerd moet worden en welke dependencies nodig zijn om een moderne Node.js back-end te ontwikkelen die aansluit bij een MEVN stack. (Quinn, 2019)

De belangrijkste dependencies zijn:

- Express
- MongoDB
- Mongoose
- Nodemon

Express

Express is een back-end framework om een webserver te ontwikkelen voor Node.js. Deze webserver kan worden gebruikt om een webapplicaties en API’s te ontwikkelen. Express voorziet de backend van functies om HTTP request en URL routes te kunnen afhandelen. Wanneer een GET request naar een URL vanaf de front-end wordt verstuurd, ontvangt Express een verzoek en reageert op de methode. Afhankelijk van het request kan een response worden terug gestuurd. Express wordt veel gebruikt en is onderdeel van de OpenJS Fondation. OpenJS foundation is een organisatie voor het ondersteunen van grote en belangrijke projecten binnen het JavaScript ecosysteem.

MongoDB

MongoDB is een document georiënteerde database waarbij document in BSON worden opgeslagen (binair JSON). MongoDB is een NoSQL database dat betekent dat MongoDB geen relationele database is. Data wordt opgeslagen in collections, een collection bestaat uit meerdere documents. Een document is in dit geval een collega. MongoDB is gekozen als database doordat de data van collega’s geen relaties hebben tot andere data velden. Hierdoor is een relationele database niet nodig. Doordat MongoDB het JSON format aanhoudt is het makkelijk om snel integratie met de back-end te ontwikkelen.

Mongoose

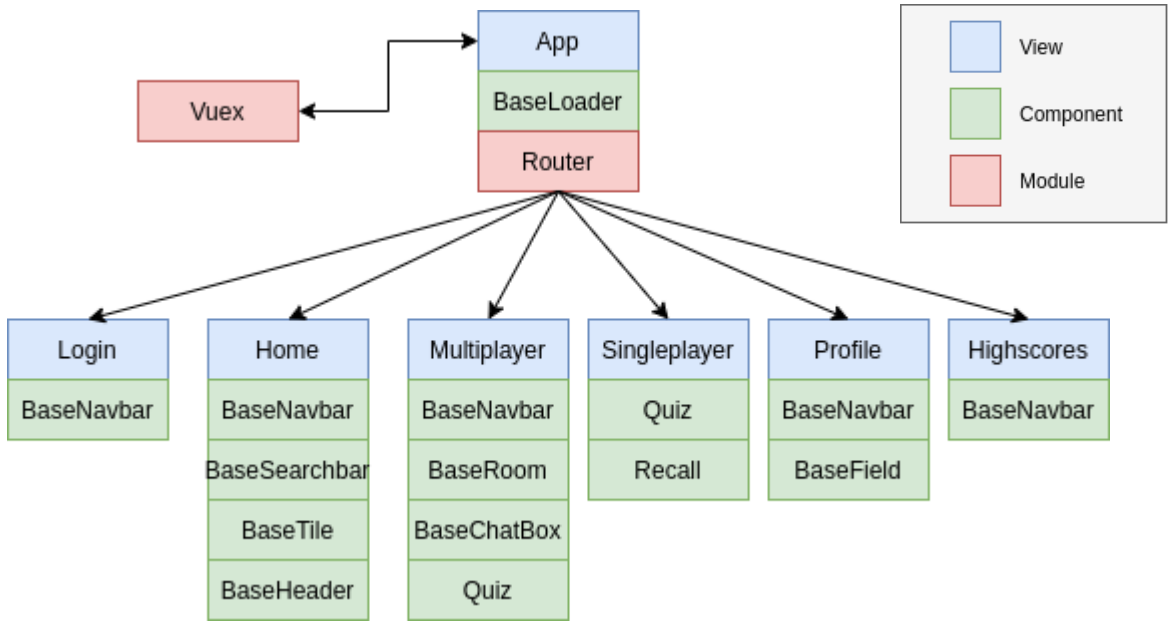
Mongoose is een ODM, een Object Document Mapping tool om queries te kunnen uitvoeren op de MongoDB database. Mongoose biedt ondersteuning voor methodes zoals find, skip, limit, sort en select. Deze queries kunnen met of zonder callback functie worden uitgevoerd. Mongoose geeft ook de mogelijkheid om schema's te definiëren. Deze schema's kunnen worden gezien als data models waarbij velden worden gedefinieerd. Daarnaast biedt een schema de mogelijkheid om geneste documenten te definiëren. Dit is normaliter niet mogelijk met MongoDB.

Nodemon

Nodemon is een tool die gebruikt wordt bij het ontwikkelen van de applicatie. Het wordt gebruikt om de applicatie te draaien en houdt bij of code is veranderd. Wanneer Nodemon veranderingen ziet, wordt de server automatisch opnieuw gestart met de nieuwe code.

5.1.2 - Front-end

De front-end is opgebouwd uit modules, views en components. Hierbij worden components hergebruikt in verschillende views om code redundantie te verminderen. Hieronder is de structuur van de applicatie weergeven.



Figuur 5.1.2-1: Structuur front-end

De applicatie begint vanuit de app view. Hier wordt een loader component gebruikt om de laadstatus van de applicatie te tonen. De router zorgt voor de navigatie van de applicatie. Hiermee wordt via de router aan de hand van de URL de verschillende views van de applicatie geladen. Elke view wordt dynamisch geïmporteerd. Hierdoor wordt de code alleen geladen wanneer de gebruiker de URL bezoekt. Voor elke navigatie naar een URL wordt gekeken of de gebruiker is ingelogd. Wanneer dit niet het geval is wordt de gebruiker doorverwezen naar de login view. Is de gebruiker wel ingelogd dan laadt de applicatie de

URL. Aan de applicatie wordt de Vuex store gebonden. Vuex is een state management module. Hierdoor kunnen alle views en componenten bij een centrale state van de applicatie.

Dependencies

De front-end bevat een aantal belangrijke dependencies voor het goed functioneren van de applicatie.

Vuex

Zoals eerder genoemd is Vuex een state management module. Een Vuex store houdt een patroon aan van een state, getters, mutations, actions en modules. State is een object waar alle data van de applicatie wordt bewaard. Met Vuex wordt deze state als “single source of truth” beschouwd. Dit betekend dat de data uit de Vuex state leidend is voor de views en components in de applicatie. Getters zijn functies waarmee data uit de state kan worden gehaald. In deze getter functies kan de data eerst worden gemanipuleerd voordat de functie een waarde terug stuurt naar een component of view. Mutations zijn functies die de waarde in de state aanpassen. Actions zijn functies die mutations aanroepen. In een action mag de state niet direct worden aangepast omdat actions vaak asynchroon zijn doordat een request naar een API wordt uitgevoerd. Wanneer het request voltooid is wordt de mutation aangeroepen. Modules kunnen worden gebruikt om de Vuex store in stukken te verdelen. Elke module heeft zijn eigen state, getters, functions, mutations en actions, die worden genest in de hoofd store. Hierdoor blijft de store overzichtelijk (What Is Vuex? | Vuex, z.d.).

Vue Router

De router is verantwoordelijk voor de navigatie tussen de verschillende views. De applicatie maakt gebruik van de “navigation guard”. Dit is een functie die wordt aangeroepen voor elke route wisseling. Deze functie zoals eerder genoemd, controleert of de gebruiker is ingelogd. Daarnaast controleert functie of de store al collega data heeft. Wanneer de store nog leeg is wordt een action aangeroepen om collega’s uit de back-end op te halen. Daarnaast biedt Vue router de mogelijkheid om parameters aan een URL mee te geven. Hiermee kan naar een profiel URL worden genavigeerd met een specifieke id. Wanneer de profielpagina laadt kan met het id de juiste collega informatie worden opgehaald uit de store.

Axios

Axios is een “promised-based” HTTP library. “Promised-based” betekent dat de requests asynchroon worden uitgevoerd waarbij de promise nadat hij klaar is, de “then” functie aanroept om het resultaat te verwerken. Met Axios kan de front-end HTTP requests naar de back-end sturen. Axios is voor zowel de back-end als front-end beschikbaar. Voor de front-end maakt Axios gebruik van de XMLHttpRequest API (Getting Started | Axios Docs, z.d.).

Vue-Socket.io

Vue-Socket.io is een Socket.io integratie voor Vue.js. Met deze library kan de socket.io-client worden gekoppeld aan de options API van Vue components. Dit betekend dat in het <script> blok van een Vue bestand een sockets object kan worden toegevoegd. In dit object zijn functies gedefinieerd die luisteren naar events die vanaf de server worden gestuurd. Daarnaast heeft de library integratie met Vuex waarmee vanuit de store geluisterd kan worden naar events.

5.2 - Inloggen met Google Workspace account

Om gebruik te maken van Google login moet via het Google Cloud platform een nieuw project worden gemaakt. Vervolgens moet de Analytics en People API worden geactiveerd. Daarnaast moet een OAuth credential worden gemaakt via het Google Cloud platform. Aan de credential moet worden gespecificeerd welke specifiek URL's gebruik mogen maken van de OAuth, hiermee krijgt de applicatie toegang tot het consent scherm. Omdat het project binnen de Adwise organisatie is gemaakt worden alleen accounts die tot de organisatie behoren geaccepteerd. Nadat het project met credential is aangemaakt kan de client ID van de credential worden gebruikt om vanuit de front-end een authenticatie verzoek te doen. Deze client ID is toegevoegd aan de environment variabele zodat deze voor de configuratie van de Google auth in de applicatie kan worden gebruikt.

Met de eerder genoemde navigation guard wordt de gebruiker doorverwezen naar het inlogscherm.

```
router.beforeEach((to, from, next) => {
  if (to.name !== 'Login' && !store.state.user) {
    next({ name: 'Login' })
  } else if (!store.state.colleagues.length > 0 && store.state.user) {
    store.dispatch('fetchData')
    next()
  } else {
    next()
  }
})
```

Figuur 5.2-1: Navigation guard

Wanneer de login pagina wordt geladen wordt in de mounted lifecycle hook van Vue de "initAuth" functie aangeroepen. De mounted lifecycle hook is een functie die wordt aangeroepen wanneer de Vue instantie is gekoppeld. In deze initAuth functie wordt de client API voor de Google login geladen. Vervolgens wordt de Auth2 instantie opgeslagen.

```
const initAuth = () => {
  store.dispatch('setLoading', true);
  // Wait for gapi to load the client
  gapi.load('client', {
    callback: () => {
      // Handle gapi.client initialization.
      gapi.client
        .init({
          apiKey: process.env.VUE_APP_GOOGLE_API_KEY,
          clientId: process.env.VUE_APP_GOOGLE_CLIENT_ID,
          scope: 'profile email',
          prompt: 'select_account',
        })
        .then(() => {
          GoogleAuth = gapi.auth2.getAuthInstance();
          // Listen for sign-in state changes.
          GoogleAuth.isSignedIn.listen(
            updateSigninStatus(GoogleAuth.isSignedIn)
          );
          store.dispatch('setLoading', false);
        });
    },
    onerror() {
      store.dispatch('setLoading', false);
    },
    timeout: 5000,
    ontimeout() {
      store.dispatch('setLoading', false);
    },
  });
};
```

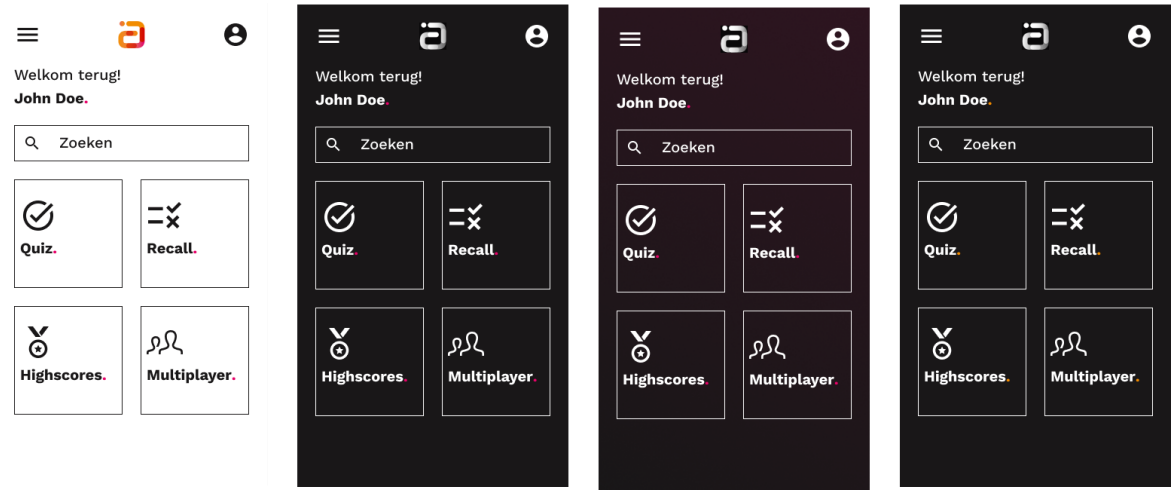
Figuur 5.2-2: Initialisatie van de Google auth client

Wanneer de gebruiker op de inlog knop klikt wordt het OAuth 2.0 process gestart. De applicatie roept de “handleAuthClick” functie aan waar wordt gekeken of de gebruiker is ingelogd of uitgelogd. Is de gebruiker ingelogd dan wordt de signIn functie van de Google auth client aangeroepen. Vervolgens wordt het consent scherm geopend waar de gebruiker zijn Adwise account kan selecteren.

De applicatie ontvangt na een succesvolle login, een user object die naar de back-end wordt gestuurd en op basis van het e-mailadres aan de betreffend collega toegevoegd. Daarnaast wordt een access token als cookie in de browser gezet. Wanneer de gebruiker de pagina herlaadt wordt de access token gebruikt om opnieuw te authenticeren. Is deze access token verlopen dan moet de gebruiker opnieuw inloggen.

5.3 - De applicatie voorzien van white-label support

Aan het begin van de ontwikkeling van de applicatie is een thema ontwerp gemaakt met het gebruik van Figma. Dit ontwerp houdt de huisstijl van Adwise aan die is afgeleid van het Adwise brand handboek en van de Adwise website. Deze designs zijn teruggekoppeld naar Elke Tiehuis en daarbij gaf zij aan de voorkeur te hebben voor het laatste ontwerp waarbij de accentkleur oranje is. Het Adwise roze wordt gebruikt voor extern gebruik en oranje voor intern gebruik, omdat deze applicatie bestemd is voor intern gebruik is gekozen voor het laatste ontwerp met de accentkleur oranje. Aan de hand van het thema ontwerp is de applicatie ontwikkeld.



Figuur 5.3-1 : Figma thema ontwerp

Voor de styling van de applicatie is gebruik gemaakt van Sass wat een afkorting is voor Syntactically Awesome Style Sheets. Dit is een extensie van de standaard CSS, Cascading Style Sheets. Sass voegt een aantal extra functionaliteiten toe boven op die van CSS. De belangrijkste voor whitelabel styling zijn variabelen, modules en functions.

```
$bg: #fff;
$primary: #191718;
$secondary: #f28c00;
$tetriary: #cecacc;
```

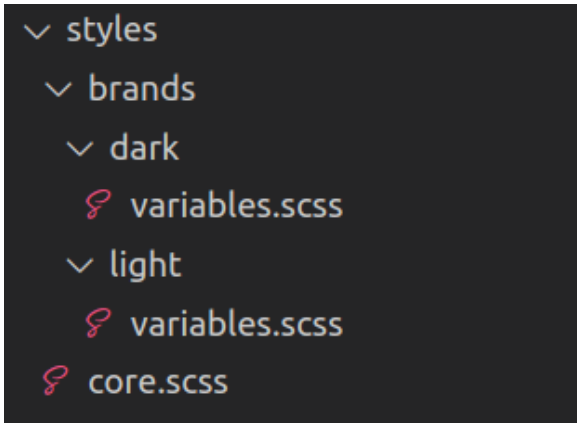
Variabelen kunnen in een .scss bestand worden gedefinieerd en gebruikt door de gehele applicatie. Dit zorgt voor consistentie van de styling doordat altijd dezelfde variabelen worden gebruikt en een plek waar de styling voor de gehele applicatie kan worden aangepast.

Figuur 5.3-2: Sass variabelen

```
@import '~brandVariables/variables.scss';
```

Figuur 5.3-3: Brand specifieke Sass module import

Modules maken het mogelijk om styling en variabelen te scheiden in verschillende bestanden. Deze bestanden kunnen vervolgens in andere .scss worden geïmporteerd en styling toevoegen of overschrijven.



Figuur 5.3-4: Sass file structuur

Met deze variabelen en modules is een structuur gemaakt waarbij de core.scss de basis styling van de applicatie bevat en extra styling wordt toegevoegd met een import van een brand specifieke scss bestand.

```
@function url-color($color) {
  @return '%23' + str-slice('#{$color}', 2, -1);
}
```

Figuur 5.3-5: Sass functie om kleur codes te genereren

Deze functie zorgt dat een Sass variabele kan worden omgezet tot een string met een hexadecimale kleur. Hiermee kunnen SVGs van kleur worden veranderd.

Om gebruik te maken van Sass moet Vue en Webpack geconfigureerd worden.

Webpack is een module bundler die een “dependency graph” maakt van elke module die de applicatie gebruikt. Deze dependency graph kan worden gezien als een boomstructuur vanaf het entry bestand en tot alle modules die van elkaar afhankelijk zijn. Wanneer de applicatie voor productie wordt gebouwd verpakt Webpack de applicatie in een of meerdere bundels van JavaScript, CSS en andere statische bestanden zoals afbeeldingen. De browser kan deze bundels begrijpen en laden.

De standaardconfiguratie van Webpack kan Sass bestanden niet bundelen tot modules en heeft extra configuratie nodig. Webpack kan worden uitgebreid door middel van een loader toe te voegen. Voor Sass is een sass-loader ontwikkeld, deze loader zet Sass om naar CSS zodat Webpack de CSS kan bundelen. Webpack kan worden geconfigureerd door een vue.config.js bestand in de root van het project te maken. Vue beschouwd een vue.config.js bestand als optionele configuratie en wordt automatisch geladen door de CLI service. In dit bestand is de sass-loader toegevoegd en is het core.scss bestand met de basis styling als entry file gedefinieerd.

Om Webpack te laten begrijpen wanneer welke Sass variabelen gebruikt moeten worden is een alias gedefinieerd. Deze alias wordt gebruikt bij het importeren van brand specifieke variabelen. De alias wordt opgesteld uit het white-label config bestand waarbij aan de hand van een environment variabele het pad wisselt.

```
const path = require('path');
const whitelabelConfig = require('./whitelabel.config');
const brandVariablesPath =
  whitelabelConfig[process.env.VUE_APP_BRAND].scssVariables;

module.exports = {
  configureWebpack: {
    resolve: {
      alias: {
        brandVariables: path.resolve(__dirname, brandVariablesPath),
        fonts: '@/assets/fonts',
      },
    },
  },
  css: {
    loaderOptions: {
      sass: {
        prependData: `
              @import "@/styles/core.scss";
            `,
      },
    },
  },
};
```

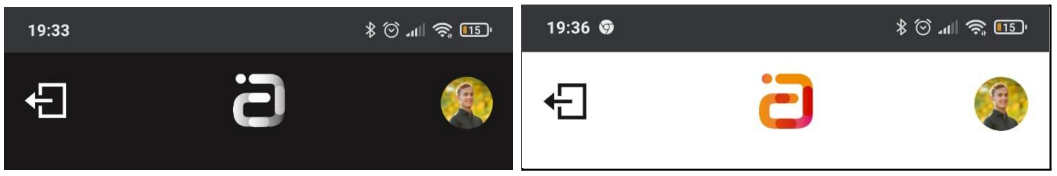
Figuur 5.3-6: vue.config.js

Wanneer als environment variabele brand light wordt meegegeven zullen de Sass light variabelen worden gebruikt en gebundeld door Webpack. Een brand style kan makkelijk worden toegevoegd door een scss bestand met variabelen aan te maken, deze in de whitelabelConfig te definiëren en vervolgens als environment variabele de brand naam mee te geven.

Logo's en plaatjes in de applicatie kunnen worden gewisseld door gebruik te maken van een computed functie. Een computed functie is reactief en wordt aangeroepen wanneer de data uit de functie veranderd. In deze functie wordt het logo geïmporteerd voor het specifieke app brand naam en de naam van het logo uit het white-label configuratie bestand.

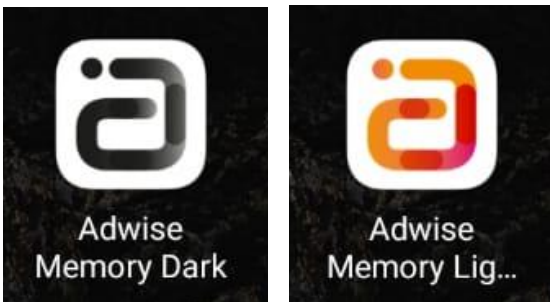
```
<script>
export default {
  computed: {
    logo() {
      return require(`../styles/brands/${process.env.VUE_APP_BRAND}/${this.$whitelabel.icon}`);
    },
  },
};
</script>
```

Figuur 5.3-7 : Computed require functie voor white-label logo



Figuur 5.3-8 : Verschillende logo's in navbar

Door de PWA functionaliteiten kan de applicatie worden geïnstalleerd. Met de white-label support kan ook het app icoon, naam en thema kleuren van het startscherm te veranderen. Deze variabelen zijn in te stellen in de vue.config.js en op basis van de white-label variabelen veranderd. Deze variabelen overschrijven het "manifest" bestand. Dit is een JSON bestand waarin alle informatie staat die wordt gebruikt bij het installeren van de applicatie.



Figuur 5.3-9 : App pictogrammen met verschillende styling en naam.

Voor het wisselen van app pictogrammen is gebruik gemaakt van "file-loader". Dit is Webpack module die het mogelijk maak om bestanden die niet in de applicatie worden geïmporteerd, mee te bundelen in de productie folder. Het eerder genoemde manifest bestand gebruikt de pictogrammen voor de verschillende apparaten en besturingssystemen. De module is geconfigureerd om pictogrammen aan de hand van de white-label variabelen mee te bundelen.

5.4 - Quiz en recall spelen

Voor de implementatie van de spellen quiz en recall moet de medewerker informatie efficiënt worden verwerkt tot vragen en antwoorden. Voor zowel het quiz spel als recall moeten vragen worden gegenereerd op basis van informatie over collega's. Deze informatie over collega's wordt uit een API opgehaald die door Adwise wordt beheert en bijgewerkt met werknemer informatie.

Genereren vragen en antwoorden

Vanaf het quiz scherm in front-end wordt een dispatch naar de "fetchQuestion" action gedaan in de Vuex store. Deze action doet een GET /question request naar de back-end met Axios.

Dit request wordt in de back-end ontvangen waarna een createQuestion functie wordt aangeroepen. De createQuestion functie genereert vervolgens willekeurig een vraag, een willekeurige collega en genereert op basis van de vraag en collega de antwoorden.

Elke vraag bevat een titel en "answerType". Deze answerTypes zijn velden van een collega object. Nadat een willekeurige vraag is gekozen is word gekeken of de willekeurige collega het betreffende "answerType" als veld heeft. Wanneer dit veld niet bestaat of leeg is wordt een andere collega recursief gezocht.

```
// Get random colleague of the list of colleagues
const getRandomColleague = async (question) => {
  // Get the count of colleague where answer type exists
  let count = await Colleague.countDocuments()
    .where(question.answerType)
    .exists()
    .exec()
  // Get a random entry
  let random = Math.floor(Math.random() * count)
  // Find colleagues where answer type exists and field is not an empty string
  let colleagues = await Colleague.find().where(question.answerType).exists()

  // If the value of the answerType field is empty or null get a different random colleague
  if (
    colleagues[random][question.answerType] === '' ||
    !colleagues[random][question.answerType]
  ) {
    return getRandomColleague(question)
  } else {
    return colleagues[random]
  }
}
```

Figuur 5.4-2: getRandomColleague functie

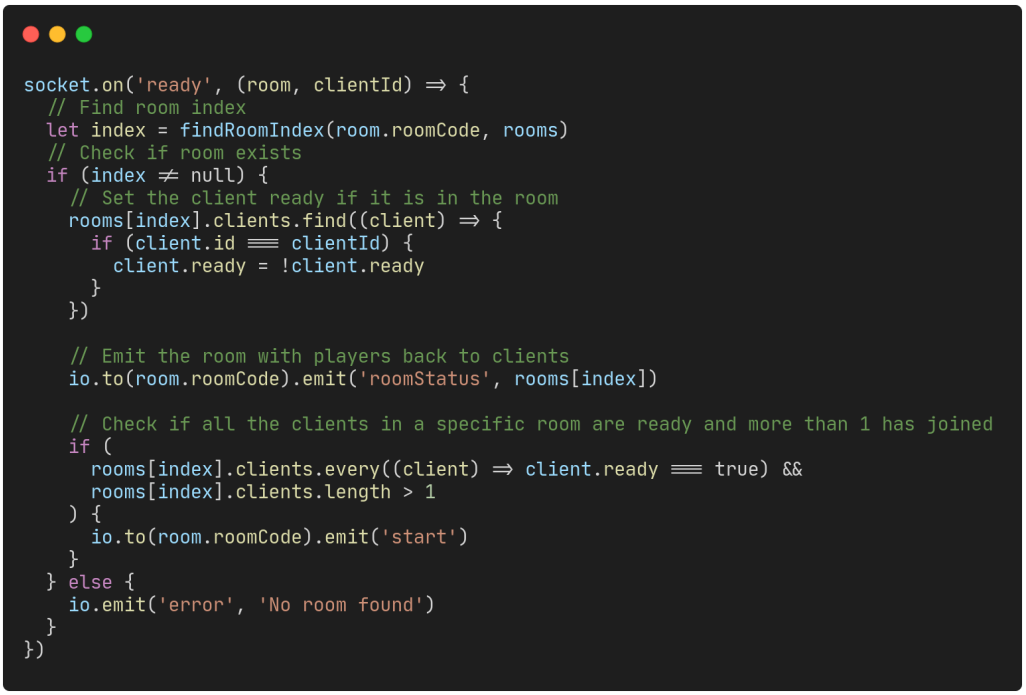
Wanneer de vraag en antwoorden zijn gegenereerd worden deze terug gestuurd naar de front-end. Vervolgens word de data uit het request in de Vuex store gezet. De data wordt vanuit de store met een computed functie opgehaald en getoond in de view.

5.5 - Multiplayer quiz spelen

Wanner de back-end opstart word de socket.io loader geladen. Dit is een bestand waar een server wordt gestart waaraan het io object wordt gekoppeld. Wanneer een client verbindt met de server wordt per verbinding een socket opgezet tussen de client en server. Wanneer de client een event emit, ontvangt de server het event en stuurt een event terug. De server luistert naar de events:

- Create: het aanmaken van een kamer
- Join: het deelnemen van een kamer
- Message: een bericht sturen naar de kamer
- Ready: een speler op ready zetten in een kamer
- FetchQuestion: een vraag genereren en terugsturen
- clientAnswered: zet een client uit een kamer op geantwoord en controleer of iedereen heeft geantwoord

Wanneer een gebruiker een kamer aanmaakt wordt het create event naar de server gestuurd. De server maakt een kamer aan met een unieke code en stuurt deze terug naar de client. Vervolgens slaat de client de kamer op in de store. Klikt de gebruiker op deelnemen dan stuurt de client het join event. De server controleert of een kamer met code bestaat en voegt de client toe aan de kamer. De gehele kamer krijgt vervolgens een update van kamer. Alle clients ontvangen het event en updaten de kamer in de store. Via deze manier van events emitten via de sockets van clients naar server wordt de communicatie geregeld.



```
socket.on('ready', (room, clientId) => {
  // Find room index
  let index = findRoomIndex(room.roomCode, rooms)
  // Check if room exists
  if (index !== null) {
    // Set the client ready if it is in the room
    rooms[index].clients.find((client) => {
      if (client.id === clientId) {
        client.ready = !client.ready
      }
    })

    // Emit the room with players back to clients
    io.to(room.roomCode).emit('roomStatus', rooms[index])

    // Check if all the clients in a specific room are ready and more than 1 has joined
    if (
      rooms[index].clients.every((client) => client.ready === true) &&
      rooms[index].clients.length > 1
    ) {
      io.to(room.roomCode).emit('start')
    }
  } else {
    io.emit('error', 'No room found')
  }
})
```

Figuur 5.5-1: Ready event server

Hierboven is het ready event van de server getoond. Hierbij wordt gecontroleerd of de kamer bestaat en of de client in de kamer aanwezig is. Wanneer dit beide het geval is wordt de client op ready of niet ready gezet en wordt een update naar de gehele kamer gestuurd. Zijn alle clients ready in de kamer dan wordt het start event gestuurd en wordt de quiz gestart.

De client importeerd de socket.io bibliotheek en definieerd de URL van de server. Wanneer de multiplayer view wordt geladen wordt een verbinding met de server opgezet. In de options API van Vue zijn de opties “sockets” en “methods” gedefinieerd. In het sockets object worden de events vanuit de server ontvangen en kan in de functie de benodigde functionaliteit worden geschreven. Bijvoorbeeld het ontvangen van berichten. In de message functie wordt het event message ontvangen van de server. Het bericht wordt toegevoegd aan de lijst van berichten en de chatbox wordt naar beneden gescrold.

```
<script>
import Vue from 'vue';
import VueSocketIOExt from 'vue-socket.io-extended';
import { io } from 'socket.io-client';

const socket = io(process.env.VUE_APP_SOCKET_SERVER);

Vue.use(VueSocketIOExt, socket);

export default {
  name: 'Multiplayer',
  components: {
  sockets: {
    message(message) {
      this.messages.push(message);
      Vue.nextTick(() => {
        let ul = document.getElementById('chat-list');
        ul.scrollTop = ul.scrollHeight;
      });
    },
  },
  methods: {
    send(value) {
      this.$socket.client.emit('message', value, this.room);
    },
  },
};
</script>
```

Figuur 5.5-2: Client socket io message implementatie

5.6 - PWA functies

Het implementeren van PWA ondersteuning kan door middel van de PWA plugin toe te voegen aan het Vue.js project. Dit is mogelijk door het via de Vue CLI te selecteren of aan een bestaand project toe te voegen. Vue CLI is een tool voor het genereren van Vue projecten. Met het toevoegen de plugin wordt Workbox aan het project toegevoegd. Workbox is een library voor het genereren van een service worker. Deze service worker is verantwoordelijk voor de functionaliteiten een PWA. Wanneer de applicatie wordt gebouwd voor productie genereert Workbox een service worker. Met deze service worker kan de applicatie worden geïnstalleerd.

Om Caching in te schakelen kan in het vue.config.js bestand de workbox opties worden aangepast. Met deze optie geconfigureerd wordt gekeken naar de URL van het request. Wanneer het request overeenkomt met de API URL wordt er gecontroleerd of de applicatie internetverbinding heeft. Heeft de applicatie geen internet dat zal de service worker proberen het request op de cache uit te voeren.

```
pwa: {
    workboxOptions: {
      runtimeCaching: [
        {
          urlPattern: new RegExp(process.env.VUE_APP_API_URL),
          handler: 'NetworkFirst',
          options: {
            networkTimeoutSeconds: 20,
            cacheName: 'api-cache',
            cacheableResponse: {
              statuses: [0, 200],
            },
          },
        },
      ],
    },
  },
```

Figuur 5.6-1: Workbox optie

5.7 - Google Cast integratie

Voor de Google Cast integratie is geprobeerd een Custom Web Receiver te ontwikkelen echter wegens tijdnood is dit niet gelukt. Tijdens het implementeren van Google Cast kwamen een aantal beperkingen aan het licht. Deze beperkingen komen doordat een Chromecast voornamelijk bedoeld is voor het casten van video en audio.

Door hardware limitaties van de Chromecast wordt tijdens het casten van een HTML pagina, het beeld vastgezet op een resolutie van 1280x720. Dit zorgt voor een onscherp beeld op een grote tv (Google Cast Receiver App screen resolution, 2017).

Het laden van een HTML met een Custom receiver kan op twee manieren. De URL van de applicatie kan worden geladen in een iframe. Een iframe HTML tag kan worden gebruikt om een HTML document in een HTML document te laden. Hiermee kan de receiver in contact blijven met de Chromecast en ook wisselen van HTML document. De andere mogelijkheid is de locatie van de window van Chromecast te wisselen met een andere URL. Echter verliest de de Chromecast hierdoor de verbinding met de het cast apparaat en kan niet meer de URL worden veranderd totdat het casten wordt gestopt en opnieuw wordt gestart. Met beide mogelijkheden kon de URL van de applicatie worden gecast naar de Chromecast (DeMille, 2015).

De applicatie controleert of de gebruiker is ingelogd wanneer de applicatie wordt geladen. Dit zorgde voor een nieuw probleem met het casten van de applicatie. Wanneer de URL werd geladen via de Chromecast werd de inlog pagina getoond doordat er geen gebruiker was ingelogd. Naast het inloggen was het ook niet mogelijk de juiste kamer te kunnen casten doordat de kamer niet wordt gespecificeerd via de URL maar met sockets.

Een mogelijke oplossing die nog niet verder is onderzocht is een URL casten die naar de back-end verwijst waarbij met het gebruik van Express een HTML pagina wordt gegenereerd die de vraag toont. Met deze oplossing zijn een aantal mogelijk problemen. De HTML pagina moet gelijk zijn aan de white-label configuratie in de front-end. Wanneer een gebruiker vanuit de front-end op de cast knop klikt moet een verificatie code en de juiste room worden meegegeven aan de URL die wordt gecast zodat de back-end kan controleren of het request valide is. Hiermee wordt de collega informatie beschermd en komt de vraag overeen met de kamer waarin de speler die het spel cast zich bevindt.

6 - Werkmethode

6.1 - Projectmanagement

6.1.1 - Kanban

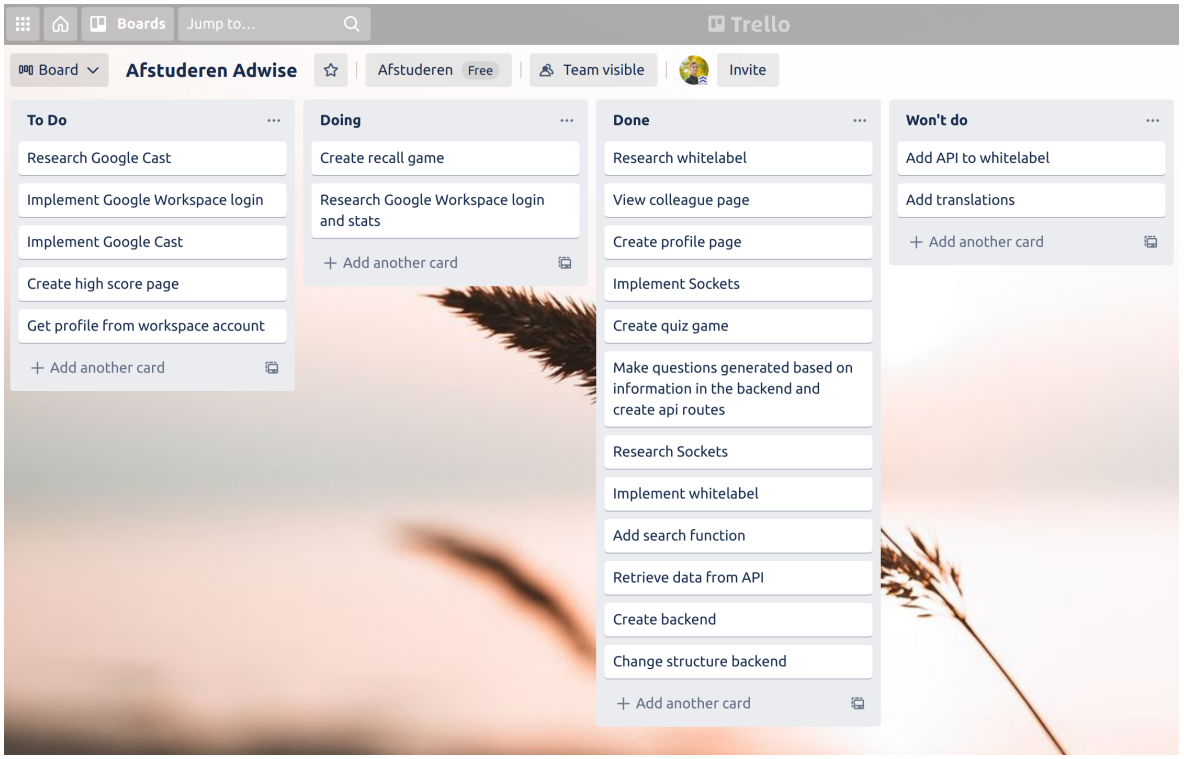
Gedurende het project is de projectmanagementmethode Kanban gehanteerd. De keuze voor Kanban in plaats van Scrum is door de individuele aard van de opdracht. Scrum heeft drie duidelijk gedefinieerde rollen waarbij Kanban geen vaste rollen heeft. Kanban is door het gebrek aan rollen een beter geschikte methodiek voor individueel projectmanagement.

Kanban heeft geen blokken van sprints maar een lang ontwikkelproces waarbij wordt gewerkt volgens prioriteit. Hierbij heeft het bovenste item op het Kanbanbord de meeste prioriteit. Een lang ontwikkelproces heeft als gevolg dat er geen planning, review, retrospective en stand-up plaatsvindt. Dit is een voordeel gedurende het project omdat deze Scrum activiteiten lastig zijn uit te voeren als individu.

Wekelijks is een voortgangsgesprek met de bedrijfsbegeleider gevoerd om de huidige voortgang te bespreken. Wanneer dringende vragen of hulp nodig was, kon dit eerder worden aangekaart door middel van een Slack bericht of een videogesprek in te plannen.

6.1.2 - Trello

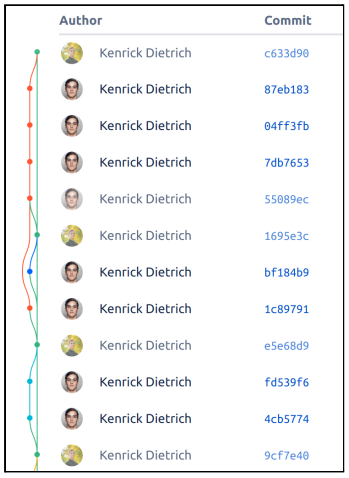
Om het project agile te managen was eerst de bedoeling om gebruik te maken van Jira Software. Dit is een tool die developers helpt met het brengen van overzicht en structuur van een project. Binnen Jira Software kan een Kanbanbord worden aangemaakt waaraan issues toegevoegd kunnen worden. Dit is veranderd doordat voor het gebruik van Jira Software extra rechten worden toegevoegd aan het Bitbucket account. Dit zou extra kosten met zich mee brengen. Hierdoor is gekozen om Trello te gebruiken, een gratis alternatief. Naast de kosten waren de extra functionaliteiten van Jira Software, zoals het koppelen van issues op het kanban bord aan branches door de individuele aard van de opdracht niet noodzakelijk.



Figuur 6.1.2-1: Trello bord

6.1.3 - Bitbucket

Binnen Adwise wordt gebruik gemaakt van Bitbucket voor de opslag van code. Doordat de applicatie intern is ontwikkeld is de source code voor de applicatie gehost op Bitbucket. Op Bitbucket is een repository aangemaakt voor de front-end en back end code. Voor beide applicaties is gewerkt door voor een issue, een nieuwe branch van de master branch te maken. Wanneer de nieuwe branch klaar was, werd deze via een pull request met de master branch samengevoegd.



Figuur 6.1.3-1: Bitbucket source tree

7 - Conclusie

Het doel van het afstudeerproject was het ontwikkelen van een applicatie waarmee de onboarding van een nieuwe medewerker kan worden verbeterd door het leren kennen van collega's makkelijker te maken.

Tijdens het onderzoek is onderzocht hoe mensen leren, hoe deze informatie het best wordt opgeslagen, Welke andere oplossingen al bestaan, Wat medewerkers vinden van het leren kennen van nieuwe collega's, Welke technologie geschikt is om de applicatie mee te ontwikkelen, Hoe multiplayer kan worden geïmplementeerd, hoe de applicatie white-label kan worden ontwikkeld en hoe Google Cast kan worden gebruikt om spel informatie te kunnen casten.

Met de ondervindingen uit dit onderzoek is een applicatie ontwikkeld waarbij medewerkers kunnen inloggen met een Adwise account, spellen kunnen spelen om informatie over collega's te leren, deze spellen samen spelen met de multiplayer modus en collega's kunnen opzoeken. De applicatie kan worden gewisseld van styling om in de toekomst ingezet te kunnen worden bij klanten van Adwise. Daarnaast kan de applicatie gebruikt worden op de desktop en als native ervaring op de telefoon.

8 - Discussie

Terugblikkend op de requirements zijn bijna alle requirements bereikt en daarmee een bijna volledig product. De requirements die niet zijn geïmplementeerd zijn:

- Chromecast integratie
- Level omhoog gaan met genoeg punten

De Google Cast functionaliteit is voorgesteld door de tv's met Chromecasts die op het kantoor van Adwise aanwezig zijn. Hiermee zou de multiplayer modus in de toekomst in een gezelschap kunnen worden gespeeld op het kantoor. Echter is deze functionaliteit niet geïmplementeerd door tijdnood, de structuur van de applicatie en de limitaties van de hardware. Om deze functionaliteit werkend te krijgen zal moeten worden onderzocht of de voorgestelde implementatie functioneel is (Hoofdstuk 5.7).

Het level omhoog gaan was een Should requirement en daarmee een lagere prioriteit. Echter is deze functionaliteit snel te implementeren. Doordat het puntensysteem werkt.

Mocht de applicatie in de toekomst verder worden ontwikkeld dan zou ik de volgende aanbevelingen doen:

- Een level systeem implementeren aan de hand van de punten
- Onderzoeken of Google Cast voorgestelde implementatie functioneel is
- Het toevoegen van extra collega informatie verplaatsen naar de applicatie zodat het gebruik en beheer op een plek is.
- Het mogelijk maken om via de applicatie een uitnodiging voor een kamer te kunnen sturen met bijvoorbeeld een link.

9 - Reflectie

In het begin van het afstudeerproject was veel nog onduidelijk over hoe de applicatie er precies ging uitzien en over welke functionaliteiten het moest beschikken. Hierdoor ging veel tijd verloren in het goed bedenken en vastleggen wat ontwikkeld moest worden. Bijvoorbeeld de back-end. In eerste instantie werd gedacht Firebase te kunnen gebruiken om meer focus te kunnen leggen op het ontwikkelen van de front-end. Door het onderzoeken van multiplayer functionaliteit kwam ik tot de conclusie dat een webserver voor sockets moest worden ontwikkeld. Hierdoor kwam ik tot het besluit om de webserver uit te breiden tot een volledige back-end en veranderde de opdracht van het ontwikkelen van een front-end tot een full-stack applicatie. Dit zorgde voor meer onderzoek dan voorzien. Door een beter beeld te vormen voorafgaand van de opdracht kan een betere inschatting worden gemaakt van de benodigde tijd en ervaring. Ondanks het niet kunnen integreren van Chromecast functionaliteit ben ik tevreden met de ontwikkelde producten.

10 - Literatuurlijst

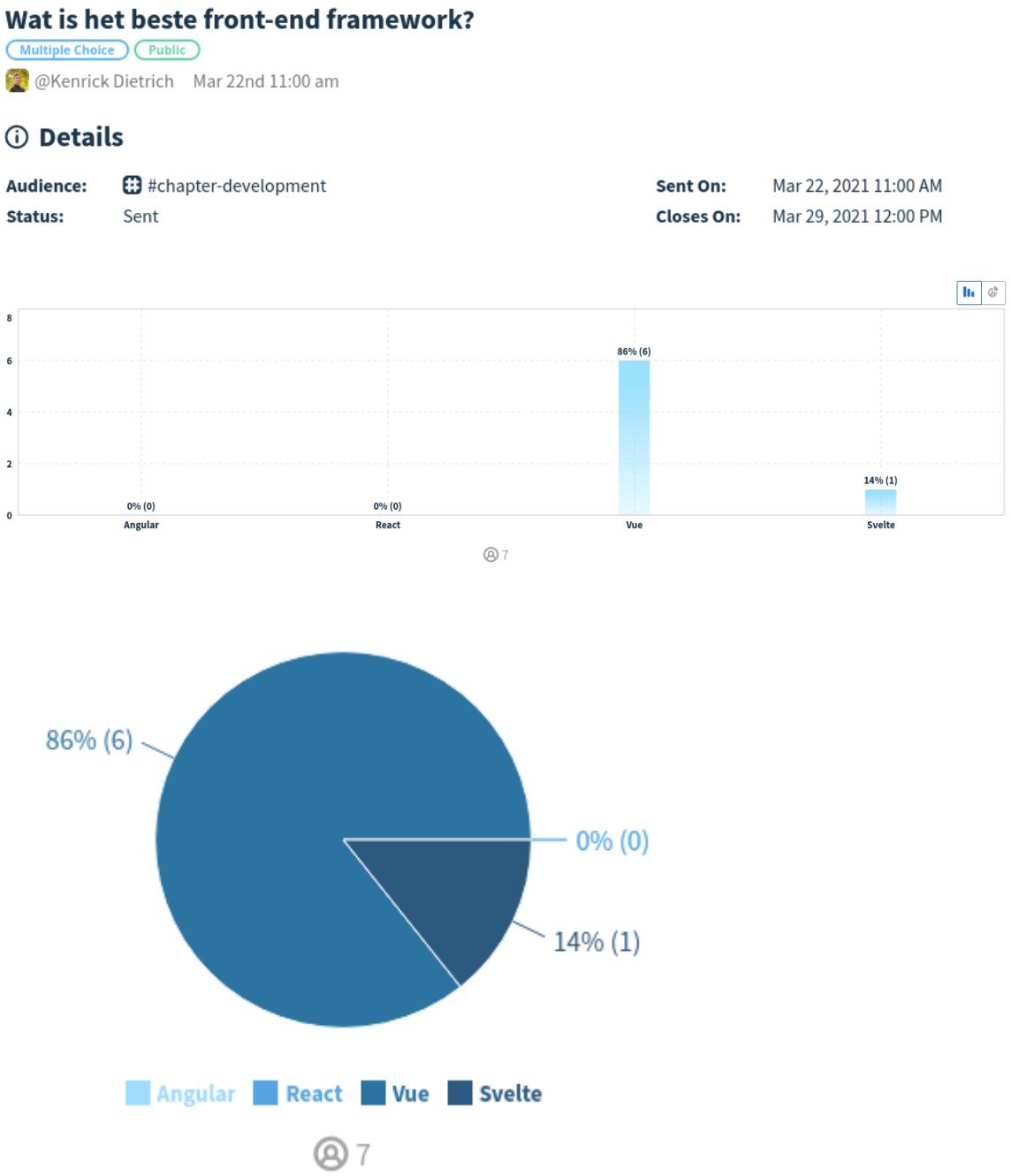
1. Dietrich, K. R. (2021). User stories & requirements
2. Cast. (z.d.). Google Developers. Geraadpleegd op 11 februari 2021, van https://developers.google.com/cast
3. Google Identity. (z.d.). Google Developers. Geraadpleegd op 11 februari 2021, van https://developers.google.com/identity
4. Atlassian. (z.d.). Kanban vs Scrum. Geraadpleegd op 12 februari 2021, van https://www.atlassian.com/agile/kanban/kanban-vs-scrum
5. Dirksen, J. (2011). Design for how People Learn. New Riders. https://ptgmedia.pearsoncmg.com/images/9780321768438/samplepages/0321768434.pdf
6. Knowingo B.V. (z.d.). Knowingo. Knowingo. Geraadpleegd op 8 maart 2021, van https://knowingo.com/
7. Incogny - Know your friends. (z.d.). Icogny. Geraadpleegd op 8 maart 2021, van https://incogny.com/
8. Saring, J. (2020, 18 november). 8 Node.js Web Socket Libraries for 2019 - Bits and Pieces. Medium. https://blog.bitsrc.io/8-node-js-web-socket-libraries-for-2018-818e7e5b67cf
9. Srivastava, V. (2020, 31 oktober). 10 Best WebSocket Libraries That You Should Know. Geeky Humans. https://geekyhumans.com/websocket-libraries/
10. Sockjs. (z.d.). sockjs/sockjs-node. GitHub. Geraadpleegd op 30 mei 2021, van https://github.com/sockjs/sockjs-node
11. Ws. (z.d.). websockets/ws. GitHub. Geraadpleegd op 30 mei 2021, van https://github.com/websockets/ws
12. Socket.io. (2021, 25 mei). Introduction. https://socket.io/docs/v4/
13. Wikipedia. (2020, 17 juni). WebSocket. https://nl.wikipedia.org/wiki/WebSocket
14. Wikipedia. (2021, 13 mei). Transmission Control Protocol. https://nl.wikipedia.org/wiki/Transmission_Control_Protocol
15. Wikipedia. (2021a, januari 11). Kwartet (kaartspel). https://nl.wikipedia.org/wiki/Kwartet_(kaartspel)
16. Wikipedia. (2019, 18 december). Memory. https://nl.wikipedia.org/wiki/Memory
17. Vuejs-templates. (z.d.). vuejs-templates/pwa. GitHub. Geraadpleegd op 30 mei 2021, van https://github.com/vuejs-templates/pwa/blob/master/docs/static.md
18. Quinn, S. (2019, 17 april). Bulletproof node.js project architecture. Softwareontheroad: Articles on Node, React, Angular, AWS.

https://softwareontheroad.com/ideal-nodejs-project-structure/?utm_source=github&utm_medium=readme

19. Wikipedia. (2021, maart 17). Express.js. https://en.wikipedia.org/wiki/Express.js
20. What is Vuex? | Vuex. (z.d.). Https://Vuex.Vuejs.Org/. Geraadpleegd op 12 juni 2021, van https://vuex.vuejs.org/
21. Getting Started | Axios Docs. (z.d.). Https://Axios-Http.Com/Docs/Intro. Geraadpleegd op 12 juni 2021, van https://axios-http.com/docs/intro
22. Wikipedia. (z.d.). Google Cast. Wikipedia. Geraadpleegd op 12 juni 2021, van https://en.wikipedia.org/wiki/Google_Cast
23. Web Receiver Overview | Cast |. (z.d.). Google Developers. Geraadpleegd op 13 juni 2021, van https://developers.google.com/cast/docs/web_receiver
24. Google Cast Receiver App screen resolution. (2017, 7 september). Stack Overflow. https://stackoverflow.com/questions/46102121/google-cast-receiver-app-screen-resolution
25. DeMille. (2015). DeMille/url-cast-receiver. GitHub. https://github.com/DeMille/url-cast-receiver
26. Zahabia. (2021, 8 februari). MEAN VS MERN Stack: Who will win the war in 2021? GKMIT. https://www.gkmit.co/blog/web-app-development/mean-vs-mern-stack-who-will-win-the-war-in-2021

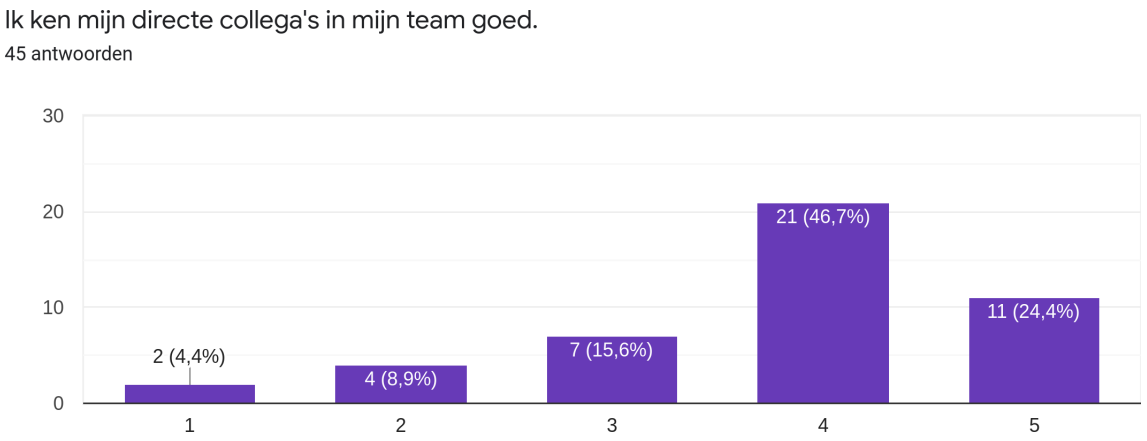
11 - Bijlage

11.1 - Poll frameworks

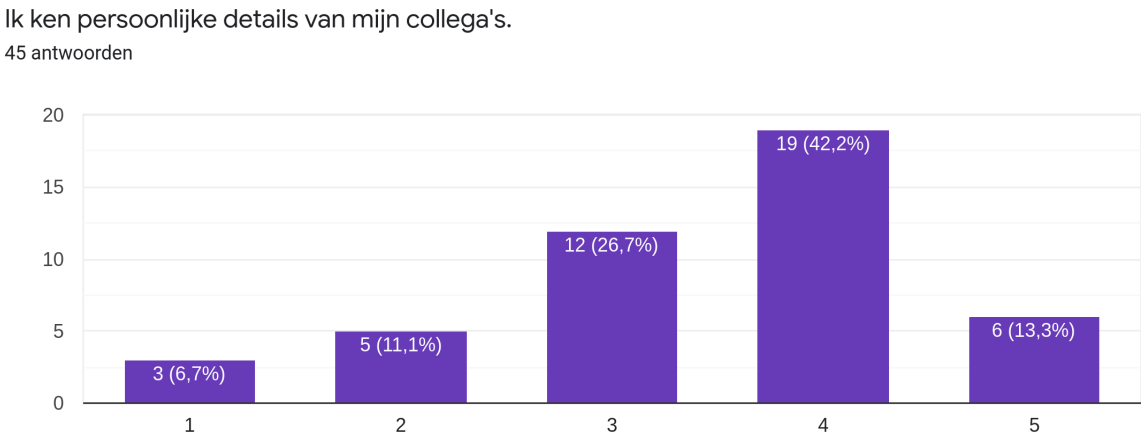


Figuur 11.1: Chapter development poll beste framework

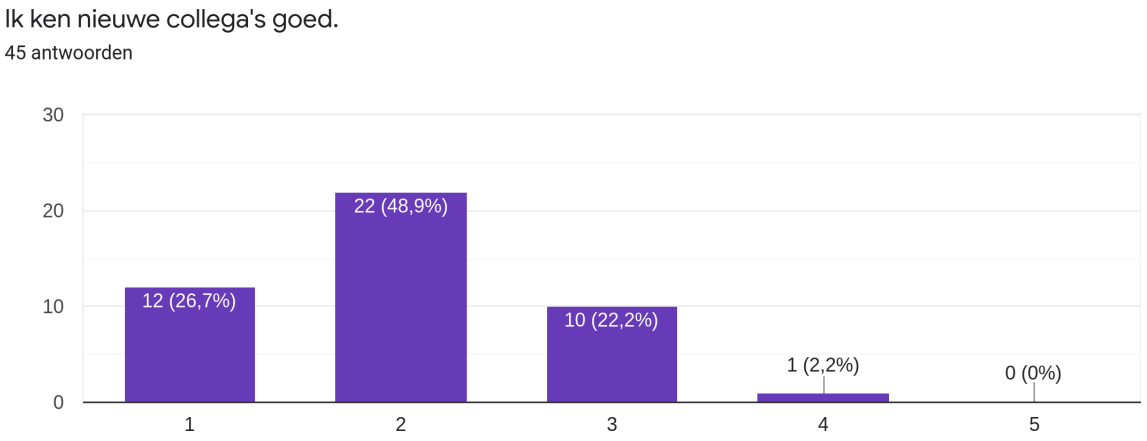
11.2 - Enquête onderzoek



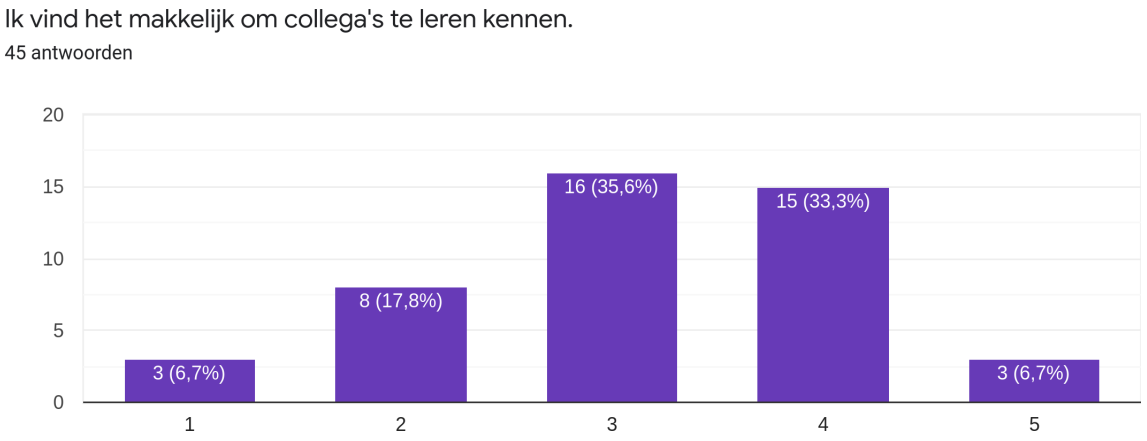
Figuur 11.2-1



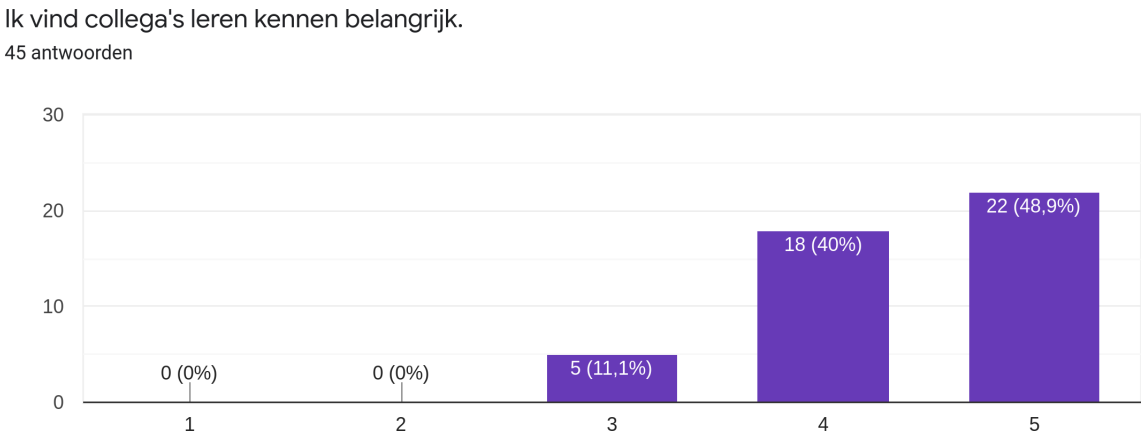
Figuur 11.2-2



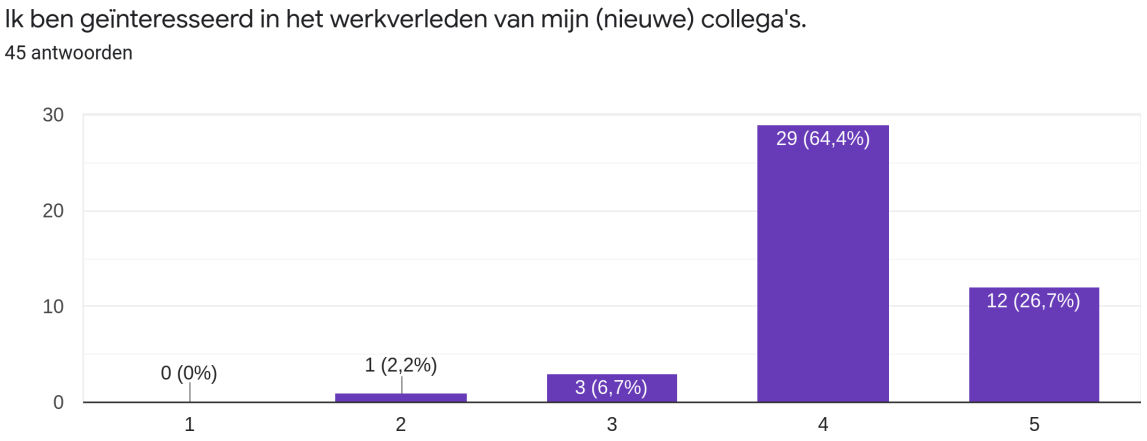
Figuur 11.2-3



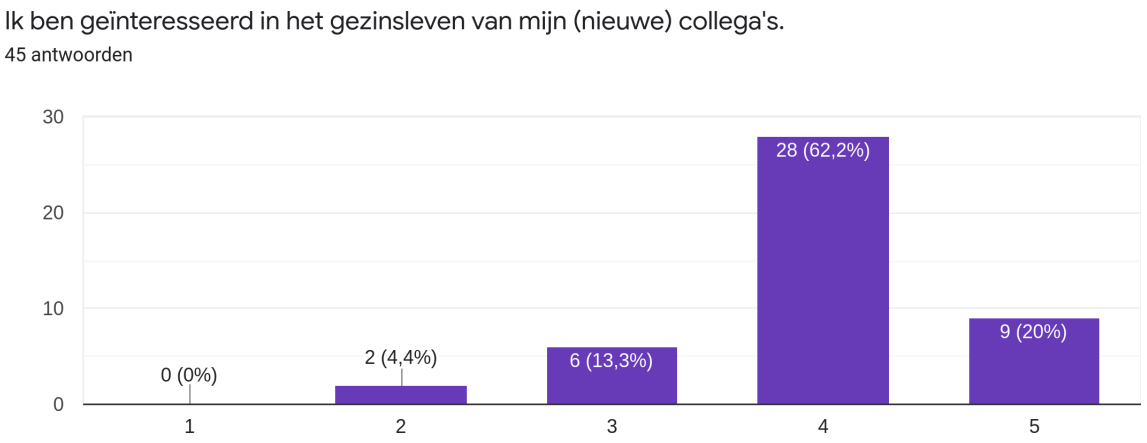
Figuur 11.2-4



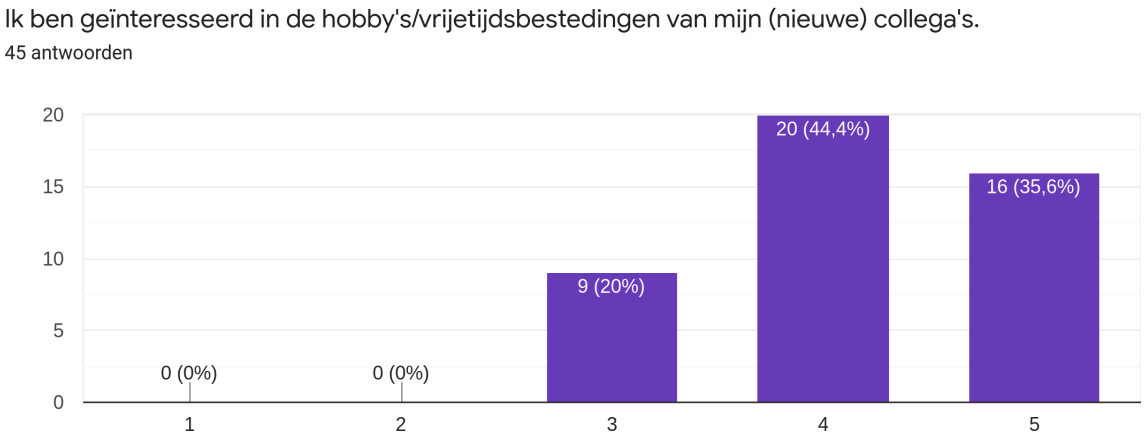
Figuur 11.2-5



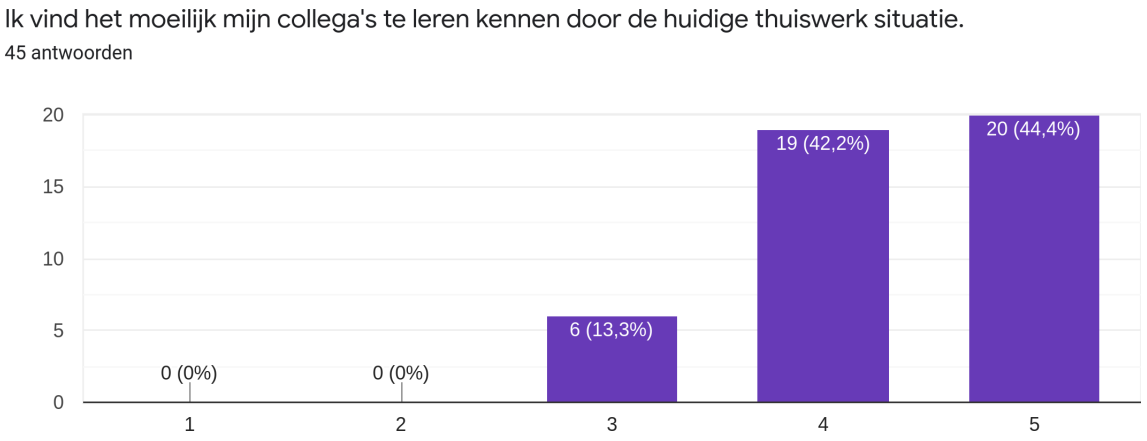
Figuur 11.2-6



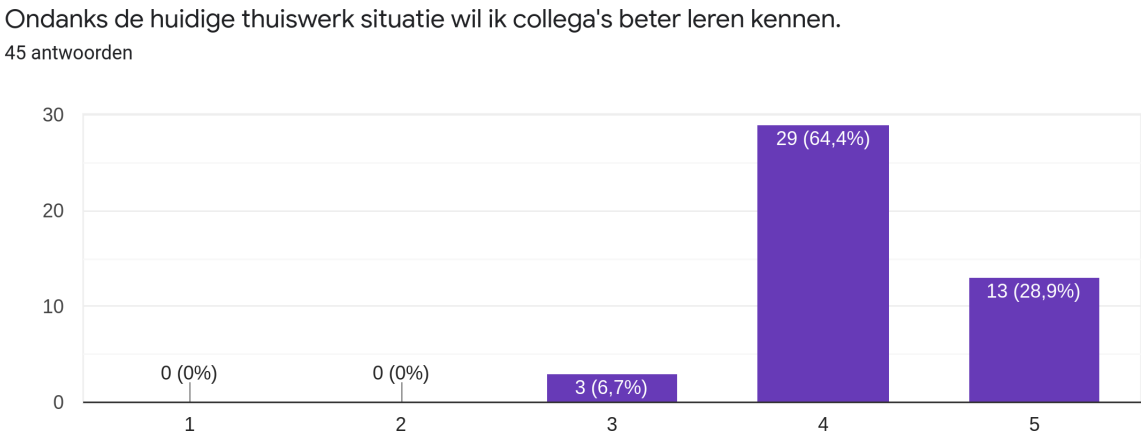
Figuur 11.2-7



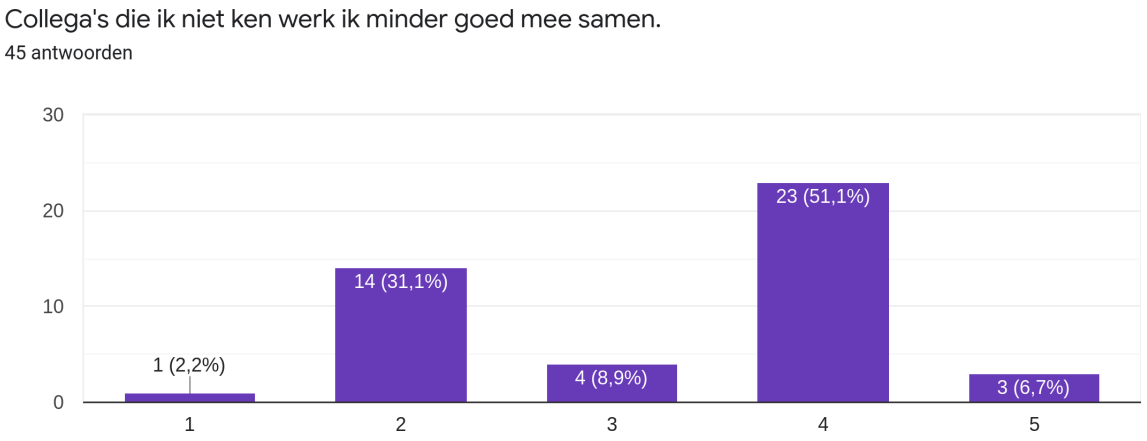
Figuur 11.2-8



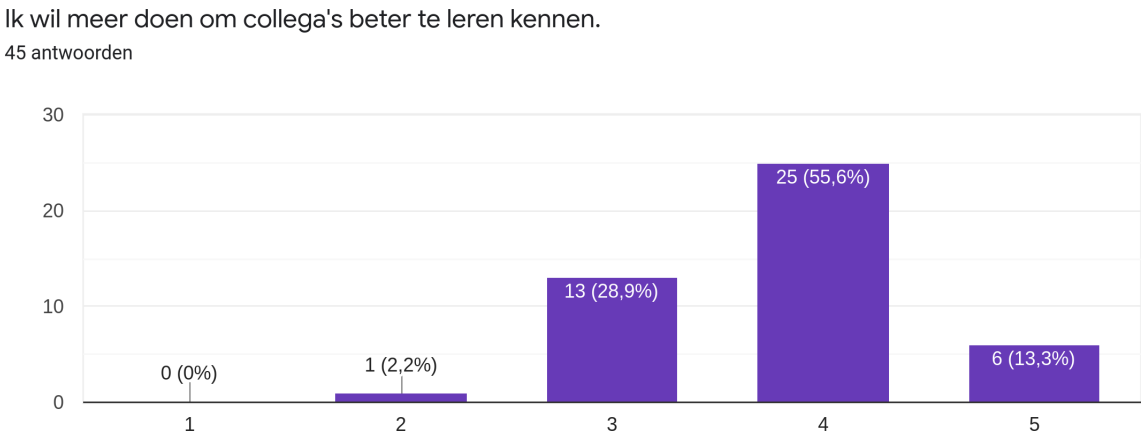
Figuur 11.2-9



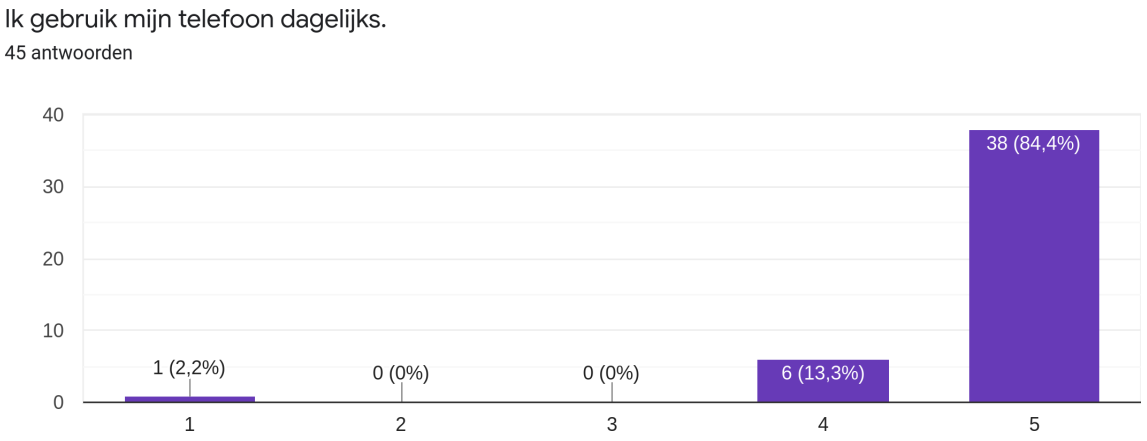
Figuur 11.2-10



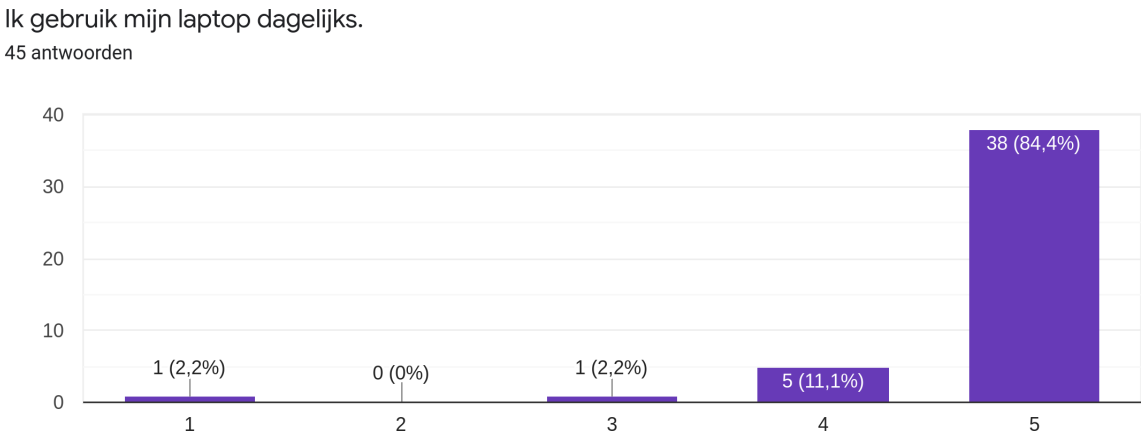
Figuur 11.2-11



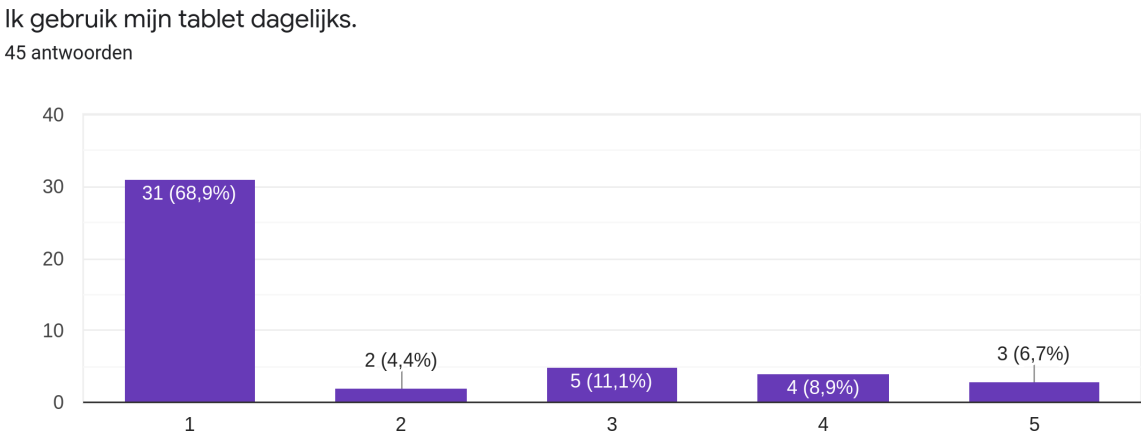
Figuur 11.2-12



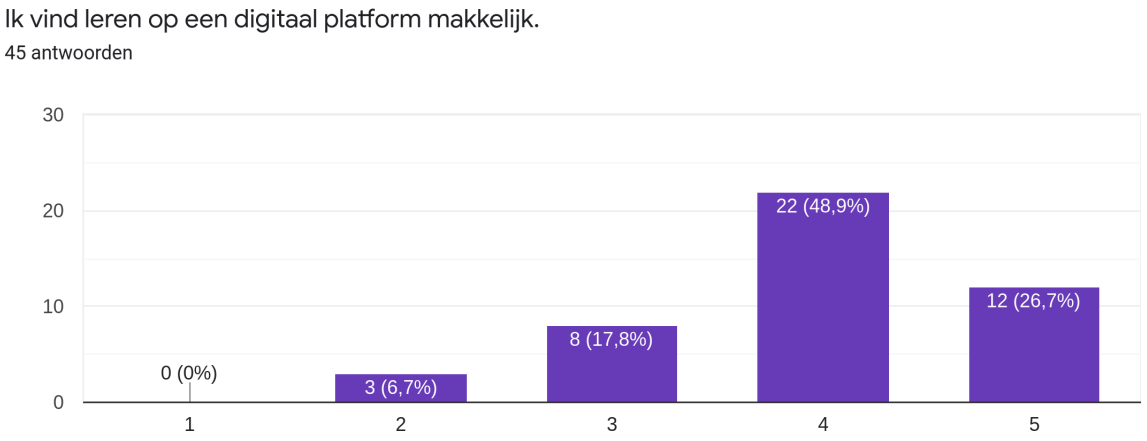
Figuur 11.2-13



Figuur 11.2-14



Figuur 11.2-15



Figuur 11.2-16