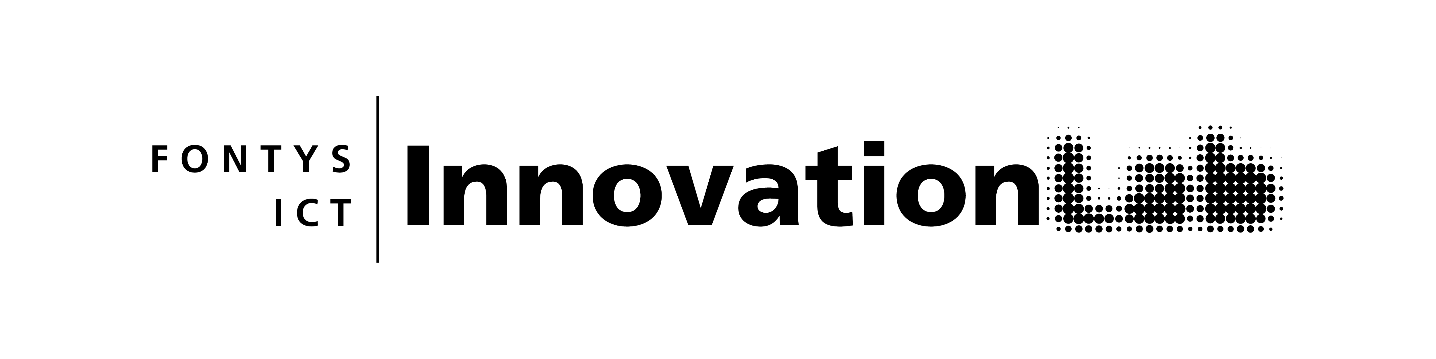
**Slimme Voetbalclub Administratie**





Angelo van Osch

Studentnummer: 5709288

Inhoud

[Probleemanalyse 3](#_Toc193027545)

[Plan van aanpak 4](#_Toc193027546)

[Cyberbeveiling 5](#_Toc193027547)

[Context diagram 9](#_Toc193027548)

[Design 10](#_Toc193027549)

[Use case diagram Kon het niet afmaken, want ik moest gaan betalen voor Lucid Chart. Volgende keer gebruik ik iets anders dan Lucid Chart. 12](#_Toc193027550)

[Databaseontwerp 13](#_Toc193027551)

[Reflectieverslag 14](#_Toc193027552)

# Probleemanalyse

***Huidige situatie***

De lokale voetbalclub heeft moeite om handmatig het ledenbeheer, de contributie-inning en de wedstrijdplanning/trainingen bij te houden. Dit leidt tot inefficiëntie en problemen zoals gemiste betalingen, dubbel geplande wedstrijden en onduidelijke teamindelingen.

Er moet een overzichtelijk en efficiënt systeem komen voor de lokale voetbalclub om leden te kunnen beheren, voetbalwedstrijden en training in te kunnen plannen en te bekijken en leden moeten een automatische herinneringen krijgen bij een achterstallige betalingen.

***Oorzaken van het probleem***

* Handmatige administratie zorgt ervoor dat foutgevoelig is en kost te veel tijd.
* Geen geautomatiseerde betalingsherinneringen zorgt ervoor dat leden soms niet betalen of te laat.
* Wedstrijden en trainingen inplannen en deze informatie verspreiden is lastig en niet overzichtelijk.

***Gevolgen van het probleem***

* De administratie gaat vaak fout en zorgt ervoor dat het bestuur en trainers veel tijd kwijt zijn aan administratieve taken.
* De lokale voetbalclub loopt inkomsten mis, omdat leden niet of te laat betalen.
* Leden komen niet opdagen voor wedstrijden en trainingen en zorgt ervoor dat leden vroegtijdig stoppen.

***Oplossing***

* Trainers en het bestuur kunnen efficiënt en overzichtelijk administratieve taken doen. Als ze iets mogelijk fout doen, dan moet er een foutmeldingen komen die duidelijk het probleem beschrijft en zegt wat er moet gebeuren.
* De leden van de voetbalclub krijgen een automatische email bij achterstallige betalingen.
* Je kan wedstrijden en trainingen inplannen en bekijken.

# Plan van aanpak

***Inleiding***

Dit project heeft als doel een slimme softwareoplossing te ontwikkelen voor een lokale voetbalclub. Het systeem zal het ledenbeheer, de contributie-inning en de wedstrijd- en trainingsplanning automatiseren.

***Basisfunctionaliteiten van het systeem.***

***Ledenbeheer***

* Registratie en login van nieuwe leden binnen de lokale voetbalclub.
* Beheer van lidmaatschap status.

***Contributiebeheer***

* Automatische herinnering bij achterstallige betalingen.
* Statusoverzicht van betalingen.

***Wedstrijd en training inplannen***

* Het plannen van een wedstrijd en een training.
* Inzicht van komende wedstrijden en trainingen.

***Technische implementatie***

Het systeem zal worden ontwikkeld als een **computerapplicatie** die compatibel is met **Mac, Linux en Windows**. Dit zorgt ervoor dat trainers en bestuursleden op verschillende besturingssystemen toegang hebben tot de applicatie. De applicatie zal een gebruiksvriendelijke interface hebben.

Voor de front-end ga ik Flutter/Dart gebruiken. Flutter is een cross platform en werkt op alle bestuurssystemen zoals Mac, Linux en Windows. Flutter heeft een moderne look met een snelle response time.

Voor de back-end ga ik Laravel gebruiken. Laravel is een krachtige PHP framework. Deze back-end zal werken als een rest API met verschillende endpoints. De rest API zal een verbinding zijn tussen de front-end en de database. Laravel werkt ook fijn samen met een MySQL database.

Voor de database ga ik een relationele database gebruiken genaamd MySQL. MySQL is een relationele database en voor dit project is dat het makkelijkst. Dit komt omdat er in veel complexe relatie zijn tussen de gegevens die in de database komen te staan. Een relationele database zorgt ervoor dat ik makkelijk en snel de gegevens die ik graag wil hebben beschikbaar zal hebben.

# Cyberbeveiling

Alle gegevens tussen de Flutter-client en de Laravel back-end worden versleuteld via HTTPS, dat gebruikmaakt van Transport Layer Security (TLS). TLS hanteert bij de eerste handshake asymmetrische encryptie, waarna het overschakelt op symmetrische encryptie (AES-256) voor betere prestaties.

Om HTTPS af te dwingen, wordt een SSL-certificaat aangevraagd via Let's Encrypt. Dit zorgt ervoor dat alle data tijdens transport versleuteld is, waardoor man-in-the-middle (MITM) aanvallen worden voorkomen. Zonder HTTPS kunnen API-verzoeken onderschept worden door aanvallers op hetzelfde netwerk.

Dit garandeert dat alle verzonden gegevens veilig blijven.

***Veilige opslag op de Flutter-client***

Voor de opslag van zeer gevoelige informatie wordt in de Flutter-client de package Flutter Secure Storage gebruikt. Dit is de veiligste manier om gegevens lokaal op te slaan binnen een Flutter-applicatie.

De opslaglocatie verschilt per besturingssysteem:

* **Android → Encrypted Shared Preferences (met Android Keystore)**
* **iOS & MacOS → Apple Keychain**
* **Windows → Windows Protected Storage (DPAPI)**
* **Linux → Secret Service API (zoals GNOME Keyring)**

Omdat deze gegevens lokaal worden opgeslagen en niet via de browser toegankelijk zijn, is Flutter Secure Storage bestand tegen XSS en CSRF-aanvallen.

***Validatie en beveiliging van gebruikersinvoer***

Alle gebruikersinvoer vanuit de Flutter-client wordt grondig gevalideerd voordat deze wordt verstuurd naar de Laravel back-end.

1. **Client-side validatie**

* Voor elk invoerveld wordt regelgebaseerde validatie (regex) gebruikt om alleen toegestane invoer te accepteren.
* Dit voorkomt fouten en verbetert de gebruikerservaring, maar client-side validatie is niet voldoende als enige beveiliging.

1. **Server-side validatie**

* Laravel gebruikt Eloquent ORM, dat automatisch prepared statements toepast om SQL-injectie te voorkomen.
* Eloquent ondersteunt ook typecasting (bijv. booleans, datums) om invoer correct te verwerken.
* Extra whitelisting kan worden toegevoegd om alleen verwachte waarden te accepteren.

Client-side validatie helpt gebruikersfouten te voorkomen, maar server-side validatie is essentieel. Alle gebruikersinvoer moet als onveilig worden beschouwd totdat het is gevalideerd, geïnspecteerd en verwerkt door de back-end.

Daarnaast worden maatregelen genomen tegen session hijacking en andere beveiligingsrisico’s om de integriteit van de applicatie te waarborgen.

***Role-Based Access Control (RBAC)***

Binnen de applicatie wordt een Role-Based Access Control (RBAC) systeem toegepast. Dit betekent dat gebruikers specifieke rollen krijgen, waarbij permissies niet individueel, maar op basis van een rol worden toegekend.

Dit systeem maakt het eenvoudiger om nieuwe functionaliteiten toe te voegen en gebruikers efficiënt aan een rol toe te wijzen.

De beschikbare rollen binnen de applicatie zijn:

* **User**
* **Trainer**
* **Bestuurslid**

Elke gebruiker heeft standaard de rol User, maar deze rol heeft geen speciale permissies. Extra functionaliteiten zijn beschikbaar voor Trainers en Bestuursleden, afhankelijk van hun specifieke rechten binnen de applicatie.

***Authenticatie en Autherization***

In de applicatie wordt JWT (JSON Web Token) gebruikt voor authenticatie en autorisatie.

JWT-token vermindert de kans op cyber-aanvals zoals, CSRF, XSS, rate limiting (max 60 API-calls per minuut per gebruiker), SQL-injection en Man-in-the-Middle.

Omdat een JWT-token een beperkte geldigheidsduur heeft van 1 uur, moet de applicatie het token op de juiste momenten vernieuwen. Dit gebeurt met behulp van een refresh token.

**1. Werking van JWT en Refresh Tokens**

**JWT Token (Access Token)** → Dit token wordt gebruikt voor authenticatie en autorisatie bij API-aanvragen. Het heeft een korte levensduur van 1 uur om veiligheidsredenen.

**Refresh Token** → Dit token heeft een langere levensduur van 30 dagen en wordt gebruikt om een nieuwe JWT-token op te halen zonder dat de gebruiker opnieuw hoeft in te loggen.

**2. Inloggen en token generatie**

Wanneer een gebruiker inlogt:

De gebruiker voert zijn email en wachtwoord in.

De Laravel back-end valideert de inloggegevens.

Laravel genereert twee tokens:

Een JWT-token voor korte termijn authenticatie.

Een refresh token voor langdurige sessies.

Beide tokens worden teruggestuurd naar de Flutter-app.

De tokens worden opgeslagen op de Client sided via de package Flutter Secure Storage.

De JWT-token wordt gebruikt voor API-aanroepen, terwijl de refresh token veilig wordt opgeslagen voor toekomstige vernieuwingen.

**3. Tokenvernieuwing wanneer JWT verloopt**

Wanneer een JWT-token verloopt, moet de app een nieuw token aanvragen met behulp van de refresh token. Dit proces verloopt als volgt:

De Flutter-app detecteert dat de JWT-token is verlopen, door een 401 Unauthorized statuscode van de Laravel back-end server.

De app stuurt de refresh token naar de back-end (bijvoorbeeld naar het /api/refresh endpoint).

Laravel controleert of de refresh token geldig en niet verlopen is.

Zo ja, dan genereert Laravel een nieuwe JWT-token en stuurt deze terug naar de Flutter-app.

De app slaat de nieuwe token op in Flutter Secure Storage en gebruikt deze voor verdere API-aanvragen.

***Wachtwoord beveiliging***

Wachtwoorden worden veilig verzonden van de Flutter-client naar de Laravel back-end via HTTPS, zodat ze tijdens transport niet kunnen worden onderschept. Voordat wachtwoorden in de database worden opgeslagen, worden er meerdere beveiligingsmaatregelen genomen om de gegevens te beschermen tegen aanvallen zoals database breaches en brute-force aanvallen.

**1. Hashing met een One-Way Function (Argon2)**

* Wachtwoorden worden niet in platte tekst opgeslagen, maar gehashed met een cryptografische one-way functie, zoals Argon2.
* Een one-way functie betekent dat het onmogelijk is om een hash terug te converteren naar het originele wachtwoord.
* Dit voorkomt dat zelfs databasebeheerders of aanvallers bij een datalek de wachtwoorden kunnen achterhalen.

Als twee gebruikers exact hetzelfde wachtwoord kiezen, zou het zonder extra maatregelen nog steeds dezelfde hash genereren. Dit kan aanvallers helpen om patronen te ontdekken in een databaselek. Daarom wordt er een salt toegevoegd.

2. **Salt toevoegen om Hash Collisions en Rainbow Table-aanvallen te voorkomen**

* Salt is een willekeurige waarde die wordt toegevoegd aan elk wachtwoord voordat het wordt gehasht.
* Dit zorgt ervoor dat zelfs als twee gebruikers hetzelfde wachtwoord hebben, de hashes totaal verschillend zullen zijn.
* Salting helpt ook tegen rainbow table-aanvallen. Een rainbow table is een vooraf gegenereerde lijst van veelgebruikte wachtwoorden en hun hashes, waarmee hackers gehashte wachtwoorden kunnen terugvinden.
* Doordat elke hash een unieke salt heeft, wordt deze aanvalsmethode nutteloos

**3. Slow Hashing tegen Dictionary Attacks en Brute Force-aanvallen**

* Er wordt gebruikgemaakt van slow hashing, waarbij de hash-functie opzettelijk traag wordt uitgevoerd met een instelbare work factor.
* Dit is effectief tegen dictionary attacks (waarbij een lijst met veelvoorkomende wachtwoorden wordt geprobeerd) en brute-force aanvallen.
* Hoe werkt dit? Bij elke loginpoging moet de server een berekening uitvoeren om het ingevoerde wachtwoord te hashen en te vergelijken met de opgeslagen hash. Door deze berekening kunstmatig te vertragen, wordt het moeilijker voor aanvallers om snel veel wachtwoorden te testen.
* Laravel gebruikt standaard Argon2, die beide slow-hashing ondersteunen met een aanpasbare work factor.

# Context diagram

Afbeelding met tekst, diagram, schermopname, lijn

Door AI gegenereerde inhoud is mogelijk onjuist.

# Design

***Inleiding***

Dit design is gemaakt voor het visuele en functionele ontwerp van een lokale voetbalclub. Het doel is om een aantrekkelijke, gebruiksvriendelijke en consistente ervaring te creëren voor de gebruikers.

In dit design volg ik de regels die zijn beschreven in material design 3.

In dit hoofdstuk dat bestaat uit meerdere paragrafen lees je meer over de keuzes die ik heb gemaakt op het gebied van doelgroep, typografie, kleurenschema en responsiviteit en toegankelijkheid. Ik leg uit waarom deze keuzes passen bij het project en hoe ze bijdragen aan de uitstraling en gebruiksvriendelijkheid van de website.

***Doelgroep***

Mijn doelgroep voor dit design bestaat voornamelijk uit mannen van verschillende leeftijden, van jong tot oud. Voor de primaire kleur heb ik toch gekozen voor paars, ondanks dat het vaak als een vrouwelijke kleur wordt gezien. Paars straalt een moderne, verfijnde uitstraling uit die goed past bij het ontwerp en de uitstraling die ik wilde creëren.

Bij het ontwikkelen van de functionaliteiten heb ik extra aandacht besteed aan de onderdelen die vooral door de oudere gebruikers in mijn doelgroep gebruikt gaan worden. Het design is zodanig opgebouwd dat deze functionaliteiten duidelijk en gebruiksvriendelijk zijn, zodat zelfs mensen met minder ervaring de applicatie eenvoudig kunnen gebruiken.

Aangezien mijn gebruikers allemaal Nederlands zijn en de Nederlandse cultuur doorgaans als low-context wordt beschouwd, heb ik het design zo eenvoudig en direct mogelijk gehouden. Low-context gebruikers waarderen duidelijkheid en snelheid. Ze willen geen onnodige informatie of visuele ruis, maar juist een strak en begrijpelijk ontwerp om snel vertrouwen op te bouwen.

***Typografie***

Ik heb gekozen voor de typografie Roboto. Dit moderne en veelzijdige lettertype wordt veel gebruikt voor websites, mobiele apps en desktopdesigns, vooral vanwege de uitstekende leesbaarheid.

Roboto is ontworpen door Google en dient als het standaardlettertype voor Material Design. Het combineert strakke, geometrische vormen met een natuurlijke en toegankelijke uitstraling, waardoor het zowel professioneel als gebruiksvriendelijk is.

Voor de typografie-indeling heb ik de volgende gewichten van Roboto gekozen:

* **Standaard bodytekst → Roboto Regular (400)**
* **Knoppen en labels → Roboto Medium (500)**
* **Koppen en subkoppen → Roboto Bold (700)**
* **Subteksten en hints → Roboto Light (300)**

***Kleurenschema***

Voor het kleurenpalet heb ik de 60/30/10-regel toegepast, een veelgebruikte designtechniek die zorgt voor een gebalanceerde en harmonieuze uitstraling.

* **60% - Primaire kleur: 6A1B9A – De dominante kleur in het design, gebruikt voor achtergronden en grote oppervlakken en buttons.**
* **30% - Secundaire kleur: #FFFFFF – Een ondersteunende kleur die contrast biedt en wordt toegepast op elementen zoals secties en wit ruimte rondom de primaire kleur.**
* **10% - Accentkleur: #000000 – Een opvallende kleur die strategisch wordt ingezet voor knoppen en teksten.**

***Reponsiviteit en toegankelijkheid***

Deze applicatie is ontworpen om te functioneren op laptops en desktops met verschillende besturingssystemen, zoals Windows, MacOS en Linux. Omdat de applicatie zich richt op een desktop-only ervaring, is er geen ondersteuning voor mobiele apparaten of tablets. Dit maakt dat mijn ontwerp zich niet hoeft aan te passen voor verschillende schermformaten zoals bij mobiele apps, maar specifiek gericht is op de optimalisatie voor grotere schermen.

Hoewel er geen responsief design voor mobiele apparaten is, blijft toegankelijkheid een belangrijke prioriteit. Het design is dusdanig opgebouwd dat alle teksten goed leesbaar zijn, knoppen gemakkelijk te vinden en te gebruiken zijn, en er voldoende contrast is tussen de verschillende elementen. Dit zorgt ervoor dat de applicatie ook voor minder ervaren gebruikers toegankelijk is, zonder in te boeten op de visuele en functionele kwaliteit.

# Use case diagram Kon het niet afmaken, want ik moest gaan betalen voor Lucid Chart. Volgende keer gebruik ik iets anders dan Lucid Chart.

**Afbeelding met schermopname, zwart, Symmetrie, duisternis

Door AI gegenereerde inhoud is mogelijk onjuist.**

# Databaseontwerp

**Afbeelding met tekst, lijn, ontvangst, diagram

Door AI gegenereerde inhoud is mogelijk onjuist.**

# Reflectieverslag

Ik had gekozen voor de casus ‘Lokale Voetbalclub’. Het idee was om een applicatie te maken waarmee de club op een makkelijke manier wedstrijden en trainingen kon beheren en aanmaken. Daarnaast moest er een betaalsysteem in zitten dat leden herinnert aan achterstallige betalingen, zodat alles op tijd wordt betaald.

Helaas is het me niet gelukt om alles op tijd af te krijgen. Dit komt doordat ik dacht dat ik veel meer tijd had voor deze opdracht dan daadwerkelijk het geval was. Ik had verwacht dat het hele project in één keer af moest, terwijl het in werkelijkheid de bedoeling was om binnen een maand alleen de belangrijkste functionaliteiten te prioriteren, uit te werken en een UML-diagram te maken. Dit was mijn eigen misverstand, en ik zal hier bij toekomstige opdrachten beter op letten.

Ik ben ingestroomd in semester 2 omdat ik al een MBO-4 opleiding heb afgerond en redelijk wat ervaring heb met software development. Maar dat betekent ook dat ik nog niet volledig bekend ben met de werkwijze binnen deze opleiding en bepaalde kennis mis die studenten uit semester 1 al hebben opgebouwd. Desondanks heb ik de afgelopen week mijn best gedaan om alles zo goed mogelijk af te ronden binnen de tijd die ik had.

Ondanks de tijdsdruk is het me wel gelukt om een basisopzet van de applicatie te maken. Ik heb gewerkt aan het inlog- en registratiescherm, en de navigatie binnen de applicatie is duidelijk opgezet. Daarnaast kunnen gebruikers beheerd worden, mits je de juiste rechten hebt. Ook heb ik een aantal UML-tekeningen uitgewerkt.

In de toekomst zal ik beter opletten wat er precies van mij wordt verwacht, zodat ik de casus beter kan plannen en uitvoeren.