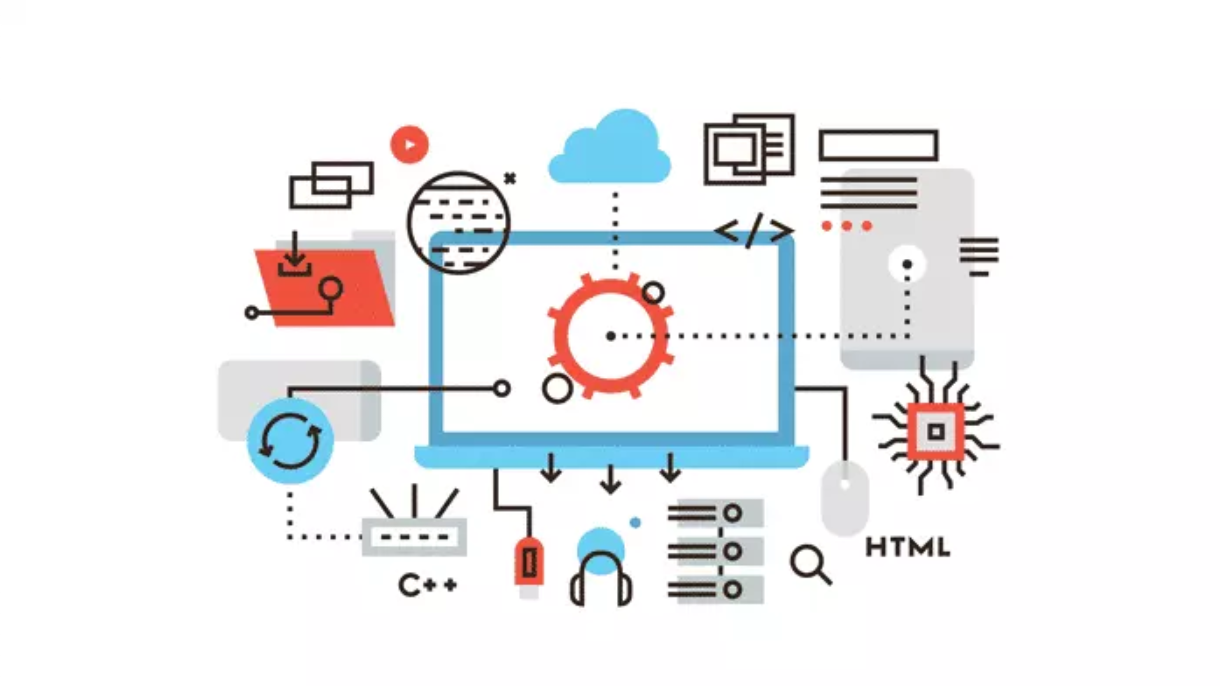
**Sjabloon 3**

Ontwerpen van software

**B1-K1-W2**



Geschreven door (voor- en achternaam): Thom Hermelink

Gecontroleerd door (voor- en achternaam):

Datum: 10-3-2025

Versie: 1

Inhoud

[Inleiding 3](#_Toc182471328)

[User Stories 4](#_Toc182471329)

[Gegevens (bijvoorbeeld ERD, klassendiagram, normaliseren) 5](#_Toc182471330)

[Gebruikersinterface (bv. usecasediagram, wireframes, mock-ups) 6](#_Toc182471331)

[Programmalogica (bijvoorbeeld activiteitendiagram, flowchart) 7](#_Toc182471332)

[Privacy en Security 8](#_Toc182471333)

# Inleiding

Maak een ontwerp voor de aan jou toegekende user stories.

Jouw ontwerp moet aansluiten op de wensen en eisen. Maak gebruik van minimaal één schematechniek per onderwerp.

1. Gegevens (bijvoorbeeld ERD, klassendiagram, normaliseren)
2. Gebruikersinterface (bijvoorbeeld usecasediagram, wireframes, mock-ups)
3. Programmalogica (bijvoorbeeld activiteitendiagram, flowchart).

Je deelontwerp bevat dus minimaal drie schematechnieken, zoals je die tijdens de opleiding geleerd hebt. Je mag er meer gebruiken om je ontwerp te ondersteunen. Het gebruik van een framework is toegestaan.

# User Stories

*Vanuit jullie taakverdeling heb je user stories toegedeeld gekregen. Geef hier jouw user stories weer waarvoor je een ontwerp gaat maken.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Naam student:** | Thom Hermelink | | |
| **Nummer & Titel**: | 01, map | | |
| **Omschrijving**: | | | |
| Als de speler wil ik een map hebben om op te racen zodat ik weet waar ik heen moet gaan | | | |
| **Business Value:** | 4 | **Story Points:** | 5 |
| **Acceptatiecriteria**: | | | |
| * Er is een map gemaakt * Het is duidelijk waar je kan rijden * Het verschil tussen baan en niet baan is duidelijk * Bepaald waar obstacles (enemies) kunnen spawnen | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Naam student:** | Thom Hermelink | | |
| **Nummer & Titel:** | 02, map select | | |
| **Omschrijving:** | | | |
| Als speler wil ik een map kunnen selecteren zodat ik invloed heb over op welke map er word gereden | | | |
| **Business Value:** | 3 | **Story Points:** | 2 |
| **Acceptatiecriteria:** | | | |
| * De map namen worden displayed via een menu * Je kan een map selecteren * Je ziet waar de selector op staat | | | |

# Gegevens (bijvoorbeeld ERD, klassendiagram, normaliseren)

*Plaats hier jouw gemaakte ontwerp(en) met betrekking tot gegevens.*

In het onderstaande klassenDiagram zie je de classes met betrekking tot het laden van de map, denk hierbij bijvoorbeeld aan het selecteren van de map, laadschermen en map obstakels.

A diagram of a computer program

AI-generated content may be incorrect.

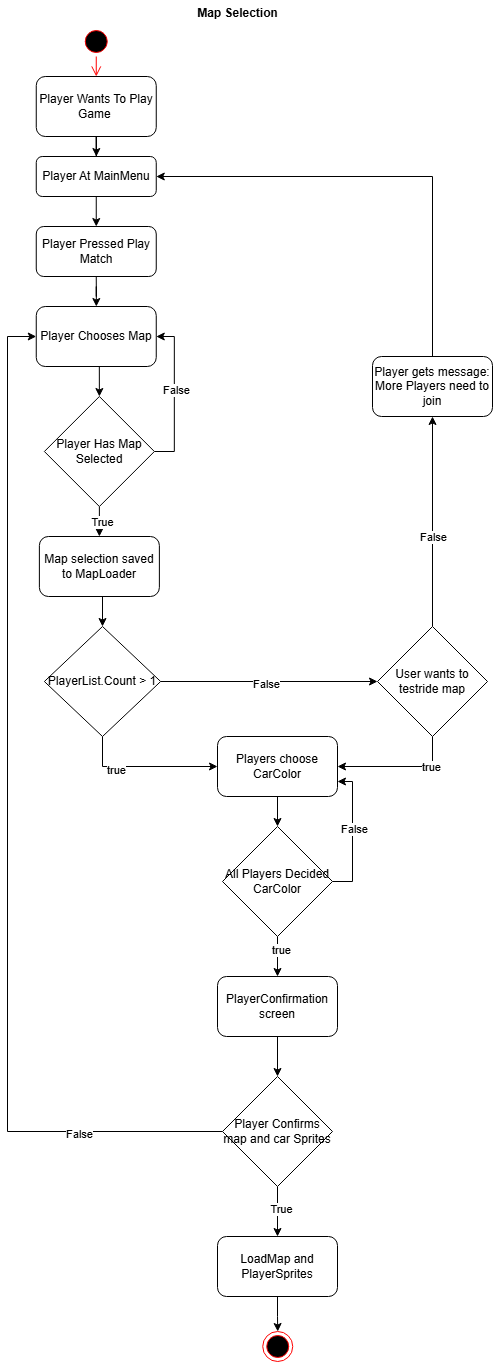
# Gebruikersinterface (bv. usecasediagram, wireframes, mock-ups)

*Plaats hier jouw gemaakte ontwerp(en) met betrekking tot de gebruikersinterface.*

A diagram of a diagram

AI-generated content may be incorrect.

# Programmalogica (bijvoorbeeld activiteitendiagram, flowchart)



# Privacy en Security

Onderbouw op basis van de onderstaande punten je ontwerpkeuzes op haalbaarheid, privacy en security.

* Laat zien dat je ontwerp voldoet aan de eisen van de AVG en de OWASP top 10.

**AVG**

*Het ontwerp van Crash N Dash gebruikt geen persoonlijke gegevens, de speler(s) kunnen nergens hun namen invullen, daarnaast is er geen database waar gegevens kunnen opgeslagen worden waardoor het ontwerp voldoet aan de AVG regels*

* Welke persoonsgegevens worden verwerkt of gebruikt?

*Er worden geen persoonlijke gegevens gebruikt, verwerkt of opgeslagen. Hierdoor voldoet het ontwerp aan de AVG regels.*

* Waarom of voor welk specifiek doel worden deze gegevens gebruikt?

*Er worden geen persoonlijke gegevens gevraagd aan de speler, dit komt doordat het spel niet anders gaat functioneren met persoonlijke gegevens.*

* Vind je het doel gerechtvaardigd?

*Het doel is gerechtvaardigd, er worden geen persoonlijke gegevens verzameld omdat er geen noodzaak voor is.*

**OWASP top 10**

* Welke bedreigingen van de OWASP top 10 kunnen wel of niet voorkomen in het door jou gebruikte framework of IDE?

*De OWASP top 10 bestaat uit:*

***Broken Access Control***

*Dit betekend dat een gebruiker zonder de juiste machtigingen toch toegang kan krijgen tot onderdelen waar de gebruiker dit niet kan, denk hierbij aan dat een normale gebruiker toch bij een admin panel kan komen doordat er geen goede checks of authorization word toegepast.*

*Dit zou kunnen voorkomen doordat de developers geen goede authorization of permission checks toevoegen waardoor de gebruiker een oneerlijk voordeel kan krijgen in het spel. Om dit te voorkomen is het belangerijk om checks toe te voegen.*

***Voor het project crash N dash kunnen we wel kwetsbaar zijn voor broken access control, maar wel minimaal, dit komt doordat wij geen API’s gebruiken en ook geen data naar een server sturen. Wel kan een gebruiker mogelijk bij lokale bestanden.***

***Cryptographic Failures***

*Dit OWASP punt gaat over het (fout) versleutelen van data, het kan gebeuren door dat je oude of zwakke encryptie programmas gebruikt, of doordat je data uberhaubt niet versleuteld tijdens transit of storage waardoor ze veel makkelijker te lezen zijn.*

*Je kunt dit voorkomen door gegevens goed te encrypten door sterkere encryptie programmas zoals AES-256.*

*In het geval dat je helemaal niet je data encrypt is het al een goede start om uberhaubt te beginnen met het encrypten van data zoals wachtwoorden en gevoelige informatie zodat ze niet direct te lezen zijn, maar zelfs hiervoor moet je ook de encryption keys goed beheren, anders zou er alsnog iedereen bij kunnen.*

***Voor het project crash N dash zijn wij niet kwetsbaar door cryptographic failures, dit komt doordat wij geen gevoelige informatie opslaan of versturen.***

***Injection***

*Dit punt betekend een SQL injection, als bijvoorbeeld een website een invoer veld heeft die gelinkt is aan een database, kan daar inplaats van een normale naam ook een stukje code worden ingevoerd waardoor de gebruiker toegang kan krijgen tot de totale database.*

*Om dit te voorkomen kun je parameters gebruiken in het stukje code die communiceert met de SQL database, hierdoor word de tekst die ingevoerd word niet gezien als code, maar als normale data en kan niet uitgevoerd worden*

**Voor het project crash N dash zijn wij niet kwetsbaar voor een SQL injection, dit komt doordat wij geen communicatie hebben met een database**

***Insecure Design***

*Bij insecure design zit het probleem in het design van de code, dit zou kunnen betekenen dat je iets programmeert, maar op een verkeerde manier de checks uitvoert.*

*In makkelijke termen kun je het zien zoals dit: je hebt een huis gebouwd, maar je bent vergeten de sloten te installeren. Daarna check je of de muren stevig zijn en concludeer je dat jouw huis veilig is.*

*Hierbij zie je dat de check die word gedaan bij de muren niks te maken hebben met de veiligheid zoals sloten. omdat de designer niet heeft gedacht aan de veiligheid (zoals sloten) is het design insecure en zouden er zomaar mensen binnen kunnen komen.*

***Voor het project crash N dash zijn wij mogelijk kwetsbaar voor insecure design, het kan natuurlijk altijd voorkomen dat wij als developers niet goed kijken wat er precies gechecked moet worden. Doordat wij als developers bepaalde dingen kunnen missen is het belangerijk dat er echte testers onderdelen van het programma grondig testen om design fouten op te sporen.***

***Security Misconfiguration***

*Bij dit punt van OWASP gaat het over de security services, je kunt hierbij er vanuit gaan dat er niks mis is met de service zelf, maar wel met de instellingen die toegepast zijn bij de service.*

*Als we weer het huis pakken van vorig OWASP punt komt het er op neer dat je de sloten hebt geinstalleerd, maar je bent vergeten om het slot daadwerkelijk op slot te doen.*

*Hierbij zie je dat er dus niks mis met het slot is, maar wel met de instelling waar het slot op staat (open)*

*Als je meer kijkt in de software wereld dan is het open slot:*

* + *Standard wachtwoorden zoals: “admin123” of “admin”*
  + *Ongebruikte of oude functies oud betekend dat het niet word onderhouden, hierdoor kan de gebruiker informatie krijgen door de ongebruikte functie*
  + *Foutmeldingen aan gebruikers laten zien, hierdoor krijgt de gebruiker een beter inzicht hoe het programma werkt en word het hacken ervan makkelijker.*

***Voor het project crash N dash zijn wij minimaal kwetsbaar voor een security misconfiguration fout, dit komt doordat wij voor het spel geen wachtwoorden gebruiken, geen (technische)foutmeldingen laten zien op het scherm. Maar wij zouden wel kwetsbaar kunnen zijn voor het punt van de ongebruikte functies, het is voor ons dus belangerijk om alle ongebruikte functies te verwijderen om zo minder risico op te lopen dat data word geleaked aan de gebruiker.***

***Vulnerable and Outdated Components***

*Dit punt gaat over software of software onderdelen die niet up to date zijn, dit betekend dat je meer risico loopt als je oudere software runt omdat die meer kans hebben om bekende zwakke plekken te hebben (die mogelijk al zijn gefixed in een latere update).*

*Als we weer aan het voorbeeld van het huis denken, dan betekend dit dat je dit keer het slot hebt geinstalleerd, en dat het slot ook op slot zit. Maar het is bekend dat het slot makkelijk kan geopend worden met een trucje.*

*De juiste actie die je moet doen in het voorbeeld is het slot vervangen met een nieuwer slot, in software termen betekend dit vaak het installeren van een nieuwere versie van de software.*

*Het kan ook betekenen dat je nog software hebt geinstalleerd die je eigenlijk niet meer gebruikt, de correcte actie die je dan moet uitvoeren is simpelweg de software verwijderen zodat het geen beveiligingsrisico meer vormt.*

***Voor het project crash N dash zijn wij medium kwetsbaar voor vulnerable en outdated components, dot kan voorkomen doordat wij als developers dingen kunnen vergeten, hiermee bedoel ik dat wij onderdelen wel eens vergeten te updaten zoals de unity versie, of de nieuwste versie van visual studio. Het kan dus verstandig zijn om te noteren wanneer er een nieuwe versie komt en mocht dit erg belangerijk zijn, dit te noteren in een agenda zodat iedereen dan zijn of haar versie moet updaten.***

***Identification and Authentication Failures***

*Dit OWASP punt gaat over het identificeren en toegang geven van gebruikers tot een bepaald systeem, maar dit kan ook fout gaan, hierbij kan het zijn dat de controle toch mensen toegang geeft tot bepaalde systemen waar zij niet bij horen te kunnen.*

*Met het voorbeeld van het huis kun je dit zien als het slot niet goed werkt en open gaat bij elke sleutel die geprobeerd wordt. Hierbij word er dus niet gefilterd welke mensen er echt in moeten kunnen en welke niet.*

*De oplossing in de software wereld ligt de oplossing deels bij de gebruiker, maar ook deels bij de developer.*

* + *De developer moet proberen waar mogelijk multifactor authentication toe te passen*
  + *Zwakke wachtwoorden niet toelaten (een weak password check implementeren)*
  + *Bruteforce handling systeem, je moet bijvoorbeeld 30 seconden wachten bij elke foute 3 verkeerde wachtwoord inputs*

***Voor het project crash N dash zijn wij niet kwetsbaar voor identification en authentication failures, dit komt doordat wij geen gebruik maken van echte “accounts”, wij gebruiken lokale speler profielen waarbij je geen gegevens moet invullen en daardoor niks kan verifiëren.***

***Software and Data Integrity Failures.***

*Dit OWASP punt gaat over de betrouwbaarheid en het gebrek aan controle over software die jij gebruikt. Dit zou kunnen komen doordat er teveel externe bronnen worden gebruikt, hierdoor kan het dus zijn dat paar van deze externe bronnen verouderd zijn en daardoor wel een risico vormen. Het kan ook zijn dat er niet gechecked word op een digitale handtekening waardoor je niet 100% zeker kan zijn dat de bron nog van de juiste afzender komt.*

*Een voorbeeld waarbij je aan zou kunnen denken is een pakketje, op het moment dat je een pakketje binnenkrijgt waarbij het lijkt op iets dat jij hebt besteld, maar nog niet hebt gechecked of het van de juiste afzender komt, hierdoor kan het dus zijn dat er iets anders in zit dan dat jij verwacht.*

*In dit voorbeeld lijkt dit onschadelijk, maar in de digitale wereld betekend het “pakketje openen” het downloaden van de bestanden waardoor er schadelijke software op de computer staat.*

***Voor het project crash N dash zijn wij niet kwetsbaar voor Software and data integrity failures, dit komt doordat (mits wij alles netjes updaten) er geen gebruik word gemaakt van ongecontroleerde bronnen buiten Unity of visual studio. Wat wij wel kunnen gebruiken zijn bronnen die via Unity zijn gecontroleerd, maar aangezien dit al is gecontroleerd zijn wij niet kwetsbaar voor dit punt.***

***Security Logging and Monitoring Failures***

*Dit OWASP punt gaat over het bijhouden van belangerijke informatie zoals login attempts, failed login attempts en high value transactions, ze noemen dit ook wel auditable events oftewel controleerbaare gebeurtenissen.*

*De failure bij logging gaat over het niet goed bijhouden van logs, hierdoor word het opsporen van activiteiten zoals verandering van rechten voor een bepaalde gebruiker, of verdachte transacties veel moeilijker.*

*Om dit op te lossen is het belangerijk om acties te loggen, denk hierbij aan het loggen van veranderingen van machtigingen, grote transacties, inlog pogingen. Daarnaast is het belangerijk dat deze logs worden bewaard op een veilige plek. Ook is het heel verstandig om een systeem te maken die meldingen stuurt wanneer een verdachte transactie of een verandering van machtigingen heeft plaats gevonden.*

***Voor het project crash N dash zijn wij minimaal kwetsbaar voor security logging en monitoring failures, dit komt doordat er geen high value transactions en inlog gegevens worden opgeslagen doordat er geen echte “accounts” zijn. Wel kunnen de spelers currency krijgen om te spenderen in de shop, hiervoor zouden we wel logs kunnen maken om eventuele checks te kunnen maken op basis van het gelogde currency en transacties.***

Server***-Side Request Forgery (SSRF)***

*Dit OWASP punt gaat over dat een server zomaar alle linkjes van een gebruiker accepteert om gegevens op te halen, hierdoor kan de server gegevens ophalen van niet vertrouwde servers waardoor de gebruiker toegang kan krijgen tot informatie waar hij normaal niet bij zou kunnen komen zoals databases of API’s.*

*Om te voorkomen dat een server niet alle linkjes accepteert, moet je een lijst maken met vertrouwde linkjes, daarnaast moet je aan de hand van de vertrouwde lijst ook de ingevoerde linkjes gaan checken. Ook kun je de netwerk toegang beperken voor de server voor wat de server echt alleen nodig heeft, en niet meer.*

***Voor het project crash N dash zijn wij niet kwetsbaar voor Serverside request forgery, dit komt doordat wij niet communiceren met een server, en daardoor niks kunnen controleren van data dat vanuit een server komt. Ook gebruiken wij geen databases en API’s waardoor er geen risico is op dit OWASP punt.***

* Welke (online)bronnen heb je gebruikt?

https://owasp.org/Top10/

[Over OWASP en de OWASP top 10 - StackSecure](https://www.stack-secure.com/blog/over-owasp-en-de-owasp-top-10)

<https://www.egenix.nl/nieuws/wat-is-de-owasp-top-10-een-leidraad-voor-veilige-websites>