# Library application

**Naam:** Gianni **&** Nalany

**Project versie:**

**Projectcode:**

**Datum:**

# Inhoudsopgave

# Inleiding

De project definitie beschrijft voor wie en waarvoor de applicatie bedoeld is. Het doel van dit project is om een gebruiksvriendelijke C# bibliotheek applicatie te bouwen voor bezoekers, personeel en beheerders, zodat zij gemakkelijker toegang hebben tot de diensten die de bibliotheek biedt. Bijvoorbeeld het uitlenen van een boek.

Het programma moet helpen om bezoekers  stukken, media, zoals boeken en cd's, te zoeken en te reserveren. Ook moeten ze kunnen lenen reserveringen en beheren op een makkelijke manier. Met beheren wordt bedoeld dat ze hun leningen en reservaties kunnen overzien, en bijvoorbeeld een boek kunnen verlengen.

Het personeel, zoals bijvoorbeeld de beheerders en eventueel een IT-team, heeft toegang tot uitgebreide gegevens over de bibliotheek, zoals uitleengegevens en inventarisbeheer. Ook kunnen ze betalingen verwerken voor verschillende kosten, zoals abonnementen, geleende media, reserveringskosten en boetes.

De projectdefinitie bevat uitgebreide informatie over de vereisten van het bibliotheekprogramma, zowel functioneel als technische vereisten. Het beschrijft ook de verschillende fasen van het ontwikkelproces, onder andere het ontwerp, de ontwikkeling, en het testen van de software. Het is van belang om de vastgestelde werkwijzen en richtlijnen te volgen, zoals het is vastgesteld in de colleges en het lesmateriaal, om het project succesvol af te ronden.

# Probleemstelling

De bibliotheek heeft behoefte aan een nieuwe applicatie die gebruiksvriendelijk is voor zowel het personeel als de bezoekers. Verouderde systemen kunnen alleen maar leningen en reserveringen bijhouden. Er is over het algemeen niet voldoende nagedacht over het verwerken van betalingen, abonnementen en het uitschrijven van boetes. ook zijn de verouderde systemen traag en complex. het is aan ons de taak om een nieuw systeem te plaatsen met de missende functionaliteiten en wat ook een stuk sneller werkt.

Omdat er zoveel functionaliteit mist is er onnodig veel druk op het werkproces. Zijn de administratieve lasten hoger dan nodig is. En zijn klanten over het algemeen minder tevreden. Er is behoefte aan een oplossing die voldoet aan de technische en functionele eisen.

Het doel van dit project is om een ASP.NET Core Web Applicatie te ontwikkelen die de genoemde problemen aanpakt en een geavanceerdere bibliotheek applicatie biedt met uitgebreidere functionaliteiten, een duidelijk gebruikersinterface en een veilig inlogsysteem. Door dit probleem op te lossen, wordt de gebruikerservaring ook verbeterd en neemt de druk af op  werkprocessen binnen de bibliotheek.

# Scope van het project

## Acceptatiecriteria

### technische requirements:

- De applicatie kan worden gebouwd en uitgevoerd zonder fouten.
- De applicatie implementeert een veilig inlogsysteem op basis van gebruikersnaam/e-mail en wachtwoord, met ondersteuning voor de ingebouwde "Authentication Type" en "Individual Accounts".
- De applicatie maakt gebruik van HTTPS in plaats van HTTP.
- De applicatie is gebaseerd op de ASP.NET Core Web Applicatie met Model-View-Controller (MVC) en .NET 7.0.
- De connection string is correct geconfigureerd en verwijst naar de juiste database.
- De applicatie maakt gebruik van een database met Entity Framework en bevat Migrations om de database door anderen te kunnen installeren.
- De database bevat Seeding-Data met logische gegevens (minimaal 100 records per tabel, met uitzondering van abonnementen en locaties).
- De code voldoet aan de Microsoft Coding Conventions, behalve de conventie "Use parentheses to make clauses in an expression apparent".
- De GitHub repository bevat een versiegeschiedenis met meerdere commits die overeenkomen met een normaal ontwikkeltraject.
- Het design document is aanwezig in de repository.

## functionele requirements:

- Bezoekers kunnen items en auteurs inzien en zoeken op basis van titel, auteur, locatie, etc.
- Bezoekers kunnen items filteren op basis van specifieke criteria, zoals auteur, jaar van uitgave, locatie, etc.
- Bezoekers kunnen items reserveren.
- Bezoekers kunnen hun eigen leningen en reserveringen beheren.
- Personeel kan alle functionaliteiten van bezoekers uitvoeren, inclusief het uitlenen en innemen van items.
- Personeel kan items en auteurs aanmaken en bewerken.
- Personeel kan alle leningen en reserveringen beheren.
- Personeel kan volledige CRUD-operaties uitvoeren op gebruikers, inclusief het aanpassen van type/niveau/rechten.
- Personeel kan openstaande posten (facturen/uitstaande bedragen) inzien.
- Personeel kan betalingen verwerken (registratie van betalingen, exclusief betaalmethoden zelf).
- Personeel kan bezoekers accounts blokkeren.
- Personeel kan locaties toekennen aan items.
- Personeel kan abonnementen toevoegen en annuleren.
- Beheerders hebben alle functionaliteiten van personeel en kunnen items en auteurs verwijderen.
- Beheerders kunnen volledige CRUD-operaties uitvoeren op gebruikers, inclusief alle type/niveau/rechten.
- Beheerders kunnen alle accounts blokkeren.
- Beheerders kunnen locaties toevoegen.

## Buiten de scope:

- Het verwerken van betalingen beperkt zich tot het registreren van betalingen en valt buiten de scope van deze applicatie.
- Integratie met externe betalingsproviders of andere externe diensten is niet vereist.

# Kwaliteitscriteria

Functionaliteit (Functional Suitability): De applicatie moet in staat zijn om aan gestelde functionele eisen te voldoen. Dit betekent dat het correct en volledig de gevraagde functies kan uitvoeren. Denk bijvoorbeeld aan: zoekopdrachten, reserveringen, leningen en beheertaken.

Betrouwbaarheid (Reliability): De applicatie moet consistent en betrouwbaar werken, met zo min mogelijk fouten of ander soort problemen. Het moet in staat zijn om te functioneren ook al komt er een fout in voor. denk hierbij aan error handlers

Bruikbaarheid (Usability): De applicatie moet gemakkelijk en fijn te gebruiken zijn, zowel voor bezoekers als personeel. Het moet een duidelijk en overzichtelijk interface hebben, met goede navigatie.

Efficiëntie (Efficiency): De applicatie moet efficiënt gebruikmaken van systeembronnen, zoals geheugen en processorcapaciteit, om optimale prestaties te garanderen. Het moet snel en zonder vertraging taken kunnen uitvoeren.

Onderhoudbaarheid (Maintainability): De applicatie moet eenvoudig te onderhouden en aan te passen zijn. Het moet goed gestructureerde en gescheiden code hebben, met duidelijke documentatie en mogelijkheden voor toekomstige uitbreidingen.

Overdraagbaarheid (Portability): De applicatie moet gemakkelijk te verplaatsen zijn naar verschillende omgevingen. Het moet compatibel zijn met verschillende besturingssystemen, browsers en hardware configuraties.

Beveiliging (Security): De applicatie moet zo veilig mogelijk gebouwd zijn. Het moet maatregelen hebben om ongeoorloofde toegang, gegevensverlies en andere beveiligingsrisico's te voorkomen.

Compatibiliteit (Compatibiliteit): De applicatie moet kunnen samenwerken met andere relevante systemen en standaarden. Het moet gegevens kunnen importeren en exporteren in de juiste formaten en integratie met andere systemen mogelijk maken.

# randvoorwaarden

## Tijdslimieten:

- Het ontwikkelteam heeft een bepaalde periode/semester om het project te voltooien.

## Samenwerking Vereisten:

- Het project wordt uitgevoerd door een groep van twee studenten.
- De samenwerking moet plaatsvinden via een gedeelde GitHub-repository.
- Specifieke afspraken zijn met betrekking tot taakverdeling, versiebeheer en communicatie tussen de teamleden.

## Technologische vereisten:

- Het gebruik van ASP.NET Core vereist een geschikte ontwikkelomgeving, zoals Visual Studio of Visual Studio Code.

## Beperkingen op externe services:

- Het verwerken van betalingen beperkt zich tot het registreren van betalingen en valt buiten de scope van deze applicatie.
- Integratie met externe betalingsproviders of andere externe diensten is niet vereist.

## Toegankelijkheid applicatie:

- De applicatie moet toegankelijk zijn voor bezoekers, personeel en beheerders met verschillende rechten en toegangsrechten, zoals beschreven in de functionele requirements.

## Documentatievereisten:

- comments en documentatie in de code moeten duidelijk vermelden hoe de code werkt.

## Collaborators:

- De docenten moeten als Collaborators worden toegevoegd aan de GitHub-repository voor evaluatie- en beoordeling doeleinden.

# Stakeholders

## Bezoekers:

- Mensen die de bibliotheek bezoeken en gebruik maken van de applicatie om media te zoeken, reserveren en beheren.
- Ze hebben belang bij een gebruiksvriendelijk interface, snelle zoekfunctionaliteit en gemakkelijk toegang tot hun leningen en reserveringen.

## Personeel:

- Medewerkers van de bibliotheek maken gebruik van de  applicatie om media uit te lenen, in te nemen, gegevens te beheren en betalingen te verwerken.
- Ze hebben belang bij versimpelde werkprocessen, een duidelijk interface voor het beheer van media, leningen en reserveringen, en geavanceerde functies zoals het inzien van openstaande posten.

## Beheerders:

- Hoger management of bibliotheek beheerders die verantwoordelijk zijn voor het beheer van de bibliotheek en het monitoren van de activiteiten.
- Ze hebben belang bij uitgebreide toegang tot alle gegevens, het beheren van gebruikersaccounts, het toevoegen en verwijderen van media en auteurs, ze hebben ook belang bij het inzien van rapporten en statistieken.

## Ontwikkelingsteam:

- De studenten of ontwikkelaars die verantwoordelijk zijn voor het ontwerpen, ontwikkelen en implementeren van de applicatie.
- Ze hebben belang bij duidelijke richtlijnen, technische ondersteuning en samenwerkingsmogelijkheden om het project succesvol af te ronden.

## IT-beheer:

- IT-personeel dat verantwoordelijk is voor de infrastructuur, het onderhoud en de beveiliging van de applicatie.
- Ze hebben belang bij een goed opgezette en beveiligde omgeving, regelmatige back-ups en bescherming tegen potentiële bedreigingen.

## Overig:

- Bibliotheekbezoekers en -personeel in andere vestigingen:
- Als de applicatie wordt gebruikt in een bibliotheek met meerdere vestigingen, kunnen medewerkers en bezoekers van andere vestigingen ook belangen hebben.

# Design

## Use cases

Use Case – Admin dashboard

| | |
|---|---|
| Nr | U1.2 |
| Versie nummer | 1 |
| Auteur | Nalany Sturm |
| Use case naam | Admin dashboard |
| Actoren | User |
| Doel (of samenvatting) | Admin dashboard beheren. |
| Triggers | De website kan onderhouden worden. |
| Pre-conditie | Beheren (up-to-date houden) |
| Post-conditie | Website aanpassingen aanmaken. |
| Hoofdscenario | 1. Admin kan inloggen op het dashboard.<br>2.CRUD dashboard. |
| Alternatieve scenario | 1-1 Admin kan niet inloggen.<br>1-2 Admin gebruikt verkeerde wachtwoord.<br>1-3 Admin logt in met juiste wachtwoord. |
| Excepties | De Admin kan geen nieuw wachtwoord aanmaken omdat het wachtwoord niet voldoet aan de eisen. |
| Kwaliteiten | Als het wachtwoord niet voldoet aan de eisen wordt de Admin verzocht om een ander wachtwoord te kiezen die wel voldoet aan de eisen |

Activiteiten Diagram – …

## Test actie 1

1. Admin zit op de website
2. Admin logt in op de website
3. Admin kan niet inloggen
4. Admin gebruikt verkeerde wachtwoord
5. Admin gebruikt juiste wachtwoord

## Test actie 2

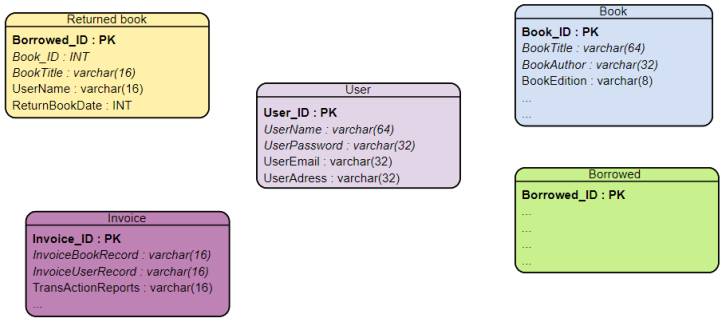
1. Admin zit op de website
2. Admin logt in op de website
3. Admin kan niet inloggen
4. Admin gebruikt verkeerde wachtwoord
5. Admin wachtwoord voldoet niet aan de eisen
6. Admin kan geen nieuw wachtwoord aanmaken
7. Admin nieuw wachtwoord kiezen
8. Admin kan inloggen

## Test actie 3

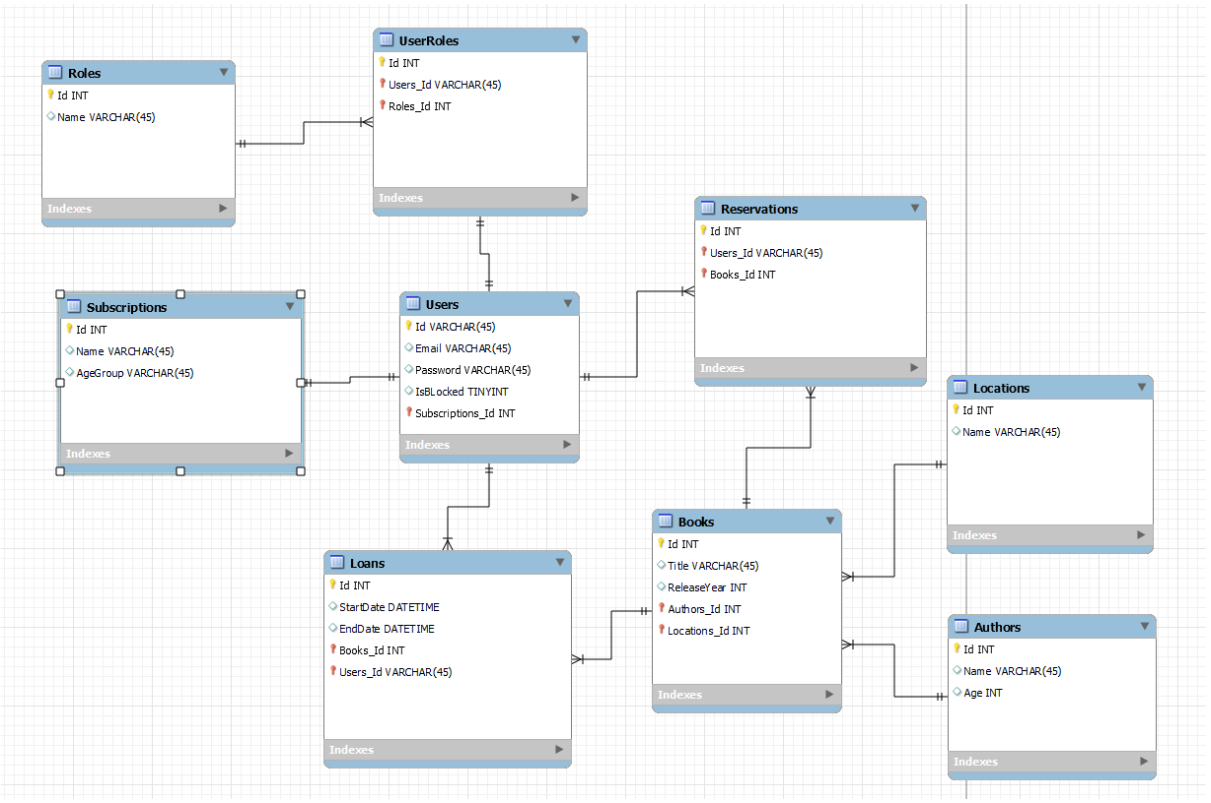
1. Admin zit op de website
2. Admin logt in op de website
3. Admin gebruikt juiste wachtwoord
4. Admin kan de CRUD dashboard beheren

# ERD diagram

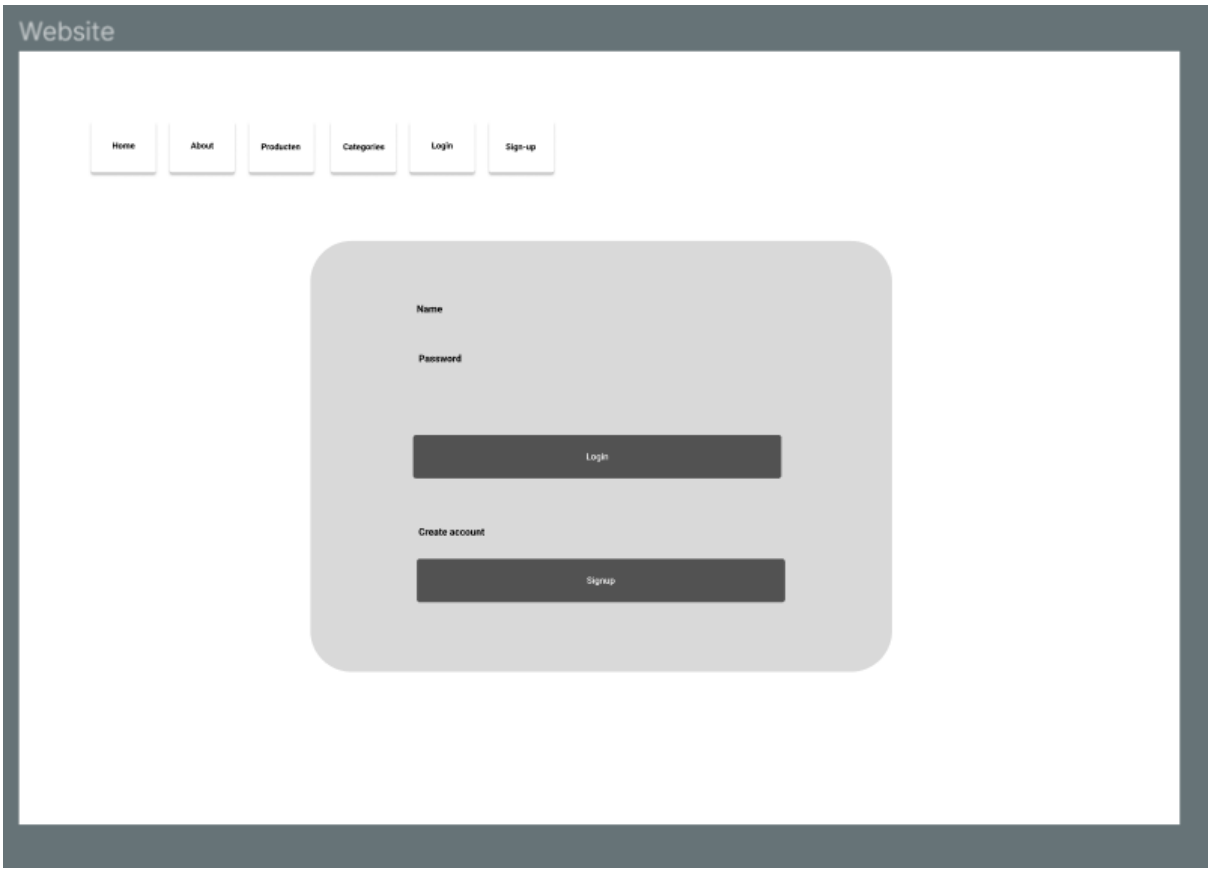
## Versie 1:



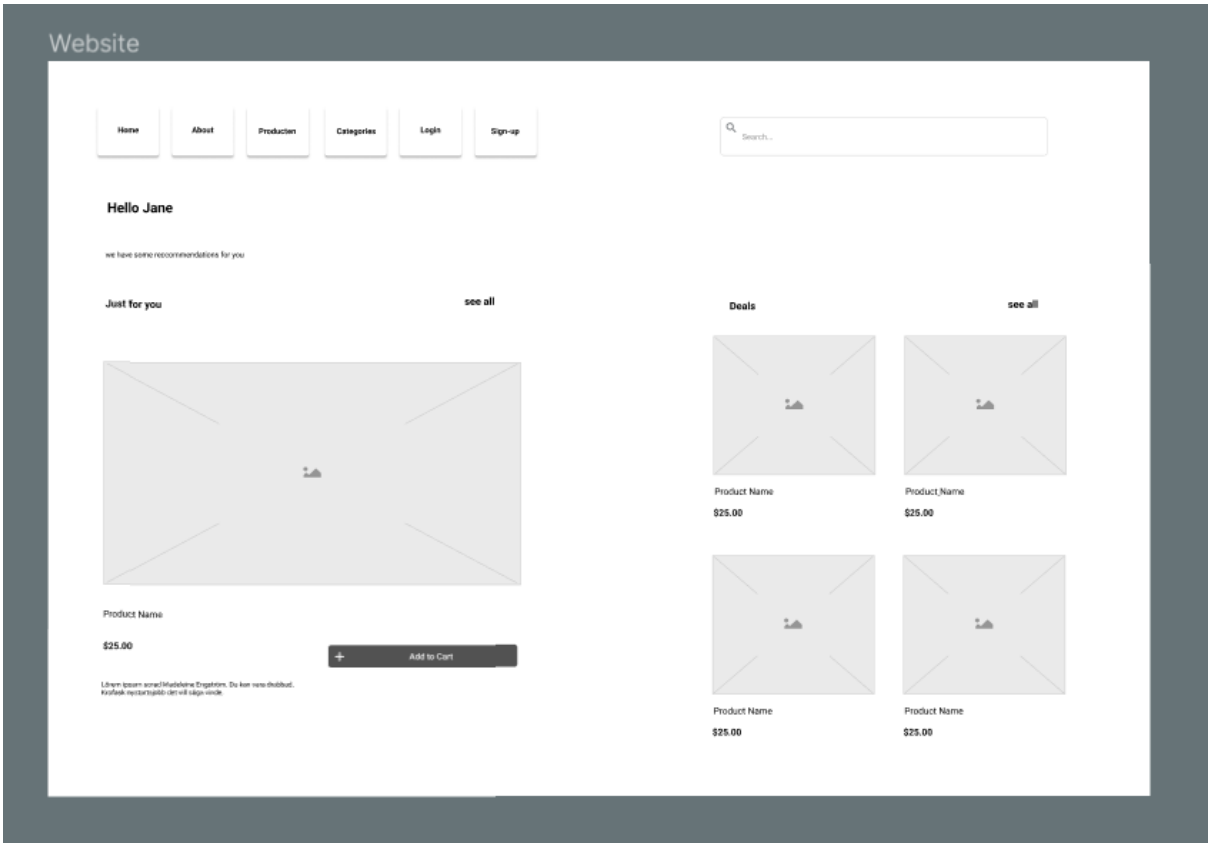

## Versie 2:

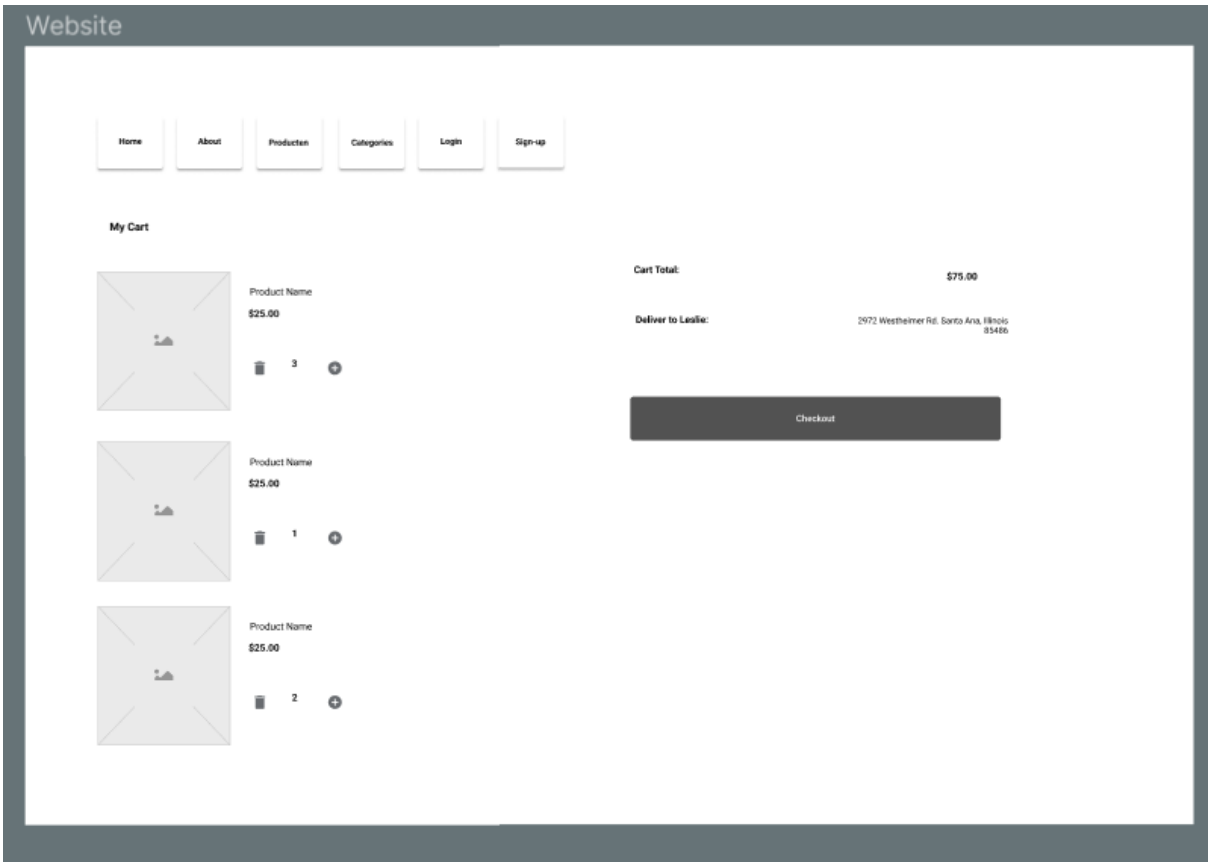


# Wireframes




Inlogpagina versie 1.


Website
Home
About
Producten
Categories
Login
Sign-up
Search...
Hello Jane
we have some recommendations for you
Just for you
see all
Product Name
$25.00
Add to Cart
Deals
see all
Product Name
$25.00
Product Name
$25.00
Product Name
$25.00
Product Name
$25.00


Website
Home
About
Producten
Categories
Login
Sign-up
My Cart
Product Name
$25.00
3
Product Name
$25.00
1
Product Name
$25.00
2
Cart Total:
$75.00
Deliver to Leslie:
2972 Westheimer Rd. Santa Ana, Illinois 85486
Checkout

## *Single Responsibility Principle (SRP)*

Het Single Responsibility Principle (SRP) is een ontwerpprincipe dat richtlijnen biedt voor het schrijven van schone en onderhoudbare code. Het houdt in  dat elke module, klasse of component binnen de software slechts één verantwoordelijkheid heeft.

## *Definitie*

Het SRP vereist dat elke module, klasse of component zich richt op één enkele taak of functie. Dit betekent dat een module verantwoordelijk is voor slechts één specifiek aspect van het systeem.

## *Voordelen*

- begrijpbaarheid: Modules met een enkele verantwoordelijkheid zijn gemakkelijker te begrijpen, omdat hun doel en functie duidelijk zijn gedefinieerd.
- Onderhoudbaarheid: Wanneer een wijziging in een module nodig is, hoeft alleen die specifieke module aangepast te worden. Dit zorgt ervoor dat er maar één aspect wordt aangeplaatst en dat er niet op meerdere plekken iets gewijzigd hoeft te worden.
- *Testbaarheid: Modules met één verantwoordelijkheid zijn gemakkelijker te testen, omdat hun gedrag specifiek en voorspelbaar is.*
- *Minder foutgevoelig: door een module compact te maken  vermindert het risico op fouten, omdat de complexiteit wordt verminderd en de kans op fouten kleiner wordt.*

## Voorbeeld

De applicatie heeft bijvoorbeeld als feature het beheren van klantgegevens en nog een voor het lenen van media. We kunnen het SRP toepassen door een aparte module te maken voor het opslaan van klantinformatie, een andere module voor het uitlenen van media. Elke module heeft een specifieke verantwoordelijkheid en kan onafhankelijk van elkaar worden ontwikkeld, onderhouden en getest.

## *Richtlijnen*

*Om het SRP effectief toe te passen, kunnen de volgende richtlijnen helpen:*

- *denk na over de verschillende taken en verantwoordelijkheden van het systeem.*
- *Zorg ervoor dat elke module slechts één specifieke taak of verantwoordelijkheid heeft.*
- *Houd de modules losgekoppeld van elkaar, zo standalone mogelijk.*
- *Zorg voor een goede naamgeving zodat het duidelijk is welke functionaliteit ze bieden.*

## *Conclusie*

*Het toepassen van het Single Responsibility Principle (SRP) zorgt voor een gestructureerde codebase, het is makkelijker onderhoudbaar en beter te testen. Doordat elke module wordt ontworpen met een duidelijke taak, verbeteren we de code kwaliteit en wordt de code minder complex.*

*Het SRP is een belangrijk principe om te volgen bij het ontwikkelen van software en kan bijdragen aan het succes van het project.*

# Testing

Voor het testen van de bibliotheek applicatie worden de volgende tests gebruikt:

## GUI-testen met Cypress:

Er worden GUI-test cases ontwikkeld en uitgevoerd met behulp van Cypress, een end-to-end test framework.
De GUI-tests zullen verschillende features van de applicatie simuleren, zoals het zoeken van media, reserveringen maken, leningen beheren en betalingen verwerken.
De tests zullen controleren of de applicatie correct reageert op gebruikersinteracties, de verwachte resultaten oplevert en geen fouten of crashes veroorzaakt.

## Unit tests:

Er worden unit tests ontwikkeld voor kritieke onderdelen van de applicatie, zoals services, controllers en modellen.
De unit tests zullen de code van afzonderlijke componenten testen en ervoor zorgen dat ze correct werken volgens de specificaties.

## Testdata:

Er wordt gebruikgemaakt van mock data, zoals nep-gebruikersaccounts en nep-transacties, om te controleren of alle scenario's goed verlopen.
Er worden zowel positieve als negatieve testcases opgesteld om de functionaliteit door grondig te testen.

## Rapportage en opvolging:

Testresultaten worden gedocumenteerd en gerapporteerd en eventuele fouten of bugs die tijdens het testen worden ontdekt worden geregistreerd als issues zodat ze zo spoedig mogelijk worden verholpen.

# Deployment

## GitHub Repository:

- De volledige broncode van de applicatie staat in een GitHub repository.
- De repository bevat alle benodigde bestanden en mappen van het project
- Een kopie van de broncode van het project wordt toegevoegd aan de inlevermap op BrightSpace.