[Functioneel ontwerp](https://sites.google.com/view/rijnijssel-ao/sjablonen/25187/algemeen?authuser=0#h.p_Ty7aduG2GfRB)

Gemaakt door: David Hanna en Levi Hentzepeter

# [Versiebeheer](https://sites.google.com/view/rijnijssel-ao/sjablonen/25187/algemeen?authuser=0" \l "h.p_eKG6nG44Kd0X)

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Versie | Wie | Wijzigingen |
| 20-10-2022 | 1.0 | David Hanna | Document opstellen |
| 20-10-2022 | 1.1 | Joep Zomerveld | Functional Requirements toevoegen |
| 20-10-2022 | 1.2 | David Hanna | Domain en use case diagram toegevoegd |
| 06-11-2022 | 1.3 | David Hanna | Fully dressed use cases 1 tot 4 gemaakt en verbeterde domain diagram toegevoegd |
| 10-11-2022 | 1.4 | Levi Hentzepeter | Fully dressed use cases 5 tot 10 gemaakt |
| 13-12-2022 | 1.5 | David Hanna | Folder structuur en library’s toegevoegd |

Inhoudsopgave

[Versiebeheer 2](#_Toc125489866)

[Inleiding 4](#_Toc125489867)

[Over dit document 4](#_Toc125489868)

[Over het project en de opdrachtgever 4](#_Toc125489869)

[Requirements 5](#_Toc125489870)

[Functional 5](#_Toc125489871)

[Non-functional 5](#_Toc125489872)

[Domain diagram 6](#_Toc125489873)

[Domain diagram beschrijving 7](#_Toc125489874)

[Use cases 8](#_Toc125489875)

[Use case diagram 8](#_Toc125489876)

[Cases 9](#_Toc125489877)

[Use Case: 1. Check lesson materials 9](#_Toc125489878)

[Use Case: 2. Check grades 10](#_Toc125489879)

[Use Case: 3. Check agenda 10](#_Toc125489880)

[Use Case: 4. Manage course 11](#_Toc125489881)

[Use Case: 5. Manage active course 12](#_Toc125489882)

[Use Case: 6. Manage grades 13](#_Toc125489883)

[Use Case: 7. Manage learningpoints 14](#_Toc125489884)

[Use Case: 8. Manage lessons 15](#_Toc125489885)

[Use Case: 9. Manage lesson materials 16](#_Toc125489886)

[Use Case: 10. Check course 17](#_Toc125489887)

[Folder structuur 18](#_Toc125489888)

[Libraries 18](#_Toc125489889)

[Conclusie 20](#_Toc125489890)

# [Inleiding](https://sites.google.com/view/rijnijssel-ao/sjablonen/25187/algemeen?authuser=0#h.p_nScE_Y3RO7c3)

## Over dit document

In dit document zullen we als HBO studenten beschrijven hoe ik een project ga aanpakken dat betrekking heeft op het bouwen van een school website/systeem. Voordat we beginnen met de ontwikkeling van de website, zal er eerst een grondige analyse plaatsvinden van de vereisten en wensen van de school. Dit zal zowel inhouden non-functionele als functionele vereisten.

Vervolgens zal er een projectplan opgesteld worden, waarin de doelstellingen, de benodigde middelen en de verantwoordelijkheden van het project worden vastgelegd. Hierin wordt ook rekening gehouden met eventuele risico's en problemen die zich kunnen voordoen tijdens de ontwikkeling van de website.

De volgende stap is de ontwikkeling van de website zelf. Hierbij wordt gebruik gemaakt van moderne webontwikkeltools en -technologieën, zoals HTML, CSS, JavaScript PHP. Ook maken we gebruik van dingen zoals Symfony , PHPUnit en composer. Deze zullen gemanaged worden via github/actions.

De website zal ontworpen en gebouwd worden met een gebruiksvriendelijke interface en een responsive design zodat de website toegankelijk is op verschillende apparaten.

## Over het project en de opdrachtgever

# Requirements

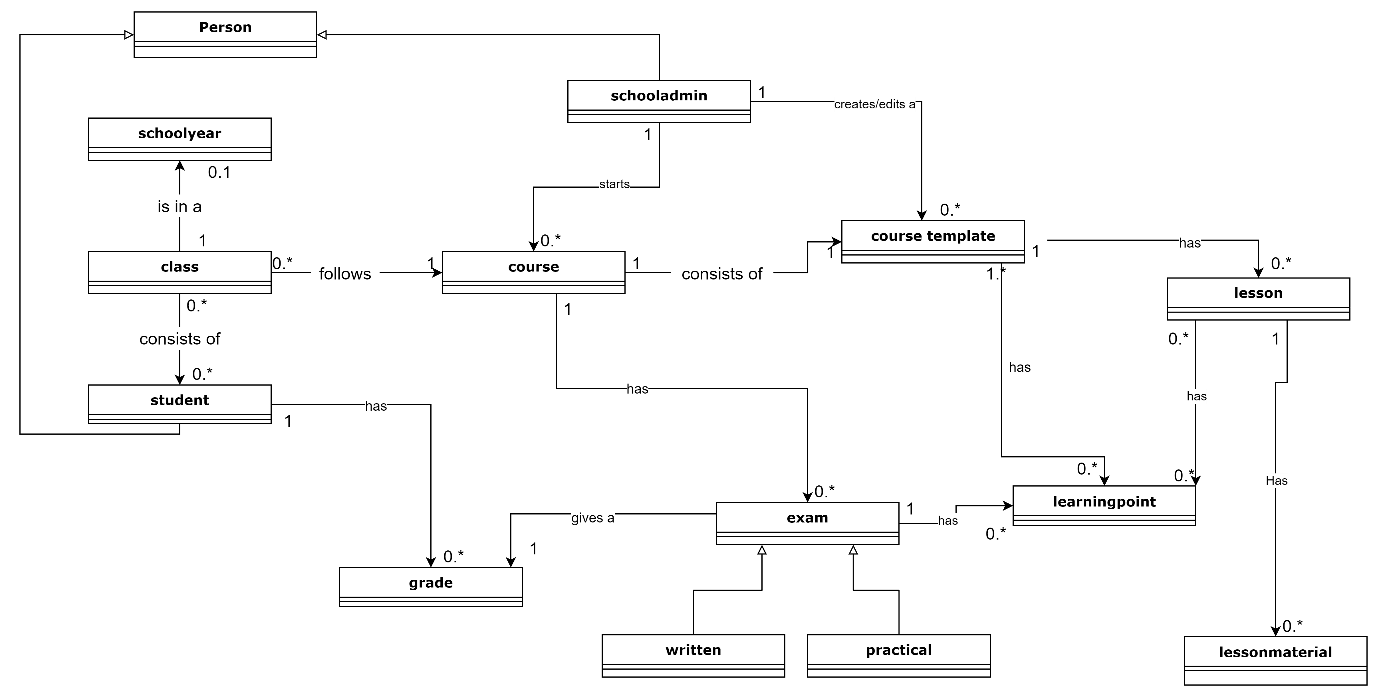
## Functional

1. administrator kan course templates maken en beheren
2. administrator kan een course vanuit een template inplannen (datum en klas toevoegen)
3. een administrator kan (en moet) controleren of de course aan voorwaarden voldoet bij het inplannen
4. een administrator kan lesmaterialen in courses en course templates uploaden en beheren bij elke les
5. een administrator kan leerlingen maken en aan een klas toevoegen
6. een gebruiker kan zijn rooster inzien
7. een leerling gebruiker kan zijn cijferlijst inzien
8. een leraar gebruiker kan cijferlijsten beheren
9. een leerling gebruiker moet de lessen van een course die hij volgt kunnen zien
10. een leerling gebruiker moet de lesmaterialen kunnen downloaden

## Non-functional

1. Compatability in de naam van student gegevens moeten gekoppeld kunnen worden met extrene omgevingen.
2. De website moet bereikbaar zijn binnen 5 seconde ook als er meer dan 1000 man tegelijk opzit
3. De website moet bruikbaar kunnen zijn bij een dataset van 3000 leerlingen
4. De website moet op alle browsers beschikbaar zijn
5. De website moet minimaal 165 uur per week beschikbaar zijn
6. De database moet gebouwd zijn op veiligheid en aan de logische veiligheidseisen voldoen (hashing , sql injector)
7. Minimal score of 70 on google insights test
8. De nieuwste IOS en Android aparaten moeten foutloos met de website werken.
9. Het moet voor de gebruiker binnen een paar klikken mogelijk zijn om alles te vinden.
10. De gebruiker moet zonder uitleg alles kunnen begrijpen.

# Domain diagram



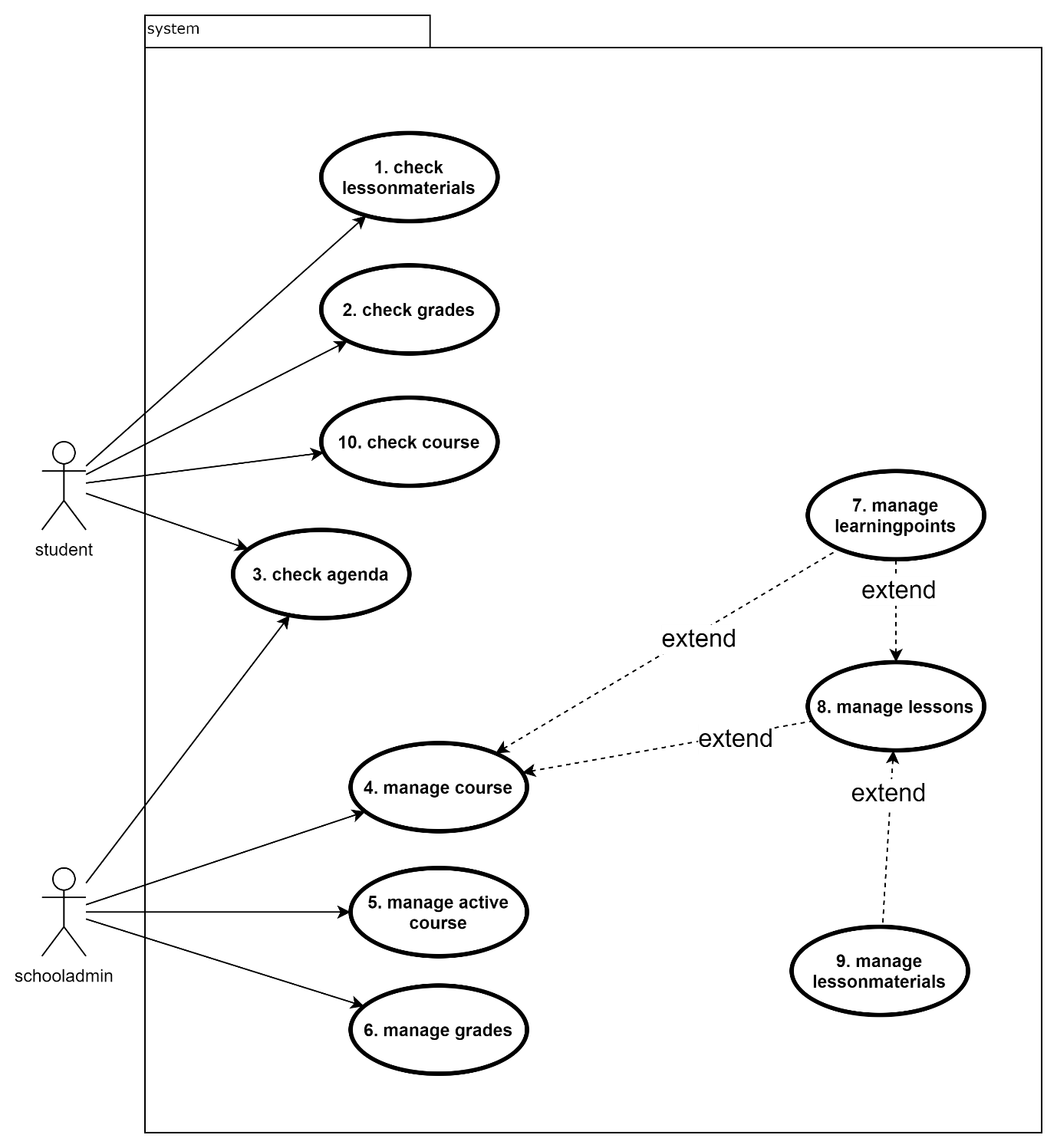
# Domain diagram beschrijving

Hieronder zal een beschrijving voor elk concept worden weergegeven

|  |  |  |
| --- | --- | --- |
| NR | Concept | beschrijving |
|  | Person | In dit concept zullen de basis gegevens voor een persoon worden opgeslagen bijvoorbeeld: leeftijd, naam, geboortedatum, etc. |
|  | Schooladmin | Een school admin is een persoon die klassen en cursussen kan beheren.  Een schooladmin kan een docent zijn maar ook administratief medewerker |
|  | Course template | Een course template is een template van een cursus wat betekent dat deze nog niet vast zit aan een klas en dus ook veranderd kan worden door een schooldadmin.  Aan een course template zitten lesson en learningpoints vast.  Om van een course template een course te maken moeten er voldoende lessons zijn om alle learningpoints te behalen. |
|  | Lesson | Lesson maakt deel uit van een course template en deze heeft ook lessonmaterials.  Een lesson kan een of meer learningpoints hebben die het doel van de les weergeven. |
|  | Lessonmaterial | Lessonmaterials zijn documenten die aan een lesson zijn vastgezet.  Schooladmin’s kunnen lessonmaterials aanmaken en die koppelen aan een les |
|  | Learningpoint | Learningpoints zijn punten en onderwerpen die de leerlingen tijdens de cursus moeten leren. Learningpoints kunnen door de schooladmins aangemaakt worden en aan een les/examen gekoppeld worden |
|  | Exam | Er zijn 2 soorten examens, written en practical. Examens zijn aan een course gekoppeld en hebben learningpoints wat aanduit over welke onderwerpen het examen gaat |
|  | Grade | Schooladmins kunnen grades aan de leerlingen geven de hoogte van de grade wordt bepaald door middel van het examen die de leerling heeft gemaakt |
|  | Student | Student A.K.A. leerling is een persoon die in een klas is gezet en de course volgt |
|  | Class | Class is een groep leerlingen die in een schoolyear zit. De class volgt gezamenlijk een course. |
|  | Schoolyear | Schoolyear duidt de progressie van een class aan. Hoe hoger het jaar hoe moeilijkere courses ze gaan volgen. |
|  | Course | Course is een course template die gevolgd wordt door een class.  Course heeft examens die de leerlingen moeten nemen om zo hun grade voor die course te krijgen.  Een course template kan pas een course worden wanneer deze genoeg lessons heeft om alle learningpoints te behandelen |

# Use cases

## Use case diagram



## Cases

### Use Case: 1. Check lesson materials

|  |  |
| --- | --- |
| Primary actor: student | |
| Brief description:  Student kan de documenten die gekoppeld zijn aan een les downloaden in verschillende vormen als dat nodig is. | |
| Preconditions:  Actor is ingelogd als student.  Actor weet van te voren van welke Course en les hij de documenten van wilt downloaden.  Actor is op de Course list pagina | |
| Postconditions on success:  Student heeft de documenten gedownload  Postconditions on failure:  Student maakt niet deel uit van een Course of heeft geen beschikbare documenten | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op de Course waarvan hij de documenten van wilt downloaden.  3.Actor klikt op de les pagina waarvan hij de documenten van wil downloaden  5.Actor klikt op de download knop van de documenten en geeft aan in welke file typen hij ze wilt ontvangen | 2.systeem laat de info pagina van de Course zien  4.systeem laat de info pagina van de les zien  6.systeem converteer de documenten in de gespecificeerde file typen  7.systeem download de documenten |
| Alternate flows: (Variations with successful UC) | |
| 5A.[Actor klikt op download origineel]  5A1.Ga naar stap 7 |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 1A.[Actor is nog niet toegevoegd aan een Course]  1A1.systeem laat een fout melding zien | 2A.[Er zijn geen lessen aan de Course toegevoegd]  2A1.systeem laat foutmelding zien.  4A.[Er zjn geen documenten aan de les toegevoegd]  4A1.systeem laat foutmelding zien. |

### Use Case: 2. Check grades

|  |  |
| --- | --- |
| Primary actor: student | |
| Brief description:  Met deze use case zou de student zijn cijfers kunnen bekijken die door de schooladmins zijn ingeschreven. | |
| Preconditions:  Actor is als student ingelogd.  Actor weet welke cijfers hij wil bekijken | |
| Postconditions on success:  Student heeft zijn cijfers bekeken  Postconditions on failure:  Student heeft nog geen cijfers | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op de knop cijfers bekijken  3.Actor bekijkt zijn cijfers | 2.system laat alle cijfers van de Actor zien |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
|  | 2A.[Actor heeft geen cijfers]  2A1.Systeem laat een foutmelding zien |

### Use Case: 3. Check agenda

|  |  |
| --- | --- |
| Primary actor: student | |
| Brief description:  Student bekijkt zijn agenda om te controleren wanneer zijn volgende les is. | |
| Preconditions:  Actor is als student ingelogd | |
| Postconditions on success:  Student bekijkt wanneer zijn volgende les begint.  Postconditions on failure:  Student heeft geen beschikbare lessen | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op de agenda knop  3.Actor bekijkt zijn agenda | 2.systeem laat de agenda met de info van de lessen zien |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
|  | 2A.[Actor heeft geen beschikbare lessen om te bekijken]  2A1.Systeem laat foutmelding zien |

### Use Case: 4. Manage course

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Brief description:  Schooladmin creëert een course en koppelt deze aan een klas. | |
| Preconditions:  Actor is ingelogd als schooladmin.  Actor is op de course lijst pagina | |
| Postconditions on success:  Course is gecreëerd.  Postconditions on failure:  Course is niet gecreëerd. | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op de knop creëer nieuwe course knop  3.Actor vult de course info in.  4.Actor voegt leerdoelen aan de course toe.  5.Actor voegt lessen aan de course toe.  6.Actor koppelt de leerdoelen en de lessen met elkaar.  7.Actor voegt examens toe.  8.Actor klikt op opslaan. | 2.systeem laat de course create pagina zien  10.system creëert een nieuwe course |
| Alternate flows: (Variations with successful UC) | |
| 4A.[actor voegt geen leerdoelen aan de course]  4A1.naar stap 5.  4A2.stap 6 moet overgeslagen worden. | 10A.[niet alle leerdoelen worden behandelt in een les]  10A1.systeem laat foutmelding zien  10A2.terug naar stap 6 |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 9A.Actor klikt niet op opslaan |  |

### Use Case: 5. Manage active course

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Brief description:  Schooladmin verandert een course. | |
| Preconditions:  Actor is ingelogd als schooladmin  Actor weet welke course hij wilt veranderen  Actor is op de course lijst pagina | |
| Postconditions on success:  Course is veranderd  Postconditions on failure:  Course is niet veranderd | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op de edit pagina van de course  3.Actor vult zijn veranderingen in.  4.Actor klikt op opslaan | 2.systeem laat de huidige course info zien  5.systeem checkt of alle leerdoelen nog worden behandelt in een les.  6.system slaat de course op |
| Alternate flows: (Variations with successful UC) | |
|  | 5A.[niet alle leerdoelen worden behandelt in een les]  5A1.systeem laat foutmelding zien  5A2.terug naar stap 3 |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 3A.Actor verandert niks aan de course  6A.Actor klikt niet op opslaan |  |

### 

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Brief description:  Schooladmin koppelt een course aan een klas. | |
| Preconditions:  Actor is ingelogd als schooladmin.  Actor weet welke course en klas hij met elkaar wil koppelen.  Actor is op de course list pagina | |
| Postconditions on success:  De course en klas zijn gekoppeld.  Postconditions on failure:  De course en klas zijn niet gekoppeld | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op course aanmaken knop  3.Actor klikt de gewenste klas en course aan  4.Actor klikt op opslaan | 2.systeem laat de Koppel pagina zien  5.Systeem controleert of alle leerdoelen van de course in een les worden behandelt  6.systeem koppelt de klas en course aan elkaar |
| Alternate flows: (Variations with successful UC) | |
|  | 5A.[niet alle leerdoelen worden behandelt in de les]  5A1. Systeem laat error melding zien.  5A2. Terug naar stap 3. |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 3A.er zijn geen klassen beschikbaar  3B.er zijn geen courses beschikbaar  6A.Actor klikt niet op opslaan |  |

### Use Case: 6. Manage grades

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Brief description:  Schooladmin voegt cijfer toe aan een student | |
| Preconditions:  Actor moet ingelogd zijn als schooladmin  Actor weet welke cijfers hij van welke student hij wil veranderen en in welke klas deze student is ingeschreven.  Actor is op de info pagina van de klas. | |
| Postconditions on success:  Cijfer is toegevoegd/gewijzigd  Postconditions on failure:  Cijfer is niet toegevoegd | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op de gewenste student  3.Actor vult het nieuwe cijfer in  4.Actor klikt op opslaan | 2.systeem laat info pagina van de student zien  5.system update het cijfer van de student |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 4A.Actor vult het nieuwe cijfer niet in 5A.Actor klikt niet op opslaan |  |

### Use Case: 7. Manage learningpoints

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Brief description:  Schooladmin kan leerdoelen toevoegen aan een course | |
| Preconditions:  actor is ingelogd als schooladmin  actor is in de edit pagina van de course | |
| Postconditions on success:  Course leerdoelen zijn geupdate  Postconditions on failure:  Course leerdoelen zijn niet geupdate | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op de creëer nieuwe leerdoel knop  3.Actor vult de informatie van de nieuwe leerdoel in  4.Actor klikt op opslaan | 2.systeem laat de creëer leerdoelen pagina zien  5.systeem slaat de nieuwe leerdoel op |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 3A.Actor vult niks in  4A.Actor klikt niet op opslaan |  |

### Use Case: 8. Manage lessons

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Brief description:  Schooladmin kan nieuwe les aanmaken aan een course | |
| Preconditions:  Actor is ingelogd als admin.  Actor is in de edit course pagina | |
| Postconditions on success:  Les is aangemaakt  Postconditions on failure:  Les is niet aangemaakt | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op creëer nieuwe les knop  3.Actor vult de gegevens in van de nieuwe les  4.Actor koppelt leerdoelen aan de les.  5.Actor upload learningmaterials voor de les  6.Actor klikt op opslaan | 2.systeem laat de creëer nieuwe les pagina zien  5.systeem slaat de nieuwe les op |
| Alternate flows: (Variations with successful UC) | |
| 4A.[Actor koppelt geen leerdoelen]  4A1.ga door met stap 5.  5A.[Actor upload geen materialen]  5A1.ga door met stap 6 |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 6A.Actor klikt niet op opslaan |  |

### Use Case: 9. Manage lesson materials

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Brief description:  Schooladmin update de les materialen van een les. | |
| Preconditions:  actor is ingelogd als schooladmin.  Actor is op de edit pagina van de les | |
| Postconditions on success:  Les materialen zijn geupdate.  Postconditions on failure:  Les materialen zijn niet geupdate. | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op upload lesson materials knop  3.Actor kiest de gewenste file die hij wil uploaden  4.actor klikt op uploaden | 2.systeem opent de file explorer  5.systeem upload de file naar de server |
| Alternate flows: (Variations with successful UC) | |
|  | 5A.[er bestaat al een document met dezelfde naam op de server]  5A1.systeem laat fout melding zien.  5A2.terug naar stap 3. |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 4A.Actor klikt niet op uploaden |  |

### Use Case: 10. Check course

|  |  |
| --- | --- |
| Primary actor: student | |
| Brief description:  Student kan zijn de info van zijn cursus bekijken. | |
| Preconditions:  Actor is ingelogd als student.  Actor weet welke course hij wil bekijken | |
| Postconditions on success:  Student heeft zijn course gegevens bekeken  Postconditions on failure:  Geen beschikbare course | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor klikt op de courses knop  3.Actor klikt op de gewenste course  5.Actor bekijkt zijn course | 2.systeem laat een pagina met een lijst van courses zien  4.systeem laat pagina met course info zien |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
|  | 2A.actor is nog niet gekoppeld aan een course |

# Folder structuur

De basis structuur van ons project ziet er ongeveer als volgt uit:

* Api
  + Models
  + config
* Tests
  + Api tests
  + Website tests
* Website
  + Models
  + Views
  + Controllers

**Website**

Zoals je ziet hebben wij ervoor gekozen om de MVC pattern te gebruiken voor onze website. Dit hebben we gedaan omdat wij hier allebei ervaring mee hebben en omdat het goed de lagen van de website gescheiden houd.

In de Models folder zal er de abstract pattern toegepast worden. We gebruiker hier de abstract pattern zodat we de basis functies van te voren kunnen definiëren en ze dus niet voor elke model apart hoeven te maken.

De abstract pattern zal met een api controller in de controllers folder communiceren om zo data vanuit de api op te halen en via de models naar de desbetreffende controller te sturen.

In de controllers folder zal er een page controller komen. De functie van de page controller is om de data van een view op te pakken uit de models/controllers en deze data dan samen met de view aan de gebruiker te laten zien.

**Api**

In de models folder van de api komt ook een abstract pattern. Deze abstract pattern wordt gebruikt om standaard queries voor de models vast te stellen zodat deze niet voor elke model apart gemaakt hoeft te worden.

In de config folder zal de connectie met de database gemaakt worden.

In de root folder van de api zal de index.php elke request oppakken naar de api en deze naar de correcte model sturen.

**Tests**

In de api tests folder zullen alle tests die behoren tot de api gezet worden.

In de website tests folder zullen alle tests die behoren tot de website gezet worden.

# Libraries

De volgende libraries zullen aanwezig in ons project:

* Symfony router
* Symfony http foundation
* Phpword
  + Dompdf
* phpunit

**symfony**

in ons project gaan wij 2 libraries van symfony gebruiken.

De eertse librarie is de router. De router zal gebruikt worden om alle pagina’s in ons project in een lijst te stoppen en door middel van de url wordt dan bepaald welke pagina moet worden weergegeven.

De tweede librarie is de http foundation. Deze library wordt gebruikt om de get en post parameters te pakken en deze te integreren met de router.

**Phpword**

De phpword librarie gaan we gebruiken om de documenten die door de gebruikers worden geupload te converteren naar andere file types zo zou het dus mogelijk zijn om door een gebruiker geuploade word bestand te converteren naar een pdf zodat deze weer door een andere gebruiker kan worden gedownload. Helaas word het converteren van word bestanden naar pdf niet door phpword zelf ondersteund, daarom hebben wij de librarie dompdf nodig om samen met phpword de bestanden te converteren.

**Phpunit**

Phpunit wordt gebruikt om makkelijk en snel unit tests van php functies te maken en deze ook meteen te testen.

# [Conclusie](https://sites.google.com/view/rijnijssel-ao/sjablonen/25187/algemeen?authuser=0#h.p_nScE_Y3RO7c3)

In conclusie kan worden gesteld dat het bouwen van een school website/systeem een complex project is waarbij verschillende aspecten in aanmerking moeten worden genomen. Door een grondige analyse van de vereisten en wensen van de school, wordt er een duidelijk beeld verkregen van de eisen waaraan de website moet voldoen. Hierdoor kan een gericht projectplan worden opgesteld waarin de doelstellingen, benodigde middelen, verantwoordelijkheden en planning worden vastgelegd. Dit zorgt voor een efficiënte en effectieve aanpak van het project.

Tijdens de ontwikkeling van de website wordt gebruik gemaakt en programmeer talen waar nog niet het hele groepje in gewerkt heeft. Door gebruik te maken van deze technologieën, kan een website worden gebouwd met een gebruiksvriendelijke interface en een responsive design, waardoor de website toegankelijk is op verschillende apparaten.

Na voltooiing van de ontwikkeling, is een grondige testfase van groot belang. Hierbij wordt de website grondig getest op technische en functionele aspecten dit om te zorgen voor een hoog kwaliteit eindproduct.

Tot slot is de implementatie en het beheer van de website van groot belang. Met dit als doel hebben we in de complete website een logisch systeem gebouwd waardoor elke programmeur direct de logica zal zien en onderhouden heel makkelijk zal worden.

In samenvatting, het bouwen van een school website/systeem vereist een grondige analyse, een uitgebreid projectplan, de ontwikkeling van een moderne en gebruiksvriendelijke website, een grondige testfase en implementatie en beheer van de website. Door deze stappen grondig en methodisch aan te pakken, kan een website worden gebouwd die aan de eisen van de school voldoet en waar gebruikers gemakkelijk mee kunnen werken, en de website zal veilig, performant en betrouwbaar zijn.