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

# [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](#_Toc125402411)

[Inleiding 4](#_Toc125402412)

[Over dit document 4](#_Toc125402413)

[Over het project en de opdrachtgever 4](#_Toc125402414)

[Requirements 5](#_Toc125402415)

[Functional 5](#_Toc125402416)

[Non-functional 5](#_Toc125402417)

[Domain diagram 6](#_Toc125402418)

[Domain diagram beschrijving 7](#_Toc125402419)

[Use cases 8](#_Toc125402420)

[Use case diagram 8](#_Toc125402421)

[Cases 9](#_Toc125402422)

[Use Case: 1. Check lesson materials 9](#_Toc125402423)

[Use Case: 2. Check grades 9](#_Toc125402424)

[Use Case: 3. Check agenda 10](#_Toc125402425)

[Use Case: 4. Manage course 11](#_Toc125402426)

[Use Case: 5. Manage active course 12](#_Toc125402427)

[Use Case: 6. Manage grades 14](#_Toc125402428)

[Use Case: 7. Manage learningpoints 15](#_Toc125402429)

[Use Case: 8. Manage lessons 16](#_Toc125402430)

[Use Case: 9. Manage lesson materials 16](#_Toc125402431)

[Conclusie 18](#_Toc125402432)

# [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

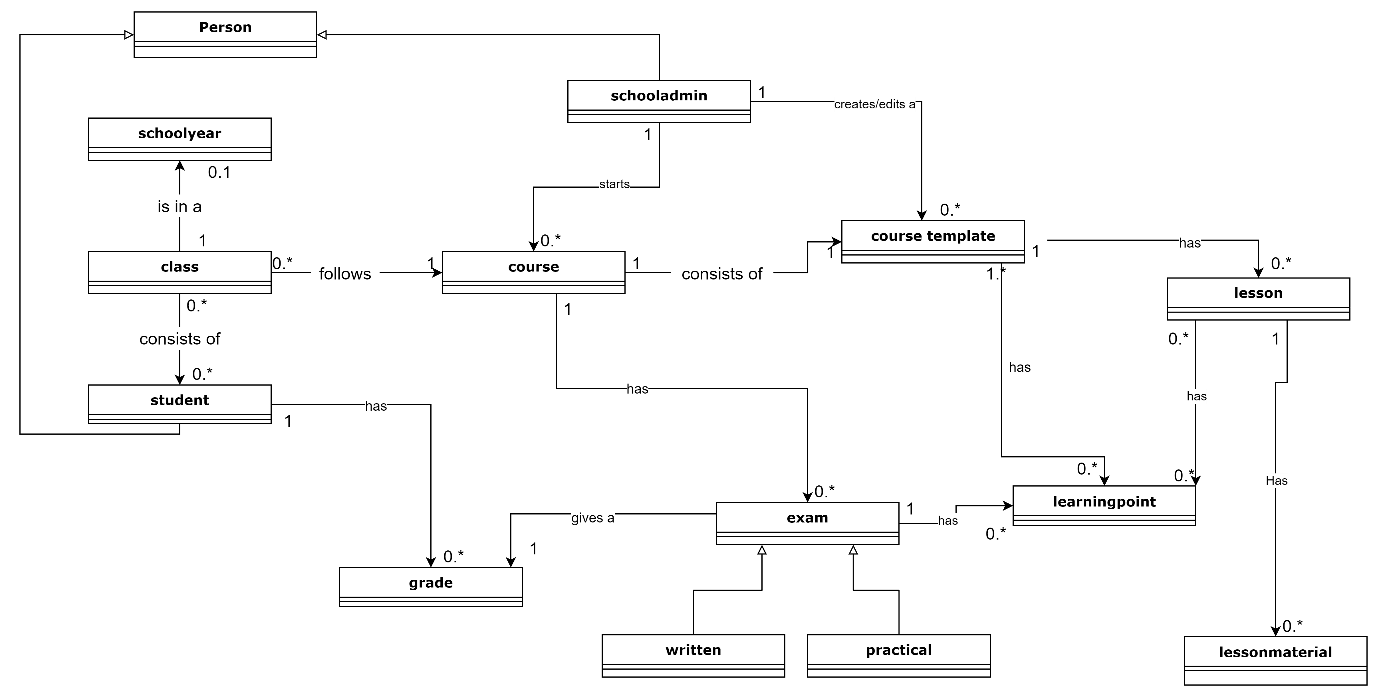
## Functional

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. Latest IOS and Android device should have no problem running the website
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



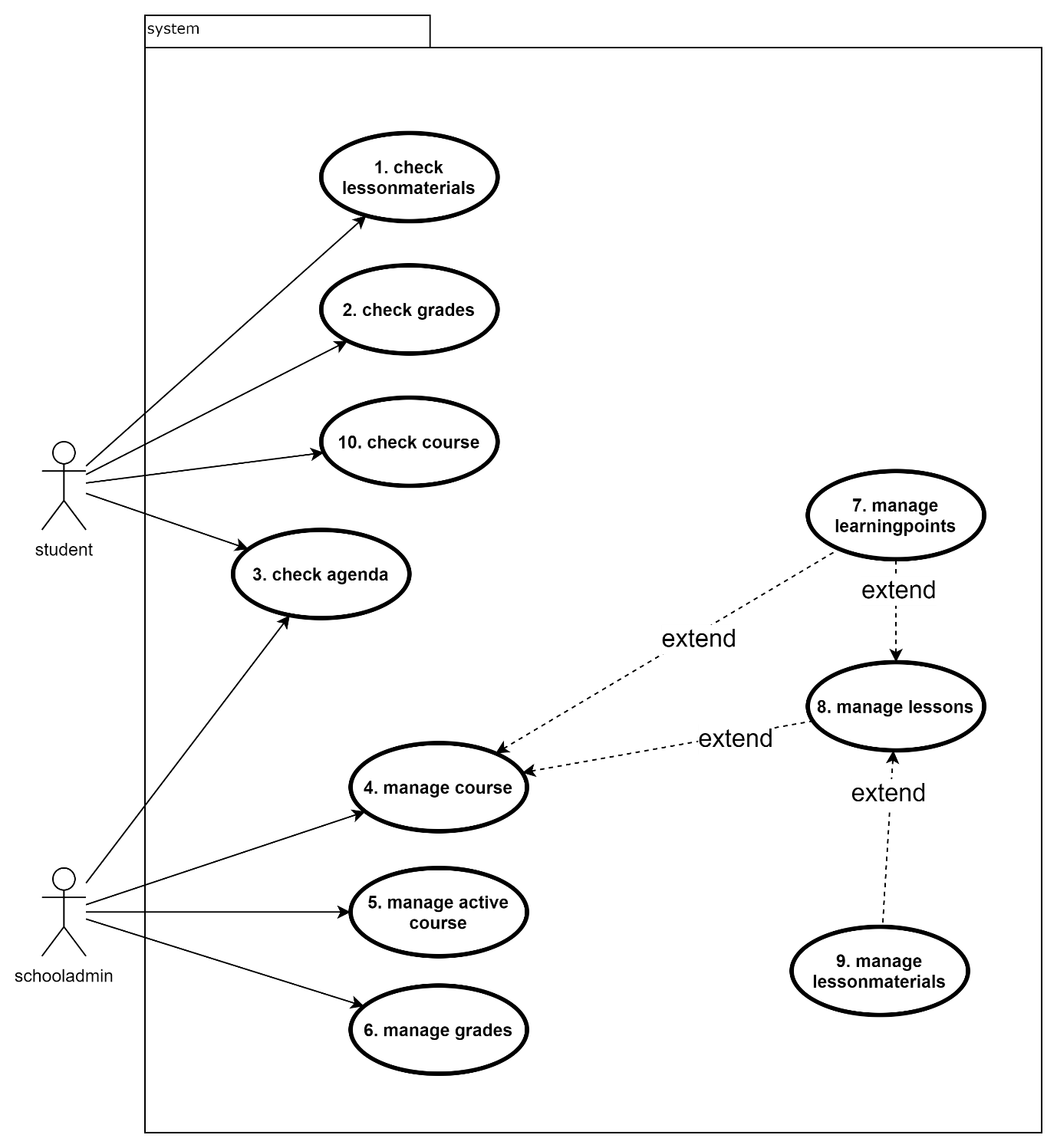
# Domain diagram beschrijving

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 | |
| Stakeholders and Interests: student who wants to check the materials available for a lesson | |
| Brief description:  Students check materials that are available for the lessons that they follow | |
| Preconditions:  Actor is logged in as student.  Actor knows which course and lesson he wants to check the materials from and is already on the check course page. | |
| Postconditions on success:  Student has checked the desired lesson materials  Postconditions on failure:  Student has no courses or materials for the lesson | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the course where he wants to check the lesson materials from  3.Actor clicks the desired lesson  5.actor clicks download on the desired documents | 2.system shows actor’s lesson for specified course  4.system shows actor’s lessonmaterials for specified lesson  6.system download documents |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 1A.Actor has not been added to any course yet | 2A.actor has no lessons for specified course  4A.no lesson materials have been added to the specified lesson |

### Use Case: 2. Check grades

|  |  |
| --- | --- |
| Primary actor: student | |
| Stakeholders and Interests: student who wants to check his grades | |
| Brief description:  Students checks grades that he has received for his courses | |
| Preconditions:  Actor is logged in as student.  Actor knows which grades he wants to check. | |
| Postconditions on success:  Student has checked grades  Postconditions on failure:  Student has no grades | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the course where he wants to check the grades  3.Actor checks his grades | 2.system shows actor’s grades of the specified course |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 1A.Actor has not been added to any course yet | 2A.actor has no grades for specified course |

### Use Case: 3. Check agenda

|  |  |
| --- | --- |
| Primary actor: student | |
| Stakeholders and Interests: student who wants to check his agenda | |
| Brief description:  Students checks his agenda to see which lessons he has and when they take place | |
| Preconditions:  Actor is logged in as student | |
| Postconditions on success:  Student knows when his upcoming lessons are  Postconditions on failure:  Student has no upcoming lessons | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the agenda button  3.Actor checks his agenda | 2.system shows actor’s agenda with lesson details |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
|  | 2A.there are no upcoming lessons |

### Use Case: 4. Manage course

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Stakeholders and Interests: schooladmin creates a course | |
| Brief description:  Schooladmin creates a course | |
| Preconditions:  actor is logged in as schooladmin | |
| Postconditions on success:  Course is created  Postconditions on failure:  Course is not created | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the create course button  3.Actor fills course title in.  4.Actor fills course description in  5.Actor adds lessons to course.  6.Actor adds exams to course.  7.Actor adds learningpoints to lessons and exams  8.Actor clicks save | 2.system shows course creation page  9.system creates new course |
| Alternate flows: (Variations with successful UC) | |
| 5A.Actor does not add lessons to course  6A.actor does not add exams to course.  7A. Actor does not add learningpoints to lessons and exams |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 8A.Actor does not click save |  |

### Use Case: 5. Manage active course

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Stakeholders and Interests: schooladmin edits a course | |
| Brief description:  Schooladmin edits a course | |
| Preconditions:  Actor is logged in as schooladmin  Actor knows which course he wants to change | |
| Postconditions on success:  Course is edited  Postconditions on failure:  Course is not edited | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the edit course button  3.Actor specifies preferred course  5.Actor changes course details  6.Actor clicks save | 2.system shows list of available courses  4.system shows details of specified course  7.system saves course |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 5A.Actor does not change course details  6A.Actor does not click save |  |

### 

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Stakeholders and Interests: add class to course | |
| Brief description:  Schooladmin adds a course to a class | |
| Preconditions:  Actor is logged in as schooladmin  Actor knows which course he wants to assign to which class | |
| Postconditions on success:  Course is added to class agenda  Postconditions on failure:  Course is not added to class | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on preferred class  3.Actor clicks preferred course  5.Actor specifies start date  6.Actor clicks save | 2.System shows which courses are available for the class  4.System checks if course details are complete  7.System adds course to class |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 6A.Actor does not click save | 2A.no courses available  4A.Course details are not complete |

### Use Case: 6. Manage grades

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Stakeholders and Interests: schooladmin creates a grade | |
| Brief description:  Schooladmin creates a grade for a specific student / course | |
| Preconditions:  Actor must be logged in as either student (can only view) or schooladmin (can view, edit and create)  student must follow a course | |
| Postconditions on success:  Student and schooladmin can check the grade  Postconditions on failure:  Schooladmin gets an error message | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the fill grades button  3.Actor chooses student  4.Actor fills in the grade  5.Actor clicks save | 2.system shows page with all student of that course  6.system updates the students grade |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 4A.Actor does not fill in the grade 5A.Actor does not click save |  |

### Use Case: 7. Manage learningpoints

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Stakeholders and Interests: schooladmin manages the learningpoints of a course | |
| Brief description:  Schooladmin manages the learningpoints of a course | |
| Preconditions:  actor is logged in as schooladmin  actor is in the edit page for the desired course | |
| Postconditions on success:  Course learningpoints are updated  Postconditions on failure:  Course learningpoints are not updated | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the the learningpoints button  3.Actor updates learningpoints to lessons and exams  4.Actor clicks save | 2.system shows course creation page  5.system updates the course learningpoints |
| Alternate flows: (Variations with successful UC) | |
| 3A. Actor does not add learningpoints to lessons and exams |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 4A.Actor does not click save |  |

### Use Case: 8. Manage lessons

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Stakeholders and Interests: schooladmin manages the lessons of a course | |
| Brief description:  Schooladmin manages the lessons of a course | |
| Preconditions: actor is logged in as schooladmin | |
| Postconditions on success:  Course lessons are updated  Postconditions on failure:  Course lessons are not updated | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the update lessons button  3.Actor updates lessons to course.  4.Actor clicks save | 2.system shows course creation page  5.system creates new course |
| Alternate flows: (Variations with successful UC) | |
| 3A.Actor does not add lessons to course |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 4A.Actor does not click save |  |

### Use Case: 9. Manage lesson materials

|  |  |
| --- | --- |
| Primary actor: schooladmin | |
| Stakeholders and Interests: schooladmin update lesson materials from a course | |
| Brief description:  Schooladmin update lesson materials from a course | |
| Preconditions: actor is logged in as schooladmin | |
| Postconditions on success:  lesson materials is updated  Postconditions on failure:  lesson materials is not updated | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the edit lesson materials button  3.Actor fills in the lesson material fields which he wants to update  4.Actor clicks save | 2.system shows lesson materials edit page  5.system creates new course |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
| 4A.Actor does not click save |  |

### Use Case: 10. Check course

|  |  |
| --- | --- |
| Primary actor: student | |
| Stakeholders and Interests: student can see his upcoming course and the lessons | |
| Brief description:  The student can check his course with the lessons, learningpoints and exams that are a part of this course | |
| Preconditions: actor is logged in as student | |
| Postconditions on success:  Student has checked his course  Postconditions on failure: | |
| Main Success Scenario (Basic Flow): | |
| Actor Action | System Responsibility |
| 1.Actor clicks on the course he wants to see the info of.  3. Actor sees the course | 2.system shows the view page for the course |
| Alternate flows: (Variations with successful UC) | |
|  |  |
| Exceptional flows: (UC failure leading to “post condition on failure”) | |
|  |  |

# 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
* 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.

**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.