Examenonderlegger jan '26 K1

Beoordelingsformulier: BF_SD_SD20_B1-K1_3

naar Werkproces 2

B1-K1-W1

Plant werkzaamheden en bewaakt de voortgang

Rubics

1. De Uitgangspunten, technische en functionele eisen en wensen zijn bepaald en gedocumenteerd.

Onderlegger uitgangspunten, eisenen en wensen

Definities

1. *Uitgangspunten (minimaal 5)*
   Kaders, radwvoorwaarden, eisen of aannames die een globale scope hebben.
2. *Functionalele eisen (minimaal 12)*
   Beschrijving van wat het systeem moet doen vanuit gebruikersperspectief.
3. *Technische eisen (minimaal 5)*
   Beschrijven architectuur, frameworks, datastromen, beveiliging, perfomance, data strucuren, database ontwerp, etc.
   Deze eigen beinvloeden de technische uitvoering maar beschrijven geen functionaliteiten.
4. *Uitgangspunten, en eisen voldoen aan:*
   Relevantie, specifiek, controleerbaar/meetbaar, consistent, herleidbaar (bron/waarom).

Samengevat

Term	Gericht op	Vraag die het beantwoordt	Testbaar?
Uitgangspunt	Kaders en randvoorwaarden	*Waar moeten we rekening mee houden?*	Indirect / randvoorwaardelijk

Term	Gericht op	Vraag die het beantwoordt	Testbaar?
Functionele eis	Functionaliteit	*Wat moet het systeem doen?*	Ja, via testcases
Technische eis	Technische uitvoering	*Hoe moet het systeem technisch functioneren?*	Ja, meetbaar

1. Je benoemd kort, puntsgewijs minimaal 4 uitgangspunten. Elke uitgangspunt is onderbouwd (waarom) en het is duidelijk waar het uitgangspunt vandaan komt.
2. Je benoemd kort, puntsgewijs minimaal 12 functionele eisen. Elke eis beschrijft observeerbaar gedrag van de software (je kan het zien) en is testbaar.
3. Je benoemd kort, puntsgewijs mininaal 5 technische eisen. Elke eis is concreet (en dus controleerbaar) en elke eis is onderbouwd (waarom).

2. Op basis van de functionaliteit is een complete en realistische planning gemaakt.

Onderlegger planning

1. **Alle functionele** eisen komen terug in de planning
2. Elke functionaliteit is opgesplitst in concrete taken van **max. 4 uur** per taak
3. Er is per **taak** aangegeven op welk onderdeel van de functionaliteit deze betrekking heeft
4. Bij elke taak staat een **tijdsinschatting** in uren (geheel of halve uren).
5. Totoale ontwikkeltijd is **minimaal 40 uur**, waarvan ongeveer 20% testen en 10% documenteren. Je neemt naast de 40 geplande uren nog een extra 4-8 uur op voor onvoorziene omstandigheden (dus totaal 44 uur).
6. De **volgorde** is logisch (chronolgisch).
7. De planning is **concreet**, **testbaar** (eenduidig).

3. De gestelde doelen en planning zijn bewaakt.

Onderlegger bewaking

(bewaking wordt pas uitgevoerd vanaf werkproces 3).

1. Voortgang is gedurende de gehele planning minimaal 5 maal bijgehouden.

2. Voortgang bevat een status: wat is af en wat had moeten zijn en wat is de afwijking.
3. Bij elke afwijking wordt ene actie genomen en beschreven.
4. Aan het eind is een evaluatei/reflectie opnomen.

Checklist

	Criterium	Nee (0)	Beetje (1)	Ja (2)
1	**Uitgangspunten, eisen en wensen (6, 4, 2)**			
1.1	Er zijn minimaal 5 uitgangspunten (kaders/randvoorwaarden) benoemd. Ze zijn onderbouwd en de bron is duidelijk.			
1.2	Er zijn minimaal 12 functionele eisen benoemd. Deze beschrijven observeerbaar gedrag en zijn testbaar.			
1.3	Er zijn minimaal 5 technische eisen benoemd. Deze zijn concreet (controleerbaar) en onderbouwd.			
2	**Planning (10, 6, 3)**			
2.1	Alle functionele eisen komen terug in de planning.			
2.2	Taken zijn opgesplitst (max. 4 uur per taak) en gekoppeld aan een specifiek onderdeel van de uitgangspunten, eisen of wensen (1.1, 1.2, 1.3). Taken zijn eenduidig beschreven.			

2.3	De totale ontwikkeltijd is minimaal 40 uur (deze 40 uur is gebaseerd op niveau dat verwacht wordt van een MBO-4 examenkandidaat).			
2.4	De planning is logisch en chronologisch van opbouw, concreet en testbaar.			
2.5	In de planning zijn minimaal 5 overleg/ voortgangs momenten opgenomen.			
3	**Bewaking (6, 4, 2)**			
3.1	De voortgang is minimaal 5 keer bijgehouden gedurende de uitvoering van het project. (bevat datum en daadwerkelijke bestede uren)			
3.2	Elke voortgangsmeting bevat status, afwijking en een beschreven actie op die afwijking.			
3.3	Er is een afsluitende evaluatie/reflectie opgenomen van het verloop van het project.			

B1-K1-W2

Ontwerpt software

Rubics

1. De eisen en wensen zijn vertaald naar een passend, eenduidig en volledig ontwerp.

Onderlegger eisen en wensen
Alle (minimaal 12) functionele eisen uit de planning zijn vertaald naar een ontwerp.

1. Functionaliteiten met betrekking op de GUI zijn voorzien van illustraties (schets, wireframe) en/of bevatten een eenduidige beschrijving en zijn zodoende **testbaar**. Complexe functionaliteiten moeten illustraties bevatten.
2. Elke functionele beschrijving bevat: **doel**, **invoer**, **uitvoer** en **foutafhandeling**.
3. Wanneer een functionaliteit uit stappen (flow) bestaat, dan wordt deze beschreven.
4. **Alternatieve** scenarios (de niet standaard flow, zoals cancelen van de winkelwagen of error flow) worden beschreven.

Technische eisen worden in het ontwerp concreet gemaakt. Zo wordt bijvoorbeeld d edatabase in een volledig en juist ERD uitgewerkt.

1. **Technische beschrijvingen** worden **onderbouwd** (waarom) en er worden **alternatieven** beschreven.

2. Er is gebruikgemaakt van relevante of toepasselijke schematechnieken (bijv. activiteitendiagram, klassendiagram, ERD, use case diagram).

Onderlegger schematechnieken

1. Er worden minimaal 2 **relevante schema technieken** op een juiste en volledige manier toegpast. Bijvoorbeeld een volledig en juist ERD en een (aantal) programma flows.

3. De gemaakte keuzes in het ontwerp zijn onderbouwd met steekhoudende argumenten, waarbij rekening is gehouden met haalbaarheid, privacy en security.

Onderlegger onderbouwing

1. Ontwerpkeuzes worden **onderbouwd** (waarom).
2. Bij tenminste **3 onderbouwingen** zijn één of meer **alternatieven** is/zijn overwogen.
3. Er wordt beschreven welke keuzes er ten aanzien van **security** of **privacy** (AVG) wat specifiek toepasselijk is op hun project. (de onderbouwing waarom security of privacy niet toepasselijk is voldoende).

Checklist

	Criterium	Nee (0)	Beetje (1)	Ja (2)
1	**Vertaling eisen naar ontwerp (10,6,3)**			
1.1	Alle (minimaal 12) functionele eisen uit de planning zijn vertaald naar het ontwerp.			
1.2	GUI functionaliteiten hebben schetsen/wireframes of een eenduidige beschrijving en zijn testbaar.			
1.3	Elke functionele beschrijving bevat: doel, invoer, uitvoer en foutafhandeling.			
1.4	Indien van toepassing zijn stappen (flow) en alternatieve scenario's beschreven.			
1.5	Technische eisen zijn concreet gemaakt (bijv. volledig ERD), zijn onderbouwd.			
2	**Schematechnieken (4, 2, 1)**			
2.1	Er zijn minimaal 2 relevante, verschillende schematechnieken; ERD, Flow diagram			
2.2	De schematechnieken zijn helemaal juist toegepast.			
3	**Onderbouwing en keuzes (8,5,2)**			
3.1	Ontwerpkeuzes zijn onderbouwd (waarom is hiervoor gekozen?).			
3.2	Bij tenminste 3 ontwerpkeuzes zijn minstens één of meer alternatieve oplossingen overwogen en beschreven.			

3.3	Er zijn bewuste keuzes beschreven ten aanzien van security en privacy (AVG).			
3.4	Je verantwoord de keuzes die je hebt gemaakt ten aanzien van de haalbaarheid.			

B1-K1-W3

Realiseert (onderdelen van) software

Rubics

1. Er is voldoende functionaliteit gerealiseerd binnen de gestelde/geplande tijd.

Onderlegger voldoende functionaliteiten

Wat doet de code?

Bij dit punt gaat het om de hoeveelheid code en of de code werkt. Bij dit punt is minder relevant of de functionaliteiten goed aansluiten bij het ontwerp. De code moet wel een logisch onderdeel van het project zijn.

Richtlijnen hoeveelheid

- Aantal database objecten/entiteiten: 3
- Minimaal 4 to 6 frontend pagina's
- Database met 3 tot 5 tabellen
- Aantal regels code (HTML, CSS, JS en backend): 1000 - 2500 (met hulp van AI)
- bevat Crud functionaliteiten (e.g. Login)

Eisen/checklist

1. Er is minimaal 40 uur geprogrammeerd dit wordt onderbouwd door de hoeveelheid code, complexiteit en door middel van versiebeheer.
2. De code werkt en dit is aangetoond mbv video en applicatie die is geïnstalleerd en draait.

2. De opgeleverde functionaliteiten voldoen aan de eisen en wensen.

Onderlegger functionaliteiten voldoen aan ontwerp

Bij dit punt gaat het met name over de aansluiting aan de planning en het ontwerp.

1. De opgeleverde code is gebasseerd op de planning en ontwerp.
2. De opgeleverde code moet grotendeels voldoen aan de funcitonaliteiten omschreven in het ontwerp.

3. De kwaliteit van de code is goed.

Onderlegger kwaliteit code

1. code is consistent qua opbouw structuur en documa`entatie. (e.g. zelfde style comments, naamgeving, taal keuze)
2. Variabelen, functies, methods, bestanden, databaselementen hebben duidelijk en betekenisvolle namen
3. Code is voor 80% opgebouwd uit functies en functies doen één ding en zijn niet langer dan 60 regels.
4. Bestanden bevatten maximaal 400 regels code.
5. De hele code base heeft een duidelijke structuur.
6. Er is geen dubelle code (DRY).
7. Geen onnodig commentaar, maar alleen om code ter verduidelijken.
8. In de code worden fouten duidelijk afgehandeld (try/catch).
9. Wachtwoorden en dergelijke worden niet in plain taxt opgeslagen.
10. Er is bescherming tegen SQL-injection.

4. Versiebeheer is effectief toegepast.

Onderlegger versiebeheer

1. De code staat volledig in GIT versiebeheer.
2. Via versiebeheer zijn tenminste 5 versies beschikbaar en deze versies zijn min of meer gelijk verdeeld over de projecttijd zodat de opbouw en ontwikkeling te volgen is.

3. Commits zijn logisch opgebouwd, met duidelijke beschrijvingen.

Checklist

	Criterium	Nee (0)	Beetje (1)	Ja (2)
1	**Gerealiseerde functionaliteit & Kwantiteit (4, 2, 1)**			
1.1	Er is minimaal 40 uur geprogrammeerd (onderbouwd door hoeveelheid code, complexiteit en versiebeheer).			
1.3	De code werkt en dit is aangetoond (video en/of draaiende applicatie).			
2	**Aansluiting op eisen en wensen (4, 2, 1)**			
2.1	De opgeleverde code is in zijn algemeenheid zichtbaar gebaseerd op de gemaakte planning en het ontwerp.			
2.2	Alle eisen en wensen zijn verwerkt volgens de (evetnueel aangepaste) planning			
3	**Kwaliteit van de code (12, 8, 3)**			
3.1	Naamgeving: Code is consistent, variabelen/functies hebben betekenisvolle namen.			
3.2	De code heeft een gangbare, consistente en duidelijke mappen structuur.			

3.3	Geen onnodig commentaar, maar alleen om code ter verduidelijken en er is geen dubbele code (DRY).			
3.4	De code is modulair opgezet. Je gebruikt functies met één duidelijke taak. Bestanden bevatten maximaal 400 regels.			
3.5	Robuustheid: Fouten worden opgevangen, afgehandeld (try/catch), database constraints zijn aanwezig, etc.			
3.6	Security: Geen plain text wachtwoorden en bescherming tegen SQL-injection, en bijvoorbeeld CSRF, ...			
4	**Versiebeheer (8, 5, 3)**			
4.1	De code staat volledig in GIT versiebeheer.			
4.2	Er zijn tenminste 5 versies beschikbaar, verdeeld over de projecttijd (opbouw is te volgen).			
4.3	Commits zijn logisch opgebouwd en hebben duidelijke beschrijvingen.			
4.4	Temminste 5 commits zijn duidelijk te koppelen aan functionaliteiten, zoals in de planning staat beschreven.			

B1-K1-W4

Test software

1. De testcases in het testplan sluiten aan op de functionaliteiten en bevatten alle relevante scenario’s.

Definitie

Testcases: Een testcase is een gedetailleerde beschrijving van de input, acties en de verwachte output waarmee wordt vastgesteld of een specifiek onderdeel van de software correct functioneert volgens de gestelde eisen.

Onderlegger Kwaliteit Testcases

1. De testcases beschrijven duidelijk de **uitgangssituatie** (pre-condities), de **actie** (teststappen) en het **verwachte resultaat**.
2. Er is onderscheid gemaakt tussen de **Happy Flow** (alles gaat goed) en **Alternative/Unhappy Flows** (foutieve invoer of uitzonderingen).
3. De testcases zijn **herhaalbaar**: iemand anders moet de test op basis van de beschrijving exact zo kunnen uitvoeren.
4. De gekozen testdata is relevant en dekt de grenswaarden (boundary testing) indien van toepassing.

2. De kandidaat heeft voor alle toegewezen of geplande functionaliteiten testscenario’s of testcases opgesteld.

Onderlegger Volledigheid (Dekking)

1. Er is een duidelijk overzicht (bijv. een matrix) waarin elke **requirement** of functionaliteit is gekoppeld aan minimaal één testcase.
2. Er zijn geen functionaliteiten 'vergeten'; de scope van de test komt overeen met de scope van de realisatie.
3. De prioriteit van de testcases is bepaald wat moet absoluut werken, wat is nice-to-have. (Dit is anders dan de moscow tabel in de planning)

3. De kandidaat voert de testactiviteiten correct en volgens het testplan uit.

Onderlegger Uitvoering

1. De tests zijn daadwerkelijk uitgevoerd op de software (aantoonbaar via logs, screenshots of database-states).

2. Bij de uitvoering is geregistreerd wat het **werkelijke resultaat** was (versus het verwachte resultaat).

4. Het testrapport bevat testresultaten van alle functionaliteiten, voorzien van de juiste conclusies.

Onderlegger Rapportage & Conclusie

1. Het testrapport geeft per testcase duidelijk aan: **Geslaagd (Pass)** of **Gefaald (Fail)** .
2. Voor gevonden fouten (bevindingen/bugs) is een duidelijke beschrijving gemaakt.
3. Er wordt een **eindconclusie** getrokken over de kwaliteit van de software (bijv. "vrijgave voor productie" of "herstel nodig") en deze wordt onderbouwd.
4. De rapportage is begrijpelijk voor de opdrachtgever en het ontwikkelteam.

Checklist

Vul in de kolom 'Locatie & Bewijslast' exact in waar het bewijs te vinden is (Bestandsnaam + Pagina/Screenshot nr).

Nr	Criterium (Wat moet je aantonen?)	nee (0)	Beetje(1)	ja (2)
1	**Kwaliteit Testcases (8, 5, 3)**			
1.1	De testcases bevatten een **pre-conditie**, **actie** en **verwacht resultaat**.			
1.2	Er zijn zowel **Happy Flow** als **Unhappy Flow** (foutscenario's) tests uitgewerkt.			
1.3	De gekozen **testdata** is concreet en dekt randgevallen.			
1.4	Alle tests zijn reproduceerbaar met dezelfde eenduidige uitkomsten.			
2	**Volledigheid / Dekking (4, 2, 1)**			

Nr	Criterium (Wat moet je aantonen?)	nee (0)	Beetje(1)	ja (2)
2.1	Je maakt een **matrix** waarbij je een relatie legt tussen test en functionaliteit. Hiermee toon je aan hoe goed elke functionaliteit is getest..			
2.2	**Alle functionaliteiten** zijn getest.			
3	**Uitvoering (4, 2, 1)**			
3.1	Het **werkelijke resultaat** is bij elke uitgevoerde test vastgelegd.			
3.2	Je toont bewijs van de **daadwerkelijke uitvoering** (screenshots, logs, database-status) van de tests.			
4	**Rapportage & Conclusie (4, 2, 1)**			
4.1	Testresultaten en gevonden fouten (bugs) zijn duidelijk geregistreerd (omschrijving + ernst) en bevatten de juiste conclusie.			
4.2	Het rapport bevat een duidelijke **eindconclusie** /advies over de softwareversie.			

B1-K1-W5

Doet verbetervoorstellen voor de software

Rubrics

1. Analyseert systematisch alle beschikbare informatiebronnen op mogelijke aanpassingen aan de software.
2. Interpreteert en vertaalt wensen, reacties, testresultaten en meldingen naar realiseerbare verbetervoorstellen.
3. Stelt vast welke werkzaamheden nodig zijn en maakt een haalbare planning.

Onderlegger

1. Analyseert systematisch alle beschikbare informatiebronnen op mogelijke aanpassingen aan de software.

Onderlegger Analyse

1. De kandidaat verzamelt input uit **diverse bronnen** (bijv. testrapporten, gebruikersfeedback, error-logs, code reviews of backlog-items).
2. De kandidaat filtert de informatie op relevantie en urgentie (niet elke melding is een verbetering).
3. Er is sprake van een **systematische aanpak**: de analyse is navolgbaar en niet gebaseerd op willekeur.

2. Interpreteert en vertaalt wensen, reacties, testresultaten en meldingen naar realiseerbare verbetervoorstellen.

Onderlegger Verbetervoorstellen

1. Het verbetervoorstel is **concreet** beschreven: het is duidelijk wat er technisch of functioneel moet veranderen.
2. De kandidaat toont aan waarom dit een verbetering is (bijv. performance winst, betere UX, schonere code).
3. Het voorstel is **technisch haalbaar** binnen de context van het project en de architectuur.
4. De kandidaat houdt rekening met de impact van de wijziging op andere systeemonderdelen (impactanalyse).

3. Stelt vast welke werkzaamheden nodig zijn en maakt een haalbare planning.

Onderlegger Planning & Taken

1. Het verbetervoorstel is uitgewerkt in een lijst met **benodigde werkzaamheden** (taken/subtaken).
2. Er is een inschatting gemaakt van de benodigde tijd (uren/storypoints) per taak.
3. De taken zijn ingepland in de tijd (bijv. in een sprint of roadmap) en de deadline is realistisch.

Checklist

		Nee (0)	Beetje (1)	Ja (2)
1	**Analyse (6,4,2)**			
1.1	Je hebt minimaal 2 verschillende **bronnen** gebruikt (bijv. logs, feedback, reflectie en resultaten van testcases) voor je analyse.			
1.2	Je analyseert en **prioritiseert** de mogelijke aanpassingen.			
1.3	Je hebt analyseert hoeveel tijd elke aanpassing kost. De totale tijd van alle aanpassingen is **minimaal 8 uur** aan werk.			
2	**Verbetervoorstel (4,2,1)**			
2.1	Elke verbeter voorstel is **concreet** uitgewerkt (het is duidelijk *wat* er moet gebeuren).			
2.3	Elk verbeter voorstel is realistisch,			
3	**Planning (4,2,1)**			

3.1	Je hebt het voorstel opgesplitst in een (technisch) stappenplan			
3.2	Er is een realistische **inschatting** gemaakt van de tijd per taak.			

Revision #37
Created 2025-12-06 11:22:31 UTC by Max
Updated 2025-12-19 14:35:25 UTC by Max