|  |  |
| --- | --- |
| **Datum** | 18-11-2024 |
| **Klas** | PALBVSOD3B |
| **Groepsamenstelling** | Kevin Troost, Ramino Vrca, Aaron hernandez nole, Evan Cheung |

## Proef examen opdracht

## Algemeen

De opdracht duurt in totaal 10 weken, **de deadline** om het project in te leveren is 31 januari en de presentatie week is direct in de week na de deadline.

Het project moet ingeleverd worden via GitHub en de laatste commit moet **uiterlijk 31 januari zijn**.

In dit document ga je samen met je groepsgenoten vaststellen wat jullie opdracht gaat zijn om alle competenties aan te tonen die bij deze opleiding horen.

Je moet laten zien dat je kan:

* Documenteren
* Ontwerpen
* Programmeren
* Testen
* Reflecteren
* Samenwerken

Iedereen moet bewijsstukken laten zien van bovenstaande onderwerpen, het is niet toegestaan dat een persoon alleen het ontwerpen doet en de ander alleen het programmeren.   
  
Bij de presentatie zal naar ieders aandeel worden gevraagd van alle onderwerpen. Hoewel het een gezamenlijke presentatie is zal de beoordeling op individueel niveau zijn.

## Opdracht

Dit is een vrije opdracht, je bent zelf vrij om de taal, framework en onderwerp te kiezen. Om er voor te zorgen dat iedereen wel qua moeilijkheidsgraad een gelijkwaardig project heeft vul je dit document zorgvuldig in en wordt deze vervolgens goed/afgekeurd door je docent. De deadline voor dit document is **donderdag 21 november en je levert het in op Canvas.**

Vervolgens krijg je uiterlijk **maandag 25 november** een **go/no-go** van je docent.

## Beschrijving project

|  |  |
| --- | --- |
| **Beschrijving van het project (Wat ga je maken)** | **Akkoord** |
| * Algemene beschrijving wat je gaat maken * Project management systeem * Hier noteer je nog niet de gekozen technieken |  |
| **Beschrijving van de technieken (Hoe ga je het maken)** | **Akkoord** |
| * Benoem hier specifiek welke technieken je gaat gebruiken * Front-end: Boostrap, Laravel, HTML, CSS, Javascript * Back-end: Laravel, PHP, SQL * Back-end framework bijv. Welke front-end technieken |  |
| **Beschrijving van de software die je gaat gebruiken** | **Akkoord** |
| * Denk aan een IDE, Dev omgeving, versie van PHP, maar ook andere relevante software om je ontwerpen te maken * IDE: visual studio code * Frameworks: Laravel, Bootstrap * Php versie: 8.2.12 * Laravel versie: 11.32.0 * Versiebeheer: git, Github * Documentatie: Word, Figma, Trello, MSQL Workbench, Github * Programeertalen: HTML, CSS, Javascript, PHP |  |
| **Waar sla je de documentatie op?** | **Akkoord** |
| * Verantwoord hier hoe jullie effectief samen gaan werken en veilig je documenten bewaren * Documentatie word opgeslagen in Figma, Trello en in een github mapje Documentatie. |  |
| **Beschrijf waar je de planning op gaat slaan** | **Akkoord** |
| * Hoe geef je prioriteiten aan * Must-haves, Should-haves, Could-haves * Hoe werk je samen * We gebruiken, Teams en discord voor communicatie. * Hoe is de structuur * Structuur van Trello: Backlog->To-Do->Doing->Testing->Done Week 1->Done week 2->Done week 3 enzovoort tot Done Week 10 |  |
| **Hoe bewijs je de voortgang tijdens het project** | **Akkoord** |
| * Laat hier zien hoe jullie aantonen dat er elke week netjes volgens de planning wordt gewerkt. * Gebruikt maken van Trello en bij kaarten een checklist hebben en een eind datum. Door voortgang word ook bewaakt via git en github, zodat we kunnen in zien wie wat heeft gepusht en in welke week. |  |
| **Beschrijf welke schematechnieken en ontwerptechnieken je gaat gebruiken (use-case diagram, klasse diagram, activiteiten diagram, flowchart, erd, wireframe, sitemap) en benoem wie verantwoordelijk is voor wat.** | **Akkoord** |
| * Welke schematechnieken * Use Diagram, Klasse Diagram, activiteiten diagram, ERD * Welke ontwerptechnieken * Wireframe, Sitemap * Wie maakt wat |  |
| **Welke code conventions ga je hanteren** | **Akkoord** |
| * Welke afspraken maak je met je team om een uniforme code base te garanderen * Functions worden geschreven in pascalcase. * Variabelen zijn in Snakecase en mogen een laag streepje bevatten (\_) Als een variabele meerdere woorden bevat en het eerste woord eindigt met dezelfde letter als de beginletter van het volgende woord, dan wordt de eerste letter van het tweede woord een hoofdletter. * Voorbeeld: * Fout: usernameexample * Correct: usernameExample * Classen zijn met pascalcase geschreven. |  |
| **Beschrijf hier hoe je gaat bewijzen dat je versiebeheer correct hebt toegepast** | **Akkoord** |
| * Wat voor versiebeheer gebruik je * Versiebeheer word toegepast door middel van Git en Github en branches. * Hoe ga je het gebruiken * Het team pusht naar GitHub zodra een functionaliteit is afgerond. |  |
| **Welke technieken ga je toepassen om te bewijzen dat je op niveau zit? (OOP, Functioneel programmeren)** | **Akkoord** |
| * Beschrijf hier welke techniek je gaat toepassen * OOP word toegepast. |  |
| **Hoe ga je de userstories testen?** | **Akkoord** |
| * Welke technieken ga je gebruiken? * Acceptatietest en Unit testing * Hoe is de aanpak?   Stap 1: eerst moet je de User Storie en Acceptatiecriteria gebruiken  Voordat je begint met het testen, je moet eerst de user storie en de acceptatiecriteria goed begrijpen. Wat moet er precies worden bereikt? Wat zijn de randvoorwaarden voor de functionaliteit?  Wij bestudeer de user storie en de acceptatiecriteria.  Wij Definieer duidelijk wat de verwachte uitkomsten zijn en welke functionaliteit getest moet worden.  Stap 2: Wij kiezen de juiste testtechnieken  Afhankelijk van de user storie en de functionaliteit, kiezen wij de testtechnieken:  Acceptatietests voor het testen van de volledige gebruikersflow en de vereisten.  Unittests voor het testen van de kleinere onderdelen (functies, logica) die de user storie ondersteunen.  Stap 3: Ontwikkel de Tests  Wij Schrijven de tests op basis van de gekozen technieken:  Voor unit tests: Identificeren wij de functies en methoden die je moet testen en schrijven tests om de functionaliteit van deze eenheden te verifiëren.  Voor acceptatietests: Ontwikkel een end-to-end test die de volledige gebruikerservaring van de user storie simuleert.  Stap 4: Voer de Tests uit  Wij Voeren de tests uit in een geschikte testomgeving. Zorg ervoor dat de tests zowel op de unit als de gebruikerstoepassing correct worden uitgevoerd.  Unit tests worden lokaal uitgevoerd.  Acceptatietests: Elke acceptatietest start altijd vanuit 1 punt en voorbepaalde tests word er een test account geven.  Stap 5: Analyseer de Testresultaten  Wij bekijken de resultaten van de tests:  Als de tests slagen, betekent dit dat de functionaliteit naar behoren werkt.  Als de tests mislukken, analyseer dan welke delen van de code of gebruikersstroom de fout veroorzaakten en los de problemen op.  Stap 6: Documenteer de Testresultaten  Documenteer de resultaten van je tests:  Welke tests zijn geslaagd?  Welke tests zijn gefaald en waarom?  Welke oplossingen zijn geïmplementeerd?  Stap 7: Herhaal indien nodig  Wanneer er nieuwe functionaliteit wordt toegevoegd of er codewijzigingen plaatsvinden, herhaal de tests om ervoor te zorgen dat de wijzigingen geen bestaande functionaliteit breken (regressietests). |  |
| **Hoe maken jullie het testrapport** | **Akkoord** |
| * Hoe rapporteren jullie de bevindingen uit te tests   Via een testrapport en een document. |  |
| **Hoe gaan jullie om met het overleg?** | **Akkoord** |
| * Hoe vaak? * We hebben elke week een vergadering. * Welke afspraken maken jullie? * We bspreken en maken afspraken wie wat deze week gaat doen. * Waar bewaren jullie de notulen? * Nog geen idee |  |
| **Hoe bewijzen jullie dat je hebt gereflecteerd op je werk?** | **Akkoord** |
| * Waar noteer je het? * Hoe bewijs je het? |  |

|  |  |
| --- | --- |
| **Naam** | **Handtekening** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |