Dynamic Webapps
Eindopdracht

1 Evaluatie

1.1 Dynamic_oefeningen Project

We werken gedurende de lessen een **project** uit waaraan de **leerstof** wordt opgehangen. Omdat het essentieel is dat elke stap gekend is, wordt wekelijks het gerealiseerde project ingeleverd.

1.2 Toetsmomenten

Er zijn twee toetsmomenten voorzien.
De eerste toets telt mee voor **20%** van het eindcijfer en de tweede toets telt mee voor **30%** van het eindcijfer.
Hierin vragen we om een aanpassing te doen in code die de student op voorhand krijgt.

1.3 Eindopdracht

De student krijgt ook een **eindopdracht** voor **50%** van het eindcijfer waarin alle aspecten van de leerstof aan bod komen. De details van die opdracht worden in het begin van het kwartaal bekendgemaakt.

Studenten kunnen steeds feedback vragen en krijgen over de ingeleverde tussentijdse opdrachten waardoor ze de eindtaak tot een goed einde kunnen volbrengen.

Op het einde van het kwartaal is er een **mondeling examen** over de wekelijkse opdrachten én de eindopdracht om na te gaan of dit eigen werk is en of de student voldoende inzicht heeft verworven in de materie. Het niet kunnen antwoorden op relevante technische vragen leidt tot een onvoldoende op de opdracht. De student kan tijdens deze verdediging gevraagd worden om evidente technische zaken aan te passen, teneinde na te gaan of hij voldoende inzicht heeft verworven.

2 EindOpdracht Minimum Technische Specificaties

Je mag zelf een thema bedenken voor je applicatie.
Elke student maakt **individueel** een project.
Tijdens de mondelinge evaluatie moet je dit kunnen tonen in je code en ook in de code die we gebruikt hebben tijdens de les.

Minimum technische specificaties:

- werkt onmiddellijk (zonder dat de docent eerst aanpassingen moet doen)
- je project is gedeployed
- je project is correct gestructureerd in files en directories
- minstens 15 verschillende **eigen** Components (al dan niet in aparte files)
- minstens 1 keer een list/array/collection van Components correct renderen
- minstens 1 keer props.children gebruiken
- minstens 1 keer een Component **anders** (of niet) laten renderen afhankelijk van een conditie
- minstens 1 keer een Fragment gebruiken
- minstens 5 react-bootstrap Components gebruiken
- minstens 5 eventHandlers
- minstens 5 keer "use_state"
- minstens 1 state die door meerdere components gebruikt wordt
- minstens 2 **eigen** contexts
- localStorage
- minstens 5 input elementen waar de gebruiker iets moet ingeven **en** doe iets nuttigs hiermee
- maak een database in FireBase
- lees data uit de database
- schrijf data in de database
- gebruik minstens 1 async functie
- minstens 1 useEffect

Let ook op het volgende:

- maak geen copy van dynamics_oefeningen. Bouw je eigen applicatie op.
- Look&Feel: ziet er professioneel uit en voldoende verschillend van dynamics_oefeningen
- Thema en opbouw is voldoende verschillend van dynamics_oefeningen
- Look&Feel: geen broken links, alles goed uitgewerkt
- Als je edit-pagina maakt voor bestaande data dan moeten input velden in een form altijd pre-filled zijn, zodat het bruikbaar is voor de gebruiker
- probeer voor edit/new-pagina's de input velden zo handig mogelijk te maken voor de gebruiker (geen text veld als een select mogelijk is)
- de code is proper geformatteerd
- er is geen overbodige code of overbodige variabelen, functies, …
- gebruik zoveel mogelijk javascript manier van denken: werk met functies
- gebruik geen gewone for-loop om een array te verwerken – behalve als je echt een duidelijke reden hebt waarom het niet anders kan

- maak zeker geen classes – als je code van het internet haalt en die werkt met classes zet dat dan om naar functionele components
- applicatie moet nog werken als connectie met de database faalt of als er geen data aanwezig is voor jouw project
-

Gebruik git en Github:

- je project is gepushed in github
- docent is toegevoegd als collaborator
- je hebt op branches gewerkt (branches niet deleten na mergen naar master)
- je hebt op regelmatige basis gecommit

Extra punten indien:

- nog iets extra toegevoegd met behulp van documentatie in Canvas
- iets extra toegevoegd dat je zelf opgezocht hebt
- je hebt extra packages geinstalleerd met npm die je zelf opgezocht hebt
- opdracht is verrassend en creatief uitgewerkt

Noot:

- Je mag mijn componenten hergebruiken maar die tellen niet mee als **eigen** Components
- Je mag mijn Contexts hergebruiken maar die tellen niet mee als **eigen** Contexts
- Je mag code-snippets gebruiken die je op het internet gevonden hebt maar je moet deze kunnen verklaren en aanpassen. Als het grotere stukken zijn vermeld dan zeker ook waar je ze gehaald hebt.

3 Project dynamic_oefeningen indienen

Op het einde van elke bundel dien je de aanpassingen in die je gedaan hebt aan het project dat we samen in de klas opbouwen (dynamic_oefeningen).
Er zal steeds een opdracht zijn in Canvas.
De juiste datums worden tijdig gecommuniceerd op Canvas.

4 EindOpdracht resultaten indienen

Tijdens de cursus dient de student 2 keer een tussentijds resultaat in.
In het begin van de laatste week dien je de definitieve versie in.
De juiste datums worden gecommuniceerd op Canvas.

5 Een opdracht indienen in Canvas

- Dien iets in dat werkt! Zelfs al is niet alles af, hetgeen je gemaakt hebt moet correct werken.
- Alles moet gecommit en gepushed zijn en gemerged naar master. Alleen code op master telt mee. Alleen commits die op tijd gebeurd zijn tellen mee.
- Zip je project, zonder de folders **target, .git, build, node_modules, .idea, .vscode ....**
- Upload in **Canvas** in de juiste opdracht:
    - je **gezipte** project. Upload nooit een zip waar **target**, **.git (etc)** folders inzitten!
    - url van je github repository
    - url van de gedeployde applicatie