**EINDONTWERP**

|  |  |
| --- | --- |
| **Naam** | Jay Bhageloe |
| **Studentnummer** | 22072039 |
| **Klas** | SE.H5 |

**OPDRACHT**

**Geef hieronder een korte beschrijving van je opdracht (een samenvatting van je productvisie) zodat je docent OPT3 ook begrijpt met welke opdracht jij bezig bent (neem dit onderdeel over uit je laatst ingeleverde opdracht en vul aan of wijzig de casus aan de hand van eigen bevindingen of op basis van feedback die je hebt gekregen van docenten en/of medestudenten):**

Mijn opdracht helpt de productgever met het plannen dat een andere manier gebruikt dan wat de school aanbeveelt aan de studenten. Een applicatie waar je invoert hoeveel lessen, projecten, deadlines of toetsen je hebt. Integendeel tot een normale kalender kan je hier toevoegen hoeveel keer je nodig hebt voor een toets voordat je het lesmateriaal onder de knie hebt. Voor lessen krijg je herinneringen dat je je werkcollege of lesmateriaal voorbereiding moet doen ook al is het maar voor 15 minuten, het is handiger dat je een herinnering krijgt en een paar minuten besteed dat helemaal niets doet.

Voor projecten en deadlines krijg je ook een countdown hoeveel dagen je nog overhebt want een visuele representatie van wanneer iets af moet is veel motiverend voor de klant dan een normale kalender bericht. Je kan ook tijden aangeven wanneer je meestal dingen voor school aan het doen bent bijvoorbeeld laat in de avond of juist tussen 5 en 8 voor werk. De app kan in die tijd aangeven wat er gedaan kan worden in die tijd en of er iets urgents afgemaakt moet worden

<https://github.com/SchoolAccount22072/ProjectPatterns1>

**USER STORIES**

|  |
| --- |
| **Als** eigenaar  **wil ik** een interface waar ik gemakkelijk mijn informatie kan invoeren en kan opslaan en terugzien **zodat ik** de deadlines, taken en toetsen kan inzien en later kan zien of de vakken, projecten en deadlines zijn voldaan of niet**.** |
| Alseigenaar wil ik interface waar ik op een gesorteerde manier de informatie kan inzien zodat ik precies weet hoeveel tijd er is voor elk vak, deadline, toets, etc. |
| Alseigenaar wil ik specifieke informatie (datum, tijd, vak) kan invoeren voor elke deadline zodat ik een duidelijk overzicht heb van mijn verplichtingen. |
| Alseigenaar wil ik dat er alleen relevante informatie in de applicatie komt en niet informatie zoals individuele lessen, vrije dagen, bijlessen, etc. zodat ik alleen de belangrijke informatie kan inzien en niet overbelast raak met onnodige informatie. |
| Alseigenaar wil ik een overzichtelijke visualisatie van de progressie op basis van gehaald of niet gehaald toetsen en of projecten zodat ik kan inzien of de applicatie effectief is en of alle taken zoals deadlines zijn voldaan.. |

**EINDONTWERP**

In het eindontwerp heb je Patterns en SOLID-principes toegepast en je hebt Classes toegevoegd die je nodig hebt om je applicatie te laten werken (zoals voor het tonen van een GUI of Command Line voor bijvoorbeeld een menu). In het ontwerp heb je ook de types van je properties, parameters en methodes toegevoegd. Zie het volgende document voor voorbeelduitwerking en aanwijzingen:

‘[OPT3 16.2 - Ontwerp en realisatie met patterns - Toelichting.pdf](https://brightspace.hhs.nl/content/enforced/18624-H-SE-S2OPT3-20_2022_VT/documenten/OPT3%2016.2%20-%20Ontwerp%20en%20realisatie%20met%20patterns%20-%20Toelichting.pdf)’

ZOOM IN VOOR DETAILS IN DE UML DIAGRAM

A diagram of a company

Description automatically generated

\* Controleer of het plaatje goed leesbaar is.

**PATTERNS**

Beschrijf hieronder hoe je het Template Method Pattern en een tweede Pattern hebt toegepast in bovenstaand eindontwerp:

|  |  |
| --- | --- |
| **Pattern** | **Beschrijving** |
| Template Method Pattern | Na(a)m(en) van template methods: deadline, project, toets  Dus de template methode patroon wordt in de deadline klasse geimplementeerd door methodes de gebruiken zoals voerNaamIn en voerSpecifiekeInfoIn en deze abstract te maken waardoor de DeadlineConcrete een andere waarde kan geven aan de deadline. Hier wordt dus gebruik gemaakt van polymorfisme. |
| Factory Pattern | Als je kijkt naar de code zie je dat de factory pattern is toegepast bij de student en docent klasse met de hoofdklasse van gebruiker waar je in de student en docent informatie kan aanpassen zonder dat de hoofdklasse veranderd wordt. |

**SOLID-PRINCIPES**

Beschrijf hieronder voor alle SOLID-principes hoe je die hebt toegepast in Bovenstaand eindontwerp:

|  |  |
| --- | --- |
| **SOLID-principe** | **Beschrijving (gebruik een spatie als je geen toelichting geeft)** |
| Single Responsibility | Als je kijkt naar de daglijst en rapportage klasse zie je hoe de klasse alle deadlines van alleen alle data verzamelt en sorteert in de console. Menu klasse is er specifiek voor om je te verwijzen naar een specifieke pagina. |
| Open-Closed | Als je kijkt naar de abstracte klasse deadline zie je dat het een standaard deadline klasse waar de project klasse en de toets klasse van kunnen overerven en aanpassen zonder de deadline klasse zelf aan te passen. |
| Liskov | Als je kijkt naar de subklasse Student en Docent, en de klasse deadlineContrete, project en toets zie je dat de subklasse je kan gebruiken samen met de hoofdklasse zonder dat de hoofdklasse zelf veranderd waaronder de Deadline en Gebruiker klasse. |
| Interface Segragation | Als je kijkt naar de code zie je dat er gebruikt wordt gemaakt van implementaties en in plaats van dat de code 1 hele grote interface heeft dat elke klasse implementeert zijn en kleinere interfaces die specifieke rollen hebben voor wat zij moeten doen. |
| Dependency-Inversion | Als je kijkt naar de code zie je dat de code in verschillende delen interfaces en implementaties gebruikt. Hij Dependency inversion is het zo dat de belangrijke interfaces niet afhankelijk zijn op kleinere interfaces die specifieke acties verrichten terwijl de belangrijke interfaces grotere en meer globale taken verrichten. |