**TESTONTWERP & -REALISATIE**

|  |  |
| --- | --- |
| **Naam** | Xue Kun Zhou |
| **Studentnummer** | 20170173 |
| **Klas** | 6 |

**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.

MyLibrary is een webapp dat bijhoudt welke films, series, boeken en games je hebt gezien, gelezen en gespeeld. De gebruiker heeft daarmee een digitale “collectie” van media die hij heeft geconsumeerd. Per item wordt er ook een score bijgehouden zodat wanneer de gebruiker op een later moment kan terugzien wat hij ervan vond. Door het bijhouden van deze items kan de gebruiker ook een overzicht krijgen van hoeveel tijd hij hierin besteed.

De app houdt niet bij welke films etc. er al bestaan en houdt dus geen gemiddelde score bij van alle gebruikers en geeft ook geen aanbevelingen van welke items de gebruiker misschien wel leuk vindt.

Vrienden kunnen wel de lijsten van elkaar zien.

**GITHUB EN TDD**

Maak je GitHub-repo Private en geef je docent toegang tot je repo:

https://github.com/XuekunZhou/OPT3\_MyLibrary

Om aan te tonen dat je volgens TDD hebt gewerkt voor deze opdracht neem je hieronder een kopie van je netwerk over (zie de toelichting op het template voor instructie):

<Kopie van je GitHub netwerkdiagram>

**GESLAAGDE TESTS**

Toon met een relevante screenshot aan dat alle onderstaande tests inderdaad zijn geslaagd:

<Screenshot van de geslaagde tests>

**SAMENGESTELDE DECISION**

In mijn software neem ik een besluit op basis van de volgende voorwaarden (maximaal 3 te testen):

Voorwaarde A: Gebruikers lijst is openbaar

Voorwaarde B: Ingelogde gebruiker is vrienden met gebruiker2

Voorwaarde C: Gebruiker is ingelogd

Het besluit is gebaseerd op de volgende combinatie van deze voorwaarden (bijv. D = A && (B || C)):

D = (A || B) && C

**CONDITION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Condition Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie):

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = (A || B) && C** |
| True | True | True | True |
| False | False | False | False |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |

**DECISION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Decision Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie):

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = (A || B) && C** |
| True | True | True | True |
| False | False | False | False |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |

**CONDITION/DECISION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Condition/Decision Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie):

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = (A || B) && C** |
| True | True | True | True |
| False | False | False | False |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |

**MODIFIED CONDITION/DECISION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Modified Condition/Decision Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie). Geef met <l>/<f> aan met welke logische en fysieke test cases je werkt:

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = (A || B) && C** |
| False | False | True | False |
| False | True | True | True |
| False | True | False | False |
| True | False | True | True |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |

**MULTIPLE CONDITION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Multiple Condition Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie):

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = (A || B) && C** |
| True | True | True | True |
| True | True | False | False |
| True | False | True | True |
| True | False | False | False |
| False | True | True | True |
| False | True | False | False |
| False | False | True | False |
| False | False | False | False |
| <waarde> | <waarde> | <waarde> | <waarde> |

**MODIFIED CONDITION/DECISION COVERAGE (JUNIT)**

Ik heb de testcases voor Modified Condition/Decision Coverage omgezet naar de volgende JUnit-code om mijn software voor deze beslissing te testen.

<De JUnit-code (o.b.v. MC/DC>

Hiermee test ik de volgende methode (van het type boolean, waarin de waarde van D = <Decision> wordt bepaald):

<De code van de geteste methode>

**EQUIVALENTIEKLASSEN EN RANDWAARDEN**

Voor het nemen van een besluit in de software heb ik een domein dat bestaat uit de volgende equivalentieklassen (maximaal 3 te testen):

|  |  |  |
| --- | --- | --- |
|  | Logische weergave | Besluit |
| Equivalentieklasse 1 | <Equivalentieklasse 1> | <Besluit 1> |
| Equivalentieklasse 2 | <Equivalentieklasse 2> | <Besluit 2> |
| Equivalentieklasse 3 | <Equivalentieklasse 3> | <Besluit 3> |

Geef nu alle relevante fysieke testgevallen (incl. de meest intensieve test van randwaarden) en geef voor elk testgeval aan voor welke equivalentieklasse dit een test is (als je de laatste regels niet meer nodig hebt, kun je deze velden vullen met een spatie):

|  |  |  |
| --- | --- | --- |
| Equivalentieklasse | Fysiek testgeval | Besluit |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |

**EQUIVALENTIEKLASEN EN RANDWAARDEN (JUNIT)**

Ik heb de testcases voor Equivalentieklassen en randwaarden omgezet naar de volgende JUnit-code om mijn software voor deze beslissing te testen.

<De JUnit-code (o.b.v. equivalentieklassen en randwaarden)>

Hiermee test ik de volgende methode (waarin de besluiten op basis van de equivalentieklassen worden bepaald):

<De code van de geteste methode>

**PAIRWISE TESTING**

Voor Pairwise Testing maak ik gebruik van de volgende parameters (op basis waarvan een besluit wordt genomen; vereenvoudig tot een set van 4 parameters die ieder 3, 2, 2 en 2 mogelijkheden hebben (anders wordt handmatig testen met Pairwise Testing veel te ingewikkeld; vul een spatie in, als je een optie niet wilt gebruiken):

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Optie 1** | **Optie 2** | **Optie 3** |
| <Parameter 1> | <Optie 1> | <Optie 2> | <Optie 3> |
| <Parameter 2> | <Optie 1> | <Optie 2> | <Optie 3> |
| <Parameter 3> | <Optie 1> | <Optie 2> | <Optie 3> |
| <Parameter 4> | <Optie 1> | <Optie 2> | <Optie 3> |

Na de tussenstappen in het proces heb ik de volgende testtabel met fysieke testgevallen opgesteld (vul spaties in, als er teveel testgevallen in de tabel staan voor jouw casus):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **<Parameter** 1> | **<Parameter** 2> | **<Parameter** 3> | **<Parameter** 4> | **Decision** |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |

**PAIRWISE TESTING (JUNIT)**

Ik heb de testcases voor Pairwise Testing omgezet naar de volgende JUnit-code om mijn software voor deze beslissing te testen.

<De JUnit-code (o.b.v. Pairwise Testing)>

Hiermee test ik de volgende methode (waarin het besluit op basis van de meegegeven parameters wordt bepaald):

<De code van de geteste methode>