Assignment 1 – IN3240

Mari K. Myrvold(marikmy), Petter Johan Sandvand(pettejs)

Part 1 – Test automation in Cypress

Task 1

1.1 –
1.2 –
1.3 **Test case**

Title: User customer care page

Description: A user should be able to successfully send a message to Customer Care.

Precondition: None

Assumption: A supported browser is being used.

Test steps:

1. Navigate to https://parabank.parasoft.com/parabank/index.htm
2. From the front page, navigate to the "Customer care" page by clicking on the 'mail' icon.
3. In the 'Name' field, enter a name. In the 'Email' field enter an e-mail address. In the 'Phone' field enter a phone number. In the 'Message' field enter a message. --- Or just 'Fill out the schema'?
4. Click 'Send to customer care'.

Expected result: A page displaying the text 'Thank you *name*. A Customer Care Representative will be contacting you.'

1.4 –

1.5 Test case fra oppg. 1.3:

```
describe("User customer care page", () => {
    it("successfully send", () => {
        cy.visit("https://parabank.parasoft.com/parabank/index.htm")
        cy.get(".contact").click()
        cy.get("#name").type("Mari")
        cy.get("#email").type("email@hotmail.com")
        cy.get("#phone").type("0000000")
        cy.get("#message").type("hei")

        cy.get(".form2").find(".button").click()
        cy.get("#rightPanel").should("contain", "Thank you Mari")
    })
})

```

Task 2

2.1 Other benefits from ensuring that our automated tests are independent of each other:
- Easier to locate defects while testing.

2.2. Test cases:

Test case 1:

Title: Request a loan

Description: A registered user should be able to successfully request a loan https://parabank.parasoft.com/parabank/index.htm

Precondition: The user must already be registered and logged into their account.

Assumption: A supported browser is being used.

Test steps:

1. Navigate to https://parabank.parasoft.com/parabank/index.htm
2. From the front page, click 'Request Loan'.
3. Fill out the schema 'Apply for a Loan'.
4. Click 'APPLY NOW'

Expected result: A message with the title "Loan Request Processed"

Test case 2:

Title: Bill Pay

Description: A registered and logged in user should be able to pay bills via the 'Bill Pay' link.

Precondition: The user must already be registered and logged into their account.

Assumption: A supported browser is being used.

Test steps:

1. Navigate to https://parabank.parasoft.com/parabank/index.htm
2. From the front page, click 'Bill Pay'.
3. Enter payee information.
4. Click 'send payment'

Expected result: A message with the title "Bill Payment Complete"

Test case 3:

Title: Request a demo of Parasoft C/C++test

Description: A visitor on the web page should be able to request a demo of Parasoft

C/C++test from https://parabank.parasoft.com/parabank/index.htm

Precondition: None

Assumption: A supported browser is being used.

Test steps:

1. Navigate to https://parabank.parasoft.com/parabank/index.htm
2. From the front page, navigate to the product page by clicking on the ‘products’ link.
3. Click ‘See It in Action’.
4. Under ‘Parasoft C/C++test’ click ‘Request Demo’ to request a demo.
5. In the ‘Company Email*’ field, enter an email address.
6. Click ‘Request Demo’.

Expected result: A message with the title “Thank You!”

2.5

```
describe("Request a loan", () => {
    it("Successfully request a loan", () => {
        cy.visit("https://parabank.parasoft.com/parabank/index.htm")
        // Log in
        cy.get(".login").find("[name=username]").type("mari")
        cy.get(".login").find("[name=password]").type("passord")
        cy.get(".login").find(".button").click()

        cy.get("a:contains(Request Loan)").click()
        cy.get("#amount").type("100")
        cy.get("#downPayment").type("50")
        cy.get("#fromAccountId").select(0)
        cy.get(".form2").find(".button").click()
        cy.get(".title").should("contain", "Loan Request Processed")
        // Log out
        cy.get("a:contains(Log Out)").click()
    })
})
```

2.6

```
describe("Bill Pay", () => {
    it("Successfully pay bills", () => {
        cy.visit("https://parabank.parasoft.com/parabank/index.htm")
        // Log in
        cy.get(".login").find("[name=username]").type("mari")
        cy.get(".login").find("[name=password]").type("passord")
        cy.get(".login").find(".button").click()

        cy.get("a:contains(Bill Pay)").click()
```

```
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(0).type("Mari")
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(1).type("Address")
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(2).type("City")
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(3).type("State")
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(4).type("0000")
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(5).type("0000")
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(6).type("0000")
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(7).type("0000")
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(8).type("0000")
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(9).select(1)
        cy.get(".form2 > tbody > tr > td:nth-child(2)").eq(10).eq(2).click()

        cy.get(".title").should("contain", "Bill Payment Complete")
        // Log out
        cy.get("a:contains(Log Out)").click()
    })
})
```

2.7-9

```
describe("Log in/log out", () => {
    beforeEach(() => {
        describe("Customer login", () => {
            it("successfully log in", () => {
                cy.visit("https://parabank.parasoft.com/parabank/index.htm")
                cy.get(".login").find("[name=username]").type("mari")
                cy.get(".login").find("[name=password]").type("passord")
                cy.get(".login").find(".button").click()
            })
        })
    })

    describe("Request demo", () => {
        it("Successfully request a demo of Parasoft C/C++test", () => {
            cy.get("a:contains(Products)").click()
            cy.get(".btn red-btn").click()
            cy.get(".product-btn").click()
            cy.get(".hs-input invalid error").type("email@hotmail.com")
            cy.get(".input").find("[name=firstname]").type("Mari")
            cy.get(".input").find("[name=lastname]").type("Lastname")
            cy.get(".input").find("[name=company]").type("Company")
            cy.get(".input").find("[name=jobtitle]").type("Jobtitle")
            cy.get(".input").find("[name=country]").click()
            cy.get(".hs-button primary large").click()
            cy.get(".content").should("contain", "Thank You!")

            cy.visit("https://parabank.parasoft.com/parabank/index.htm")
```

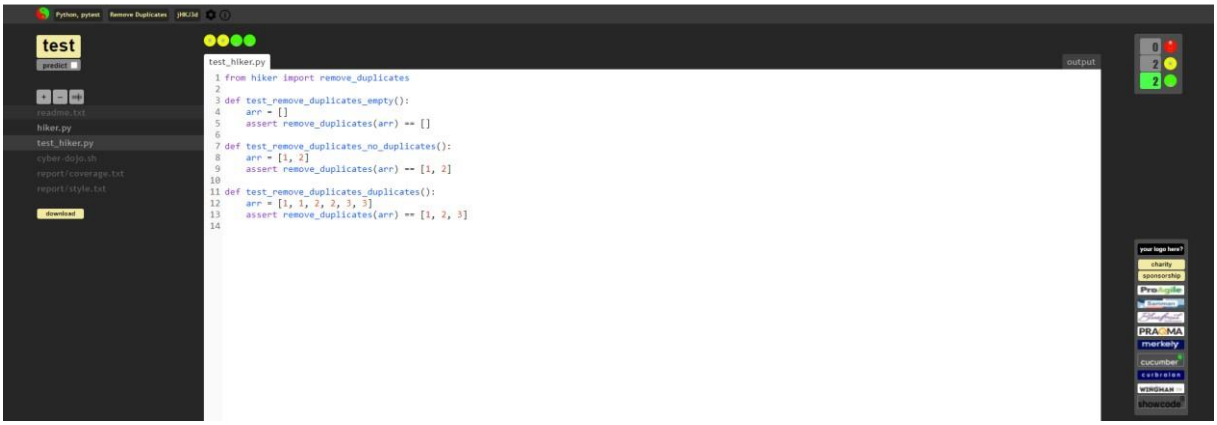
```
        })
    })

    afterEach(() => {
        describe("Customer log out", () => {
            it("successfully log out", () => {
                cy.visit("https://parabank.parasoft.com/parabank/index.htm")
                cy.get("a:contains(Log Out)").click()
            })
        })
    })
})
```

Part 2

Remove Duplicates

jHKJ3d

https://cyber-dojo.org/kata/edit/jHKJ3d



Calc Stats

https://cyber-dojo.org/kata/edit/PGVhd4

PGVhd4

Part 3

Task 0.

0.1 Specify which of the following options you have chosen. (e.g. Option 1: ”Hangman2017”)

I have chosen hangman 2017. Option 1

0.2 If you choose Option 3 ; specify the GitHub/Bitbucket URL.

Task 1.

1.1 What does the metrics say about the code/project in whole (e.g. quality gate)?

It states that the code passed the Quality gate, with is a Boolean that represents if the code is in general ok or not. To be more concrete, it tests if the code is production-ready or not. Coverage of the code is 0%, witch means that the code is not internally tested, with unit test or other dynamic tests. There is some duplications (5.2%) and but there is may bugs, 38 in total. This needs to be resolved.

1.2 Briefly explain what the types of issues listed by the analyzer mean?

1.2.1 Bugs?

There are coding errors that will in smaller or larger ways brake your code and needs to be fixed immediately.

1.2.2 Vulnerabilities?

Security vulnerabilities are code that hackers can exploit or use to their advantage.

1.2.3 Code Smells?

Code smells are in general badly written code, that can be typos or wrong naming conventions. Or other factors that make the code harder to read or maintain.

1.2.4 Blocker?

This is the severity of a bug, which means that the system of functionality is unavailable because of this bug.

1.2.5 Coverage?

Describes the number of lines in the code that is covered by a test. In percentage.

1.3 For the given project selected, name the folder(s) containing the highest number of LOC

(Lines of code), and for the given folder? The folder with highest LOC is *guihangman*

1.3.1 How many bugs are there? There

is 21 bugs here

1.3.2 How many vulnerabilities?

23 vulnerabilities

1.3.3 How many code smells?

There is 368 code smells in this folder

2.1 Select and list at least (four) 4 issues (e.g. bugs/vulnerability/code smells/. . .), and briefly explain why the analyzer has reported the given issues?

1. **https://sonarcloud.io/project/issues?resolved=false&types=BUG&id=sit%3Aint206%3AHangman2017&open=AVuJtiR7tpYg8Dj416A4**
    a. The issue here is that when I have opened up a connection there is best to close the connection after you have used it, so you don't end up with may open connections.
2. https://sonarcloud.io/project/issues?resolved=false&types=CODE_SMELL&id=sit%3Aint206%3AHangman2017&open=AVuJtiSAtpYg8Dj416B3
    a. The issue or code smell is that you should have a @override on methods that overrides others. This both from the compiler, so that you'll get a warning if it doesn't override, and to improve the readability.
3. https://sonarcloud.io/project/issues?resolved=false&severities=CRITICAL&types=CODE_SMELL&id=sit%3Aint206%3AHangman2017&open=AVuJtiR5tpYg8Dj416Ak
    a. This increases the duplication of code of the project. To write something several times. Here there is a string that is used in several places, that could be replaced with a constant.
4. https://sonarcloud.io/project/issues?resolved=false&severities=MAJOR&types=CODE_SMELL&id=sit%3Aint206%3AHangman2017&open=AVuJtiR7tpYg8Dj416A-
    a. The problem here is that old code is just commented out. This decreases the readability.

2.2 For each of the listed issues, how would you solve the problem?

1. The solution here would be to close the connection after it has been used. Or in a final statement. As a part of a try-catch statement
2. Here I would tag the method with @override
3. I would make a constant = "Enter Word here (Text)" and used that in all cases in the file that uses that.
4. I would remove the code that's commented out. And be sure to use a version control system to keep track of old versions of the code.

3.1 Briefly explain the following concepts:

3.1.1 False-positive (aka. False-fail result)

The test indicates that there is a bug in the code. But there is no bug in the code.

3.1.2 False-negative (aka. False-pass result)

The test indicates that there is no bug. But there is a bug in the code.

3.2 Provide an example of each of the concepts.

E.G the covid pandemic we have been using false positives and negatives a lot.

A false positive in that case means that the test is positive, but the person tested doesn't actually have covid.

False-negative, in that case, means that the test is negative, but the person tested actually has covid.

Part 4: Static testing – Reviews

Review is a static testing technique, which implies that it is based on manual examination and automated analysis of the code or documentation, without execution of the software under test. The purpose of reviews is to identify causes of failures/defects in the software. It can have a benefit in that is makes it possible to detect defects early in the life cycle, which makes the defect cheaper to remove. This contributes to an improvement in productivity, reduced developmental timescales and lifetime cost, fewer defects, as well as improved communication. Reviews can also find omissions in the requirements, or other part of the documentation, which are unlikely to be found in dynamic testing. It's generally easier to find defects with reviews than in dynamic testing.

Formal reviews are well structured and regulated, as opposed to informal reviews, which have no written instructions. In a formal review, e.g. an inspection, the fundamental steps are: planning, initiating review/kick off, individual review/individual preparation, issue communication and analysis in a review meeting, fixing and reporting with rework and follow up.

There are different kinds of reviews, including informal review, walkthrough, technical review, and inspection. Informal reviews can be used as an inexpensive way to get some benefit. Walkthrough is used for learning purposes, gaining understanding, finding defects and feedback. Technical reviews are used to discuss and make decisions, evaluate alternatives, find defects, solve technical problems, and check conformance to specifications and standards. Inspections are formal reviews with the purpose of finding defects.

Roles in reviews include: The author, who is responsible for the documents to be reviewed and the rework to be done. Management, who decides on the execution of reviews and assign resources like staff, budget, and time, as well as determining if the objectives of the review have been met. The review leader takes overall responsibilities, decide who will be involved, and works closely with the management and the facilitator. The facilitator, or moderator, who leads the review of the documents, plan the reviews, run meetings with follow-ups, and mediate between the various points of view if necessary. The reviewers are individuals with specific technical or business background who identify and describe the findings in the product under review, representing different perspectives and roles in the review process. The Scribe (or recorder) documents all the issues, problems and open points that are identified during meetings.

For a review to be performed successfully there are different organizational and people-related success factors; On organizational level it is necessary to have a clear objective, pick the right review type and technique, have the review material kept up to date, limit the scope of the review, have enough time, and management support. People-related success factors include picking the right reviewers, use testers, reviewers doing their review work well, limit the scope to only the most important, welcome any defects found, and that review meetings are well managed.

When using reviews in the early stages of the development process, test principle number 3 is followed: Early testing saves time and money.

Static analysis and reviews are both static testing techniques, which involves examining the code without executing it, with object to identify causes of defects. While reviews involve

manual examination of the code, static analysis uses tools to analyze program code. Both reviews and static analysis contribute to early detection of defects prior to test execution and can identify defects not easily found by dynamic testing.

Reviews may not be the most effective mean to quality assurance, depending on the size and complexity of the project since it involves many phases and personnel.

If we had to choose a review type to improve the quality of this compulsory assignment, we would maybe choose informal review with pair reviews, because it is an inexpensive and effective way to get some benefit which requires little written instructions for the reviewers. The reviews would contribute to little preparation for the reviewers, other than an instruction to read through the assignment.