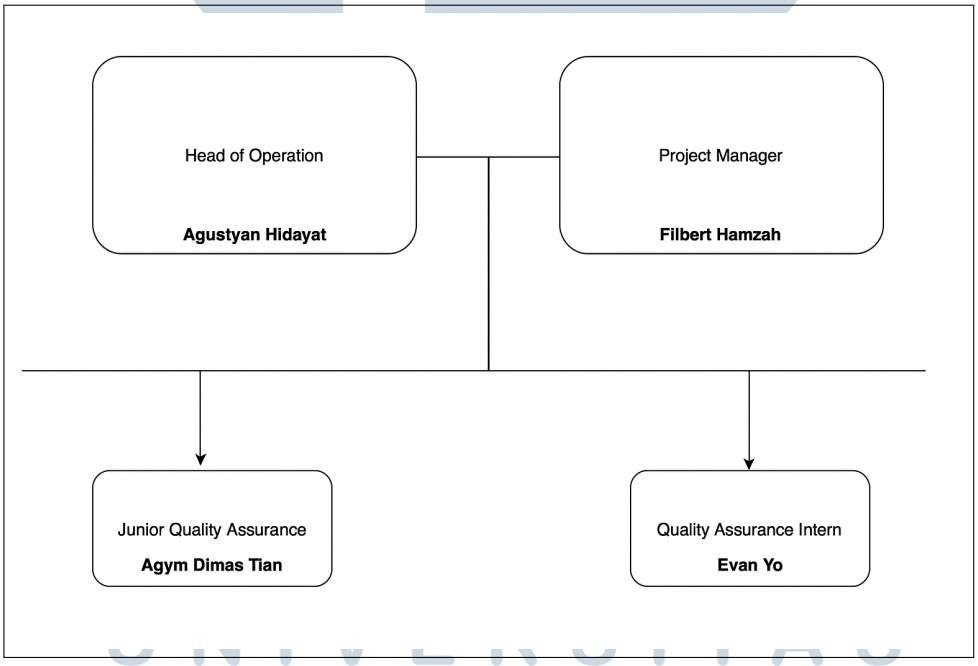
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama proses kerja magang di PT. Ganda Visi Jayatama, kedudukan yang ditempati adalah sebagai *Quality Assurance Intern* yang termasuk ke dalam divisi *QA Engineer*, yang berada di bawah *Head of Operations*. Dalam posisi ini terdapat 2 *Junior Quality Assurance*, dan 1 orang yang berperan sebagai *Quality Assurance Engineer Intern* yang dibimbing oleh Bapak Agustyan Hidayat.Dalam proses magang ini, terdapat beberapa tugas yang diberikan, yaitu mengecek seluruh fungsional *website* dan *apps* secara manual maupun otomatis, membuat *report* dalam proses *automation*, serta membuat *test case*.
Struktur organisasi perusahaan Concise bisa dilihat pada 3.1.



Gambar 3.1. Struktur organisasi perusahaan Concise

Dalam proyek ini, terdapat beberapa struktur tim proyek yang dapat dilihat pada Gambar 3.1. Proyek ini berisikan 1 orang sebagai *Project Manager*, yaitu Bapak Filbert Hamzah, 2 orang yang berperan sebagai *Frontend Engineer*, yaitu Bapak Adhitya Bagus Wicaksono dan Bapak Enrico Nathaniel, serta 1 orang *Backend Engineer*, yaitu Bapak Alwin Alamsyah. Selama proses ini, tim sering melakukan

sprint planning. *Sprint planning* sendiri dilakukan setiap 2–3 minggu sekali, yang bertujuan untuk mengetahui sejauh mana progres tugas yang telah diberikan. Setelah *sprint planning*, akan dilanjutkan dengan *sprint closing*, yang bertujuan untuk menutup dan mengevaluasi *sprint* sebelumnya. Selain itu, setiap 2–3 minggu sekali juga diadakan sesi *1-on-1* dengan *Technical Lead* untuk membahas kendala-kendala yang dihadapi selama mengerjakan proyek. Dalam sesi ini, *Technical Lead* akan memberikan *feedback* terkait permasalahan yang dialami, sehingga proses pengembangan dapat berjalan lebih baik ke depannya.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang, diajari beberapa hal seperti penggunaan *Playwright*, *JavaScript*, *TypeScript*, dan lain-lain. Adapun tugas-tugas yang di kerjakan di antaranya adalah:

- Membuat *automation* untuk melakukan pengecekan pada *website* MRO.
- Melakukan pengecekan fungsional secara manual di aplikasi MRO.
- Membuat *Test Case* versi 1.2.
- Membuat *report* dari hasil *automation website* MRO.
- Membuat *Test Case* versi 1.3.

3.3 Uraian Pelaksanaan Magang

Periode magang dimulai pada tanggal 3 Januari 2025 hingga 3 Juni 2025 di PT. Ganda Visi Jayatama (Concise), dengan mengerjakan beberapa tugas yang bertujuan untuk melakukan pengecekan terhadap *website* maupun aplikasi MRO. Pengecekan ini bertujuan untuk memastikan bahwa seluruh fitur dapat berjalan dengan lancar dan tanpa bug.
Untuk pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Masa pengenalan lingkungan kerja,peraturan kerja dan membaca boilerplate playwright

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (Lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
2	Membaca dan memahami code yang dibikin oleh *Junior Quality Assurance* dengan project yang lain
3	Mencoba untuk membuat automation dibagian *login* di *website HRIS*
4	Membuat *test case* untuk *website MRO*
5	Membuat code *automation website MRO* di halaman *Login page*
6	Membuat *Code Automation website MRO* di halaman *User N Role*
7	Membuat *Code Automation Website MRO* di halaman *Master Data*
8	Membuat *Code Automation Website MRO di Vessel Requisition*
9	Memperbaiki *code Automation* sesuai dengan SOP Perusahaan
10	Melakukan *Testing* dari awal hingga akhir
11	Membuat hasil *report* dari *test automation* pada *website MRO*
12	*Deplyoment* melalui Github perusahaan
13	Membuat *test case* versi 1.3 untuk *website MRO*
14	Melakukan *testing manual* pada aplikasi MRO
15	Membuat serta memberikan hasil *report ke Project Manager*
16	Melakukan *testing manual* versi 1.2 pada aplikasi MRO

Pada saat mengerjakan tugas dan proyek di perusahaan, mahasiswa juga menggunakan beberapa *tools* untuk kebutuhan dalam menjalankan tugas, yaitu:

- Playwright

Gambar 3.2. Logo Playwright

Gambar 3.2 Playwright merupakan alat *open-source* untuk otomatisasi *browser* yang dikembangkan oleh Microsoft dan mulai diperkenalkan pada

tahun 2020. Alat ini dirancang untuk melakukan pengujian *end-to-end* pada aplikasi web modern. Keunggulan Playwright terletak pada kemampuannya yang lintas platform, lintas *browser*, dan mendukung berbagai bahasa pemrograman. Salah satu fitur menariknya adalah *auto-waiting*, yang membantu proses pengujian menjadi lebih mudah, bahkan pada proyek yang kompleks. Playwright juga dikenal memiliki performa eksekusi yang cepat, sehingga banyak dipilih untuk kebutuhan *testing* otomatis. Dengan memanfaatkan *library Node.js*, Playwright menyediakan *API* untuk mengendalikan *browser* melalui protokol *DevTools*. Alat ini mendukung pengujian di *browser* populer seperti *Chromium*, *Firefox*, dan *WebKit*, serta dapat digunakan dengan bahasa pemrograman seperti *JavaScript*, *TypeScript*, *Python*, *Java*, dan *.NET*. Salah satu fitur andalannya adalah Playwright *Codegen*, yang bisa merekam interaksi pengguna di situs web dan secara otomatis menghasilkan kode pemrograman dari aktivitas tersebut.. [1]

- Visual Code Studio

Gambar 3.3. Logo Visual Code Studio

Gambar 3.4 Visual Studio Code merupakan editor kode yang dibuat oleh Microsoft, salah satu perusahaan teknologi terkemuka. Editor ini bisa digunakan di berbagai sistem operasi *desktop*, termasuk *macOS*, *Windows*, dan *Linux*. Visual Studio Code terkenal karena fleksibilitasnya yang tinggi serta dukungannya terhadap berbagai bahasa pemrograman, seperti *JavaScript*, *TypeScript*, *Python*, *PHP*, hingga *Java*. [2]

- Jira

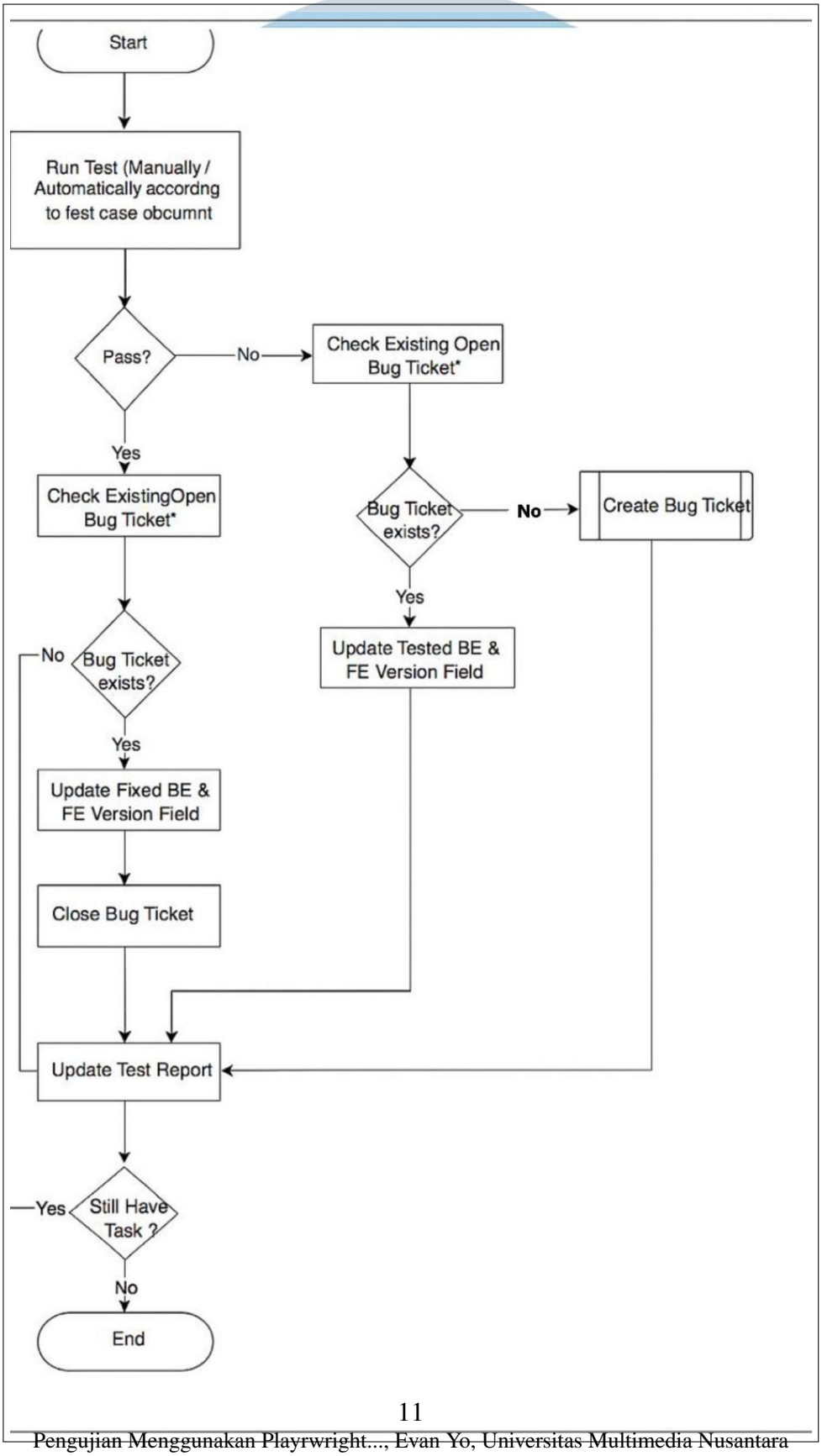
Gambar 3.4. Logo Jira

Gambar 3.5 Jira merupakan sebuah alat bantu dalam pengembangan *perangkat lunak* yang berfungsi untuk mengelola dan memantau *task*, *feature*, serta *bug*. Aplikasi ini dikembangkan oleh Atlassian, sebuah perusahaan teknologi yang berbasis di Australia. Jira menyediakan berbagai macam fitur yang mempermudah proses pengelolaan proyek, terutama dalam konteks pengembangan *perangkat lunak*.

Selain itu, Jira juga memfasilitasi perusahaan dalam memantau kebutuhan proyek dan perkembangannya secara menyeluruh. Dukungan komunitas dan keberadaan berbagai *forum* menjadikan penggunaannya semakin luas dan mudah dipelajari. Secara umum, Jira banyak digunakan dalam proses *Application Lifecycle Management (ALM)*, manajemen pengujian, dan pengelolaan proyek secara keseluruhan, terutama untuk memantau *bug* serta permasalahan lainnya yang muncul dalam pengembangan aplikasi.[3].

Pada perusahaan ini, Jira digunakan untuk mengatur *planning* dalam pelaksanaan *task* oleh tim. Tim nantinya akan diberikan *sprint planning* selama dua minggu sebagai bagian dari proses pengerjaan tugas dan proyek di perusahaan.

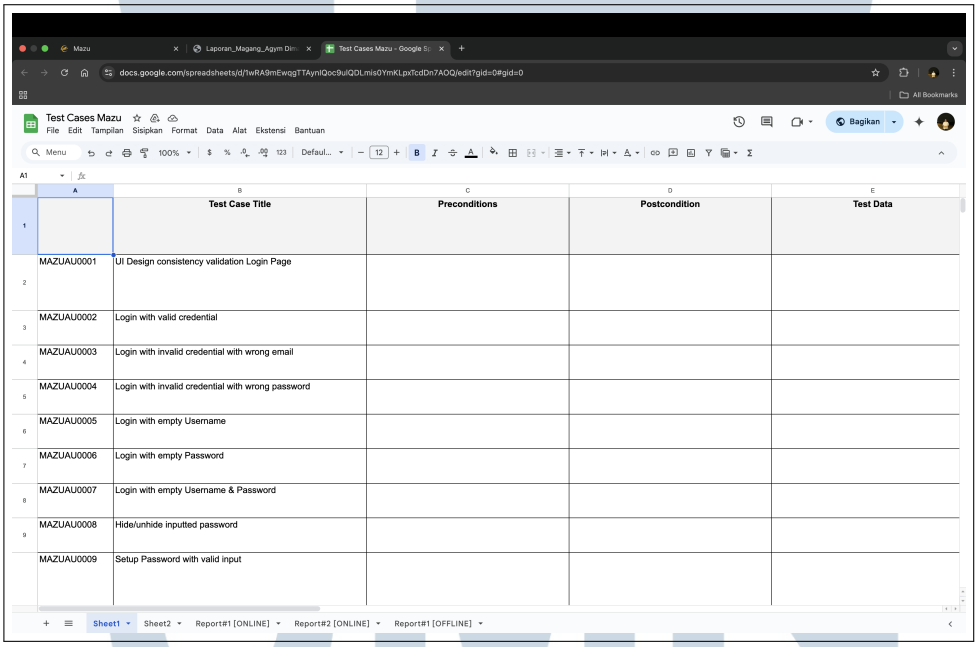
3.3.1 Flowchart



Gambar 3.5. Flowchart cara kerja Quality Assurance

3.3.2 Tugas 1: Membuat Test Case untuk website MRO

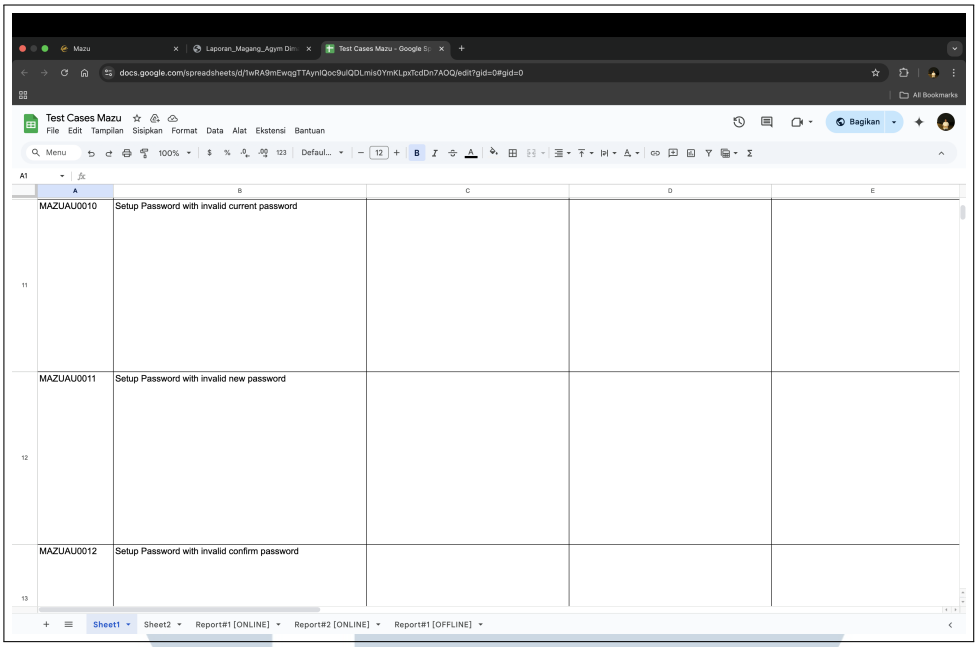
Tugas pertama dalam pelaksanaan kerja magang di PT. Ganda Jaya Tama adalah membuat *test case* terlebih dahulu, yang bertujuan untuk memastikan apakah aplikasi atau *website* berjalan sesuai aturan atau tidak. Hal ini dilakukan untuk mengetahui bug yang mungkin muncul, serta memberikan masukan terhadap pengembangan sistem. Proses ini biasanya dilakukan sebelum *deployment*, agar dapat meminimalisir terjadinya *bug* atau hal-hal lain yang tidak sesuai dengan permintaan. Di dalam sebuah *test case* terdapat beberapa elemen penting, yaitu: *test case title*, *precondition*, *postcondition*, *test data*, *steps*, dan *actual result*.
Berikut adalah gambaran *test case* yang dibuat pada *website* MRO.



	Test Case Title	Preconditions	Postcondition	Test Data
MAZUAU0001	UI Design consistency validation Login Page			
MAZUAU0002	Login with valid credential			
MAZUAU0003	Login with invalid credential with wrong email			
MAZUAU0004	Login with invalid credential with wrong password			
MAZUAU0005	Login with empty Username			
MAZUAU0006	Login with empty Password			
MAZUAU0007	Login with empty Username & Password			
MAZUAU0008	Hide/unhide inputted password			
MAZUAU0009	Setup Password with valid input			

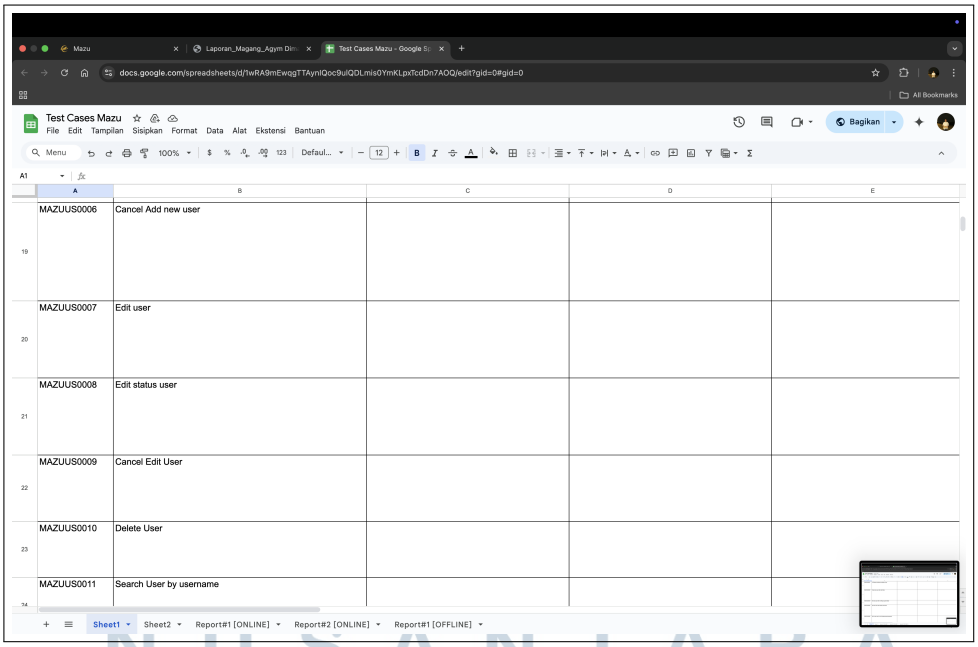
Gambar 3.6. Test Case

Gambar 3.6 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *login page*



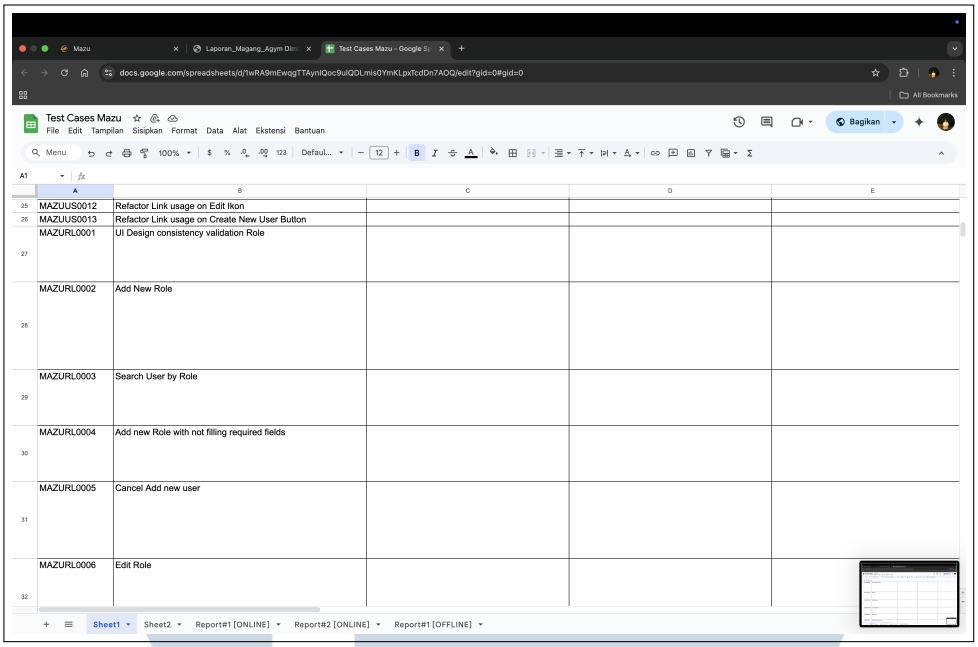
Gambar 3.7. Test case

Gambar 3.7 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *login page*



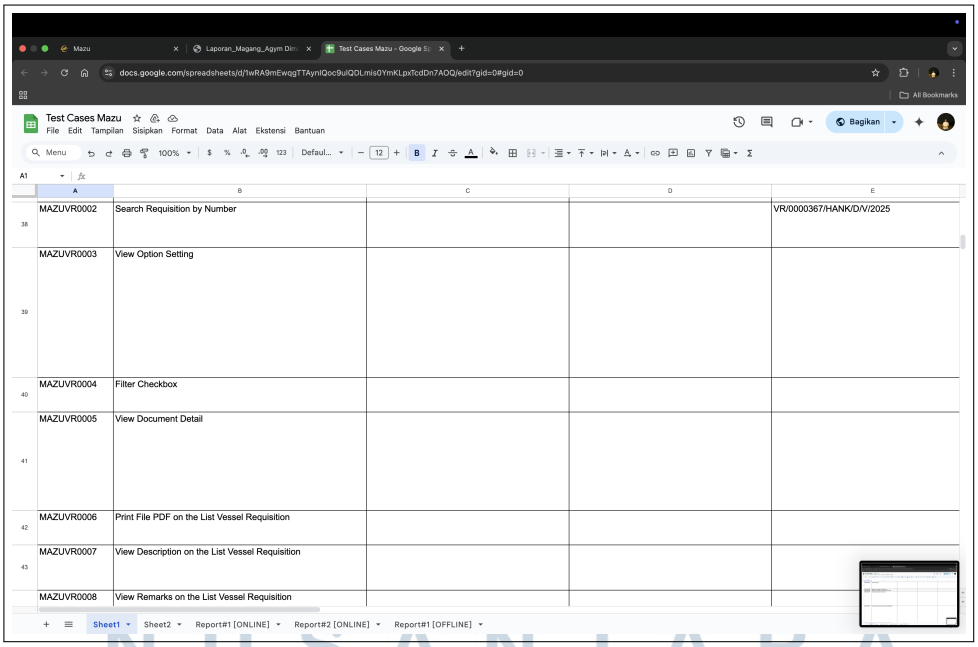
Gambar 3.8. Test case

Gambar 3.8 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *user page*



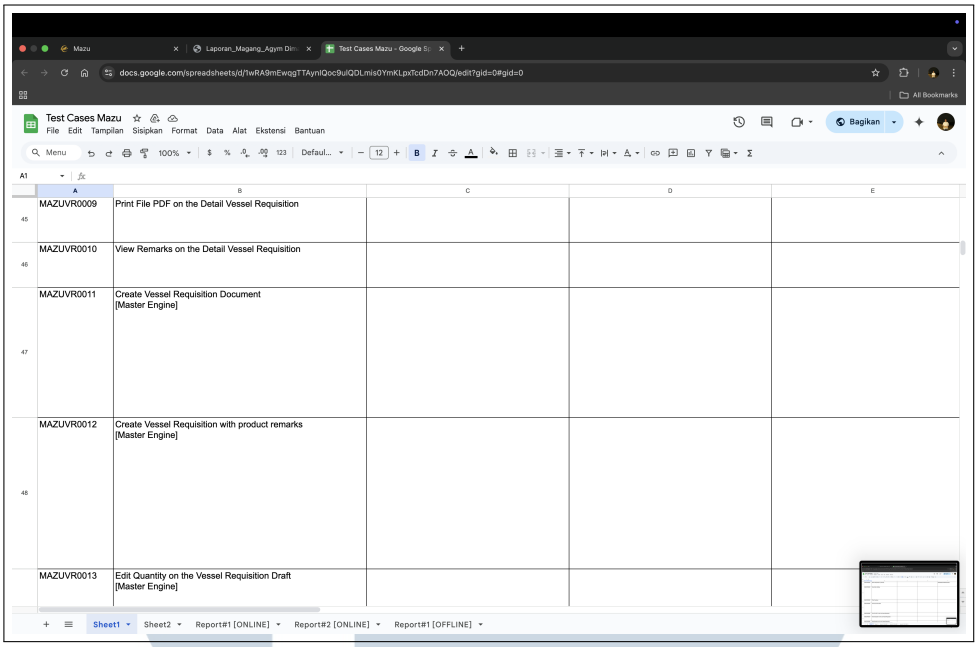
Gambar 3.9. Test case

Gambar 3.9 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *role page*



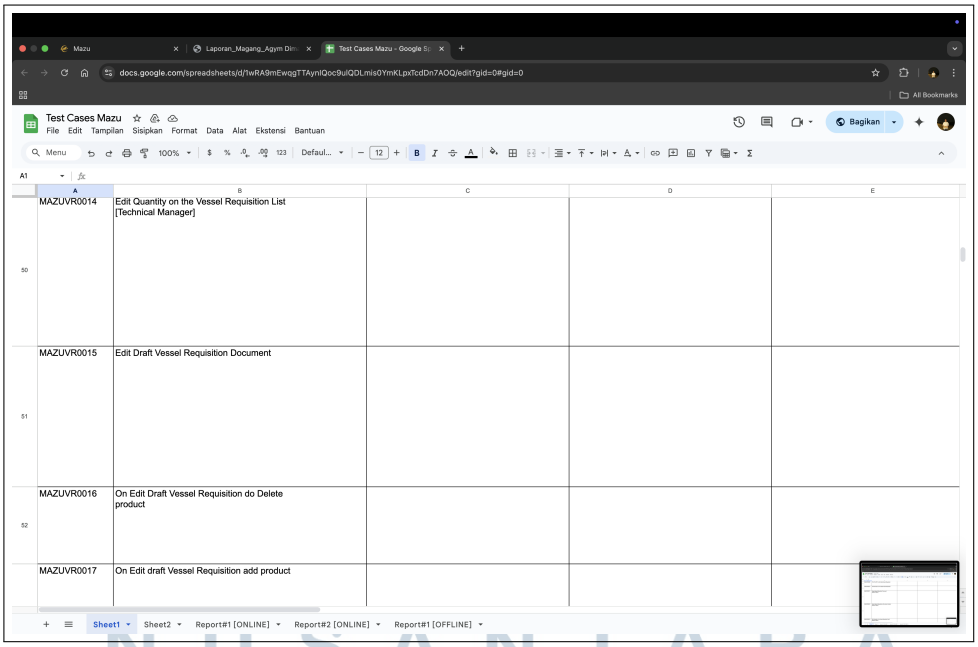
Gambar 3.10. Test case

Gambar 3.10 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *vessel requisition page*



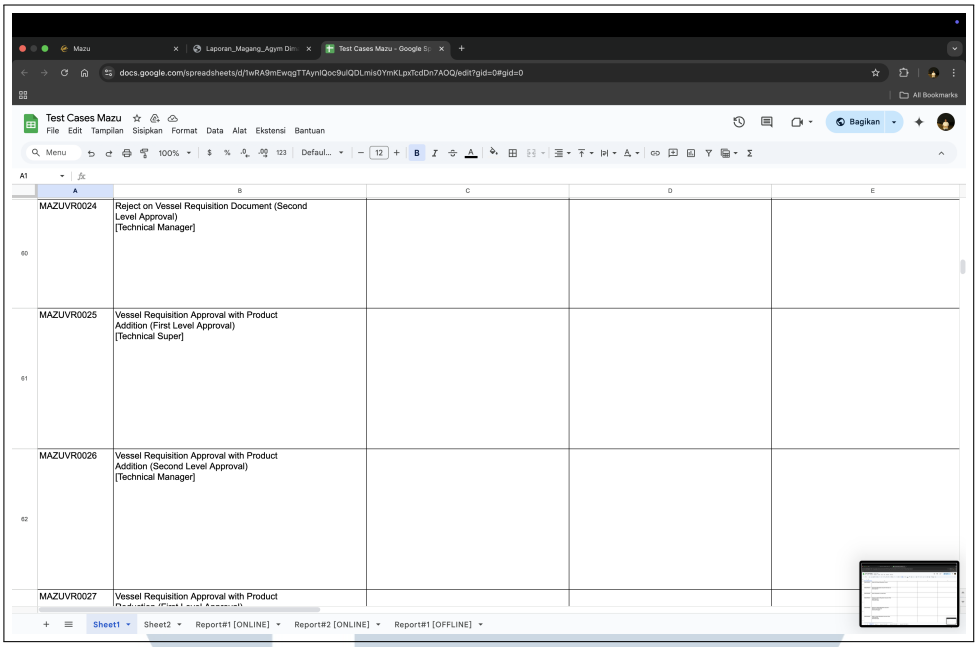
Gambar 3.11. Test case

Gambar 3.11 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *vessel requisition page*



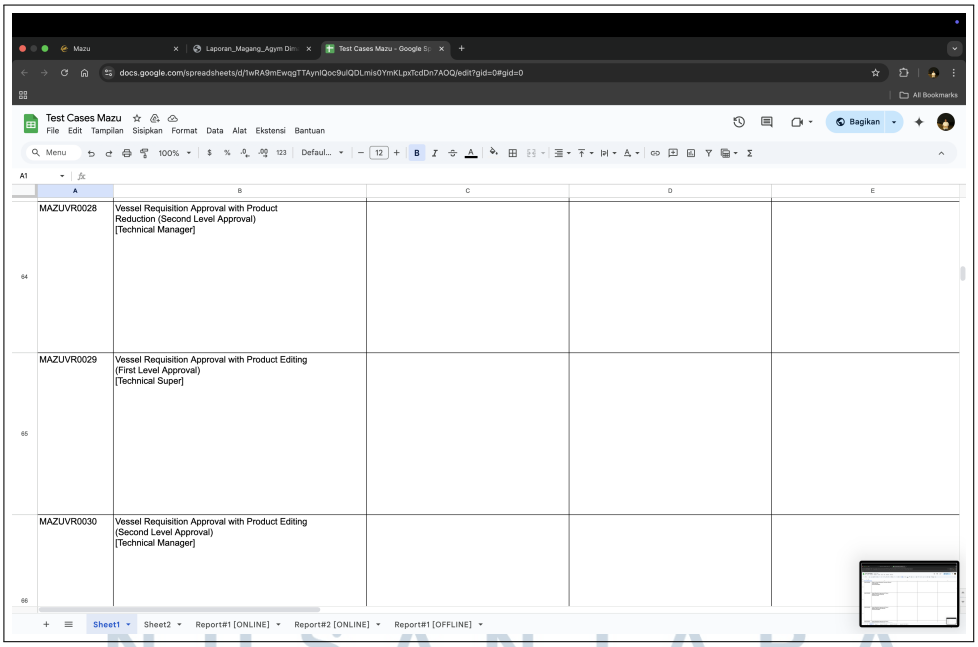
Gambar 3.12. Test case

Gambar 3.12 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *vessel requisition page*



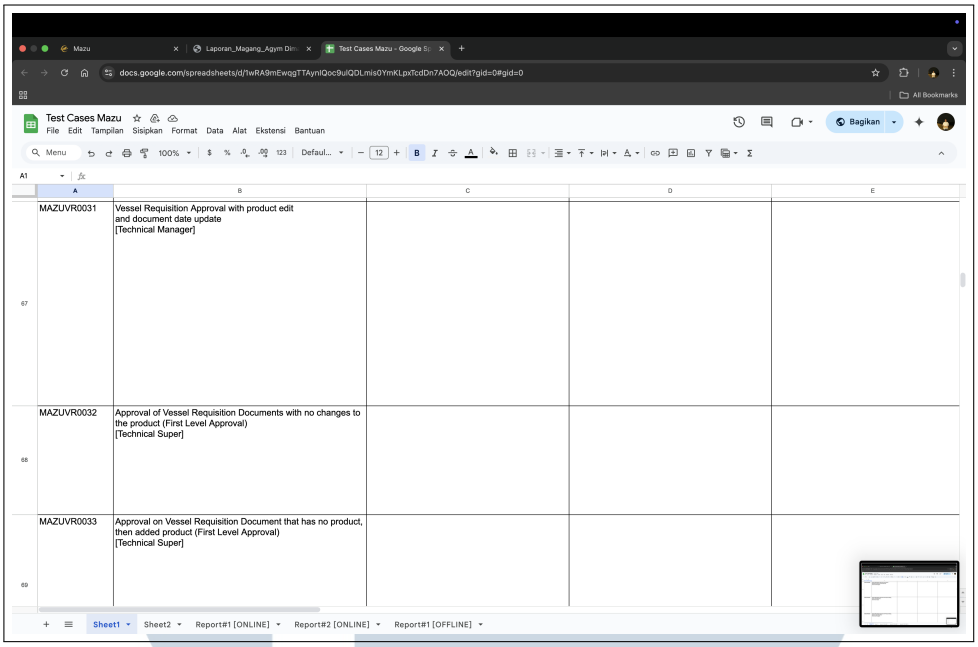
Gambar 3.13. Test case

Gambar 3.13 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *vessel requisition page*



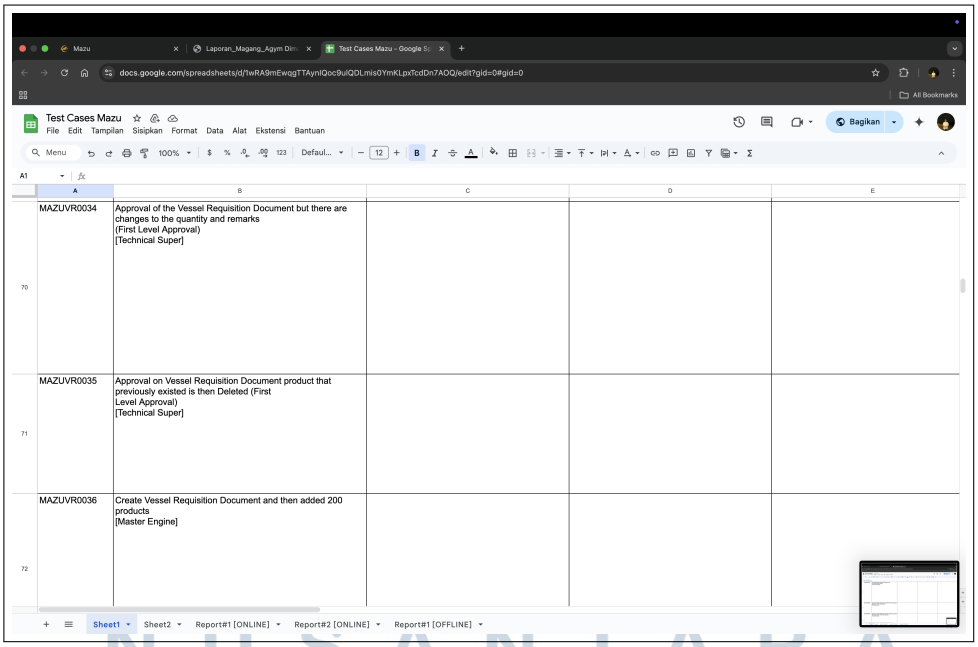
Gambar 3.14. Test case

Gambar 3.14 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *vessel requisition page*



Gambar 3.15. Test case

Gambar 3.15 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *vessel requisition page*

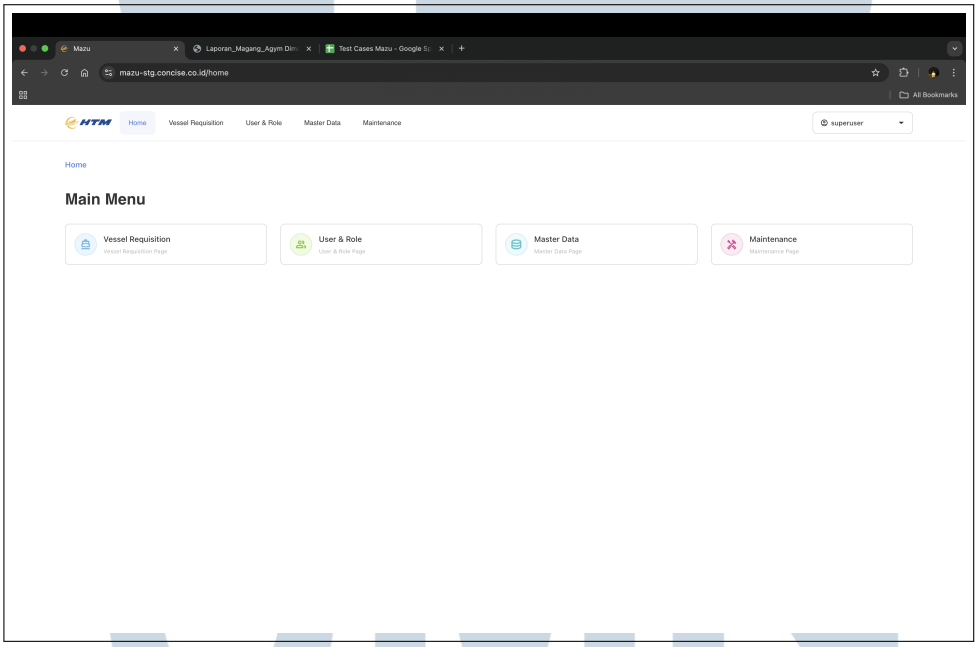


Gambar 3.16. Test case

Gambar 3.16 Berikut adalah hasil *test case* yang telah dibuat, *test case* tersebut mengecheck dibagian *vessel requisition page*

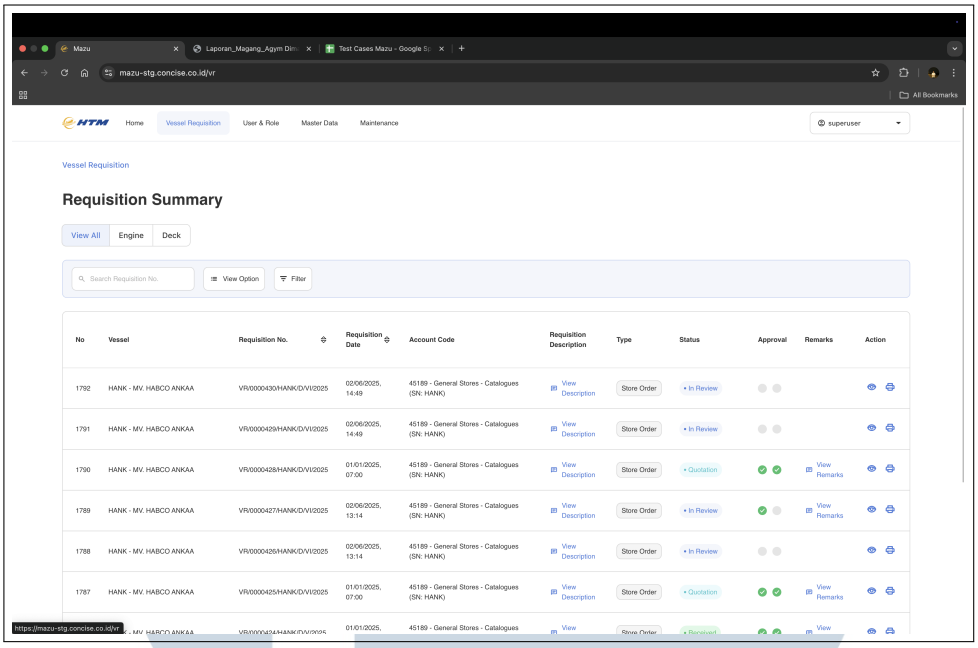
3.3.3 Tugas 2: Membuat *Code* untuk Melakukan *Automation Website* MRO

Setelah selesai membuat *test case*, untuk tugas , pihak kantor memberikan tugas lanjutan, yaitu membuat *automation* pada *website* MRO. Sebelumnya, sudah diajarkan bagaimana cara menggunakan Playwright serta pengenalan bahasa *JavaScript* dan *TypeScript*. dibimbing langsung oleh dua *Junior Quality Assurance*, yaitu Bapak Jovan Haliem dan Bapak Agym Dimas Tian. Untuk penugasan awal yang diberikan, pihak kantor meminta membuat *automation* untuk *login page*, kemudian dilanjutkan dengan *User n Role*, *Master Data*, dan yang terakhir adalah *Vessel Requisition*. Untuk setiap *module*, terdapat banyak *fitur-fitur* yang perlu diuji. Berikut merupakan gambaran *website* serta potongan *code* dari *website* MRO.



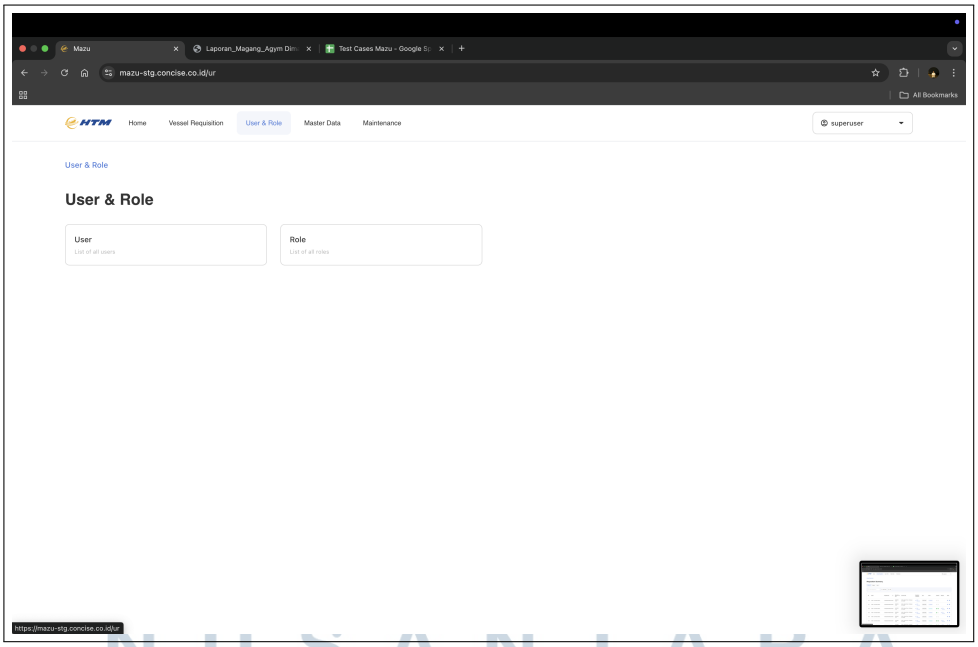
Gambar 3.17. Tampilan Home Page pada website MRO

Gambar 3.17 Tampilan awal beranda utama website setelah login.



Gambar 3.18. Tampilan Vessel Requisition pada website MRO

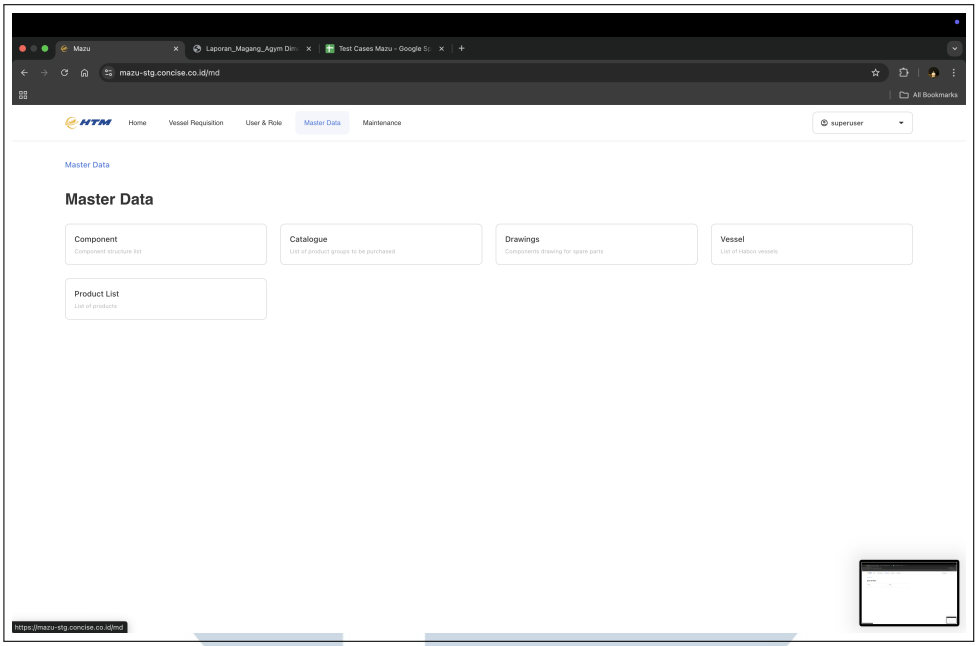
Gambar 3.18 Merupakan tampilan Vessel Requisition, yang berisikan pemesanan perlengkapan yang dibutuhkan kapal



Gambar 3.19. Tampilan User n Role pada website MRO

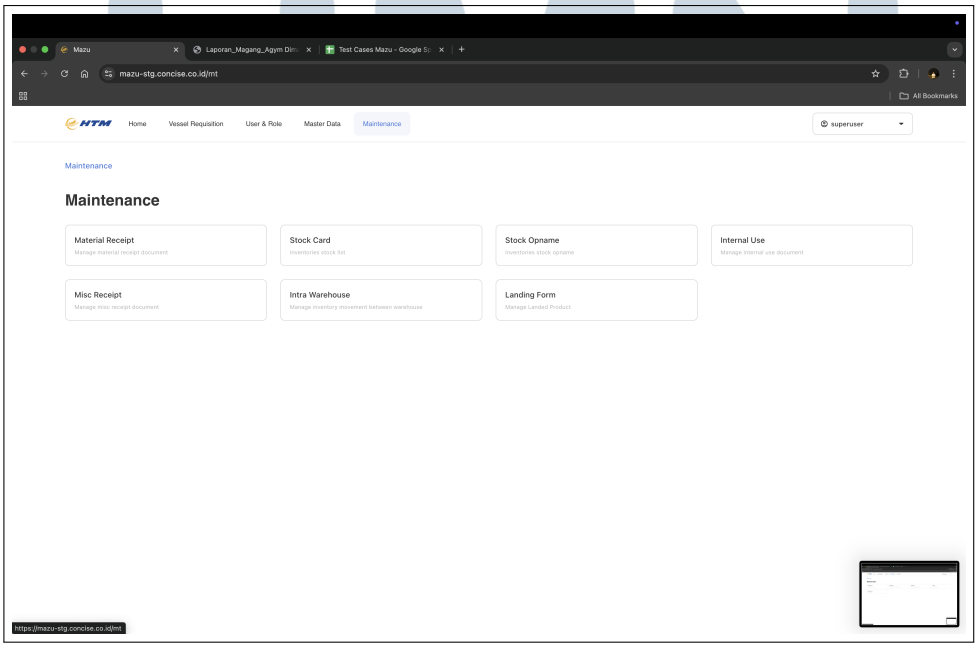
Gambar 3.19 Merupakan tampilan User n Role, yang memiliki 2 tujuan dari setiap fitur. User bertujuan untuk membuat user baru saja, jika role bertujuan untuk

memberikan role ke setiap user.



Gambar 3.20. Tampilan Master Data pada website MRO

Gambar 3.20 Merupakan tampilan Master Data, yang berisikan beberapa fitur, yaitu component, catalogue, drawings, vessel, dan product list. Di setiap fitur memiliki fungsinya masing-masing.



Gambar 3.21. Tampilan Maintenance pada website MRO

Gambar 3.21 merupakan tampilan Maintenance, yang berisikan beberapa fitur yaitu, material receipt, stock card, stock opname, internal use, misc receipt, intra warehouse, dan landing form. masing-masing dari fitur memiliki fungsinya sendiri, salah satunya ada untuk mengecek ketersediaan barang kapan, serta dapat juga memesan barang yang diperlukan.

```
import { Page, Locator } from '@playwright/test'
import dotenv from 'dotenv'
import { BasePage } from './basePage'

// Load environment variables
dotenv.config()

export class LoginPage extends BasePage {
  private readonly usernameField: Locator
  private readonly passwordField: Locator
  private readonly loginButton: Locator
  // private readonly messageLocator: Locator
  private readonly currentPasswordField: Locator
  private readonly newPasswordField: Locator
  private readonly confirmPasswordField: Locator

  constructor(public readonly page: Page) {
    super(page)
    this.usernameField = page.getByPlaceholder('Your username')
    this.passwordField = page.getByPlaceholder('Your Password')
    this.loginButton = page.locator("[data-testid='login_btn']")
    // this.messageLocator = page.locator('.toastMessages')
    this.currentPasswordField = page.locator('#currpassword')
    this.newPasswordField = page.locator('#newpassword')
    this.confirmPasswordField = page.locator('#confirmpassword')
  }
}
```

Kode 3.1: Kode TypeScript untuk Halaman Login Playwright

```
import type { Locator, Page, expect } from '@playwright/test'
import { randomNumber, randomString } from '../utils/
    randomGenerator'
import { BasePage } from './basePage'

export class VesselPage extends BasePage {
  locator(arg0: string) {
    throw new Error('Method not implemented.')
```

```
8   }
9   private tempDescription: string
10
11  constructor(page: Page) {
12    super(page)
13  }
14
15  async navigateMenu() {
16    await this.page.getByRole('link', { name: 'Vessel Requisition'
      }).first().click()
17  }
18
19  async waitLoaderAndHeader() {
20    await this.waitForLoaderAndHeader(/Requisition Summary/)
21  }
22
23  async addNewRequisitionBtn(desc: string) {
24    await this.page.waitForTimeout(3000)
25    await this.page.getByRole('button', { name: 'Create New
     Requisition' }).click()
26
27    // Requisition Type Field
28    await this.page.locator('#Input_RequisitionType').click()
29    await this.page.getByTestId('Input_RequisitionType_Option_1').
     click()
30
31    // Requisition Account Code
32    await this.page.locator('#Input_AccountCode').click()
33    await this.page.getByTestId('Input_AccountCode_Option_2').
     click();
34
35    // Requisition Category
36    await this.page.locator('#Input_Category').click()
37    await this.page.getByTestId('Input_Category_Option_2').click()
38
39    // Description
40    await this.page.getByTestId('Input_RequisitionDescription').
     fill(desc);
41
42    await this.page.getByTestId('PartsItems_Button_AddPartsItems')
     .click();
43
44    // First Locator
```

```
45      await this.page.locator('#ModalPartsItems_Input_FirstLocator')
        .click()
46      await this.page.getByTestId('
        ModalPartsItems_Input_FirstLocator_Option_1').click()
47
48      // Second Locator
49      await this.page.locator('#ModalPartsItems_Input_SecondLocator
        ').click()
50      await this.page.getByTestId('
        ModalPartsItems_Input_SecondLocator_Option_1').click()
51
52      // Catalogue
53      await this.page.locator('#ModalPartsItems_Input_Catalogue').
        click()
54      await this.page.getByTestId('
        ModalPartsItems_Input_Catalogue_Option_1').click()
55    }
56  }
```

Kode 3.2: Kode TypeScript untuk Halaman Vessel Requisition

```
1  import type { Locator, Page } from '@playwright/test'
2  import { randomNumber, randomString } from '../utils/
       randomGenerator'
3  import { BasePage } from './basePage'
4
5  export class UserPage extends BasePage {
6    private tempUserName: string
7
8    constructor(page: Page) {
9      super(page)
10   }
11
12 }
```

Kode 3.3: Kode TypeScript untuk Halaman UserPage

```
1  import type { Locator, Page } from '@playwright/test'
2  import { randomString } from '../utils/randomGenerator'
3  import { BasePage } from './basePage'
4
5  export class RolePage extends BasePage {
6    private tempRoleName: string
7
8    constructor(page: Page) {
```

```
    super(page)
  }

  async navigateMenu() {
    await this.page.getByRole('link', { name: 'Role' }).first().
   click()
  }

  async waitLoaderAndHeader() {
    await this.waitForLoaderAndHeader(/Role/)
  }

  async addCreateNewRole(name: string) {
    this.addNewRoleBtn(name)
    await this.page.getByRole('button', { name: 'Save Role' }).
   click()

    await this.page.waitForLoadState('networkidle')
    await this.page.getByTestId('primary_btn').waitFor()
    await this.page.getByTestId('primary_btn').click()
  }

  async generateTempRoleName(length: number = 3) {
    this.tempRoleName = await randomString(length)
    return this.tempRoleName
  }

  async getTempRoleName() {
    return this.tempRoleName
  }

  async addNewRoleBtn(name: string) {
    await this.page.getByRole('button', { name: 'Add New Role' }).
   click()
    await this.page.getByPlaceholder('Input Role Name').fill(name)
  }

  async removeRoleBtn(filter: Locator) {
    const rowLocator = this.page.locator('.roles')
      .getByRole('listitem')
      .filter({ has: filter })
      .locator('.trash_btn')

```

```
    await rowLocator.waitFor()
    await rowLocator.click()
  }
}
```

Kode 3.4: Kode TypeScript untuk Halaman RolePage

```
import type { Locator, Page, expect } from '@playwright/test'
import { randomNumber, randomString } from '../utils/
    randomGenerator'
import { BasePage } from './basePage'

export class CataloguePage extends BasePage {
  private tempCatalogueName: string
    tempDescription: string

  constructor(page: Page) {
    super(page)
  }

  async navigateMenu() {
    await this.page.getByRole('link', { name: 'Master Data' }).
    first().click()
    await this.page.locator('.mainMenu').getByText('Catalogue').
    first().click()
  }

  async addCatalogueBtn(name: string) {
    await this.page.getByTestId(name).click();
  }

  async waitLoaderAndHeader() {
    await this.waitForLoaderAndHeader(/Catalogue/)
  }

  async createNewCatalogue() {
    await this.page.getByRole('button', { name: 'Create New
    Catalogue' }).click();
  }

  async fillCatalogueName(name: string) {
    await this.page.getByText('New Catalogue').waitFor()
    await this.page.getByTestId('Input_CatalogueName').waitFor()
    const inputField = this.page.getByTestId('Input_CatalogueName
```

```
    ');
    await inputField.fill(name);
    await inputField.click();
    await inputField.click();
  }

  async addProductToCatalogue() {
    await this.page.getByRole('button', { name: 'Add Product to
    Catalogue' }).click();

    await this.page.getByTestId('ModalProduct_CheckBox_0').click()
    ;
    await this.page.getByTestId('addProd_btn').click();
  }

  async editCatalogueBtn(name: string) {
    await this.page.getByTestId('Button_Edit_0').click();
    await this.page.getByTestId('Input_CatalogueName').fill(name);
    await this.page.getByTestId('Button_Save').click();
    await this.page.getByTestId('ModalConfirmation_Button_Primary
    ').click();
  }
}
```

Kode 3.5: Kode TypeScript untuk Halaman CataloguePage

```
import type { Locator, Page, expect } from '@playwright/test'
import { randomNumber, randomString } from '../utils/
    randomGenerator'
import { BasePage } from './basePage'

export class DrawingsPage extends BasePage {
  private tempDrawingsName: string

  constructor(page: Page) {
    super(page)
  }
}
```

Kode 3.6: Kode TypeScript untuk Halaman DrawingsPage

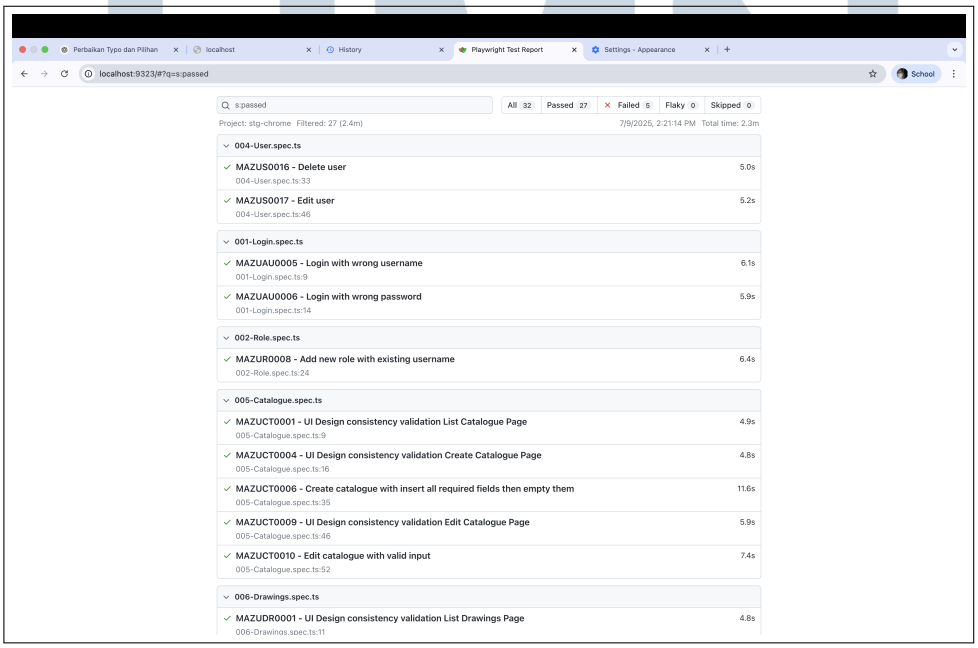
```
import type { Locator, Page, expect } from '@playwright/test'
import { randomNumber, randomString } from '../utils/
    randomGenerator'
import { BasePage } from './basePage'
```

```
4
5  export class VesselMDPage extends BasePage {
6    private tempVesselMDName: string
7
8    constructor(page: Page) {
9      super(page)
10   }
11 }
```

Kode 3.7: Kode TypeScript untuk Halaman VesselMDPage

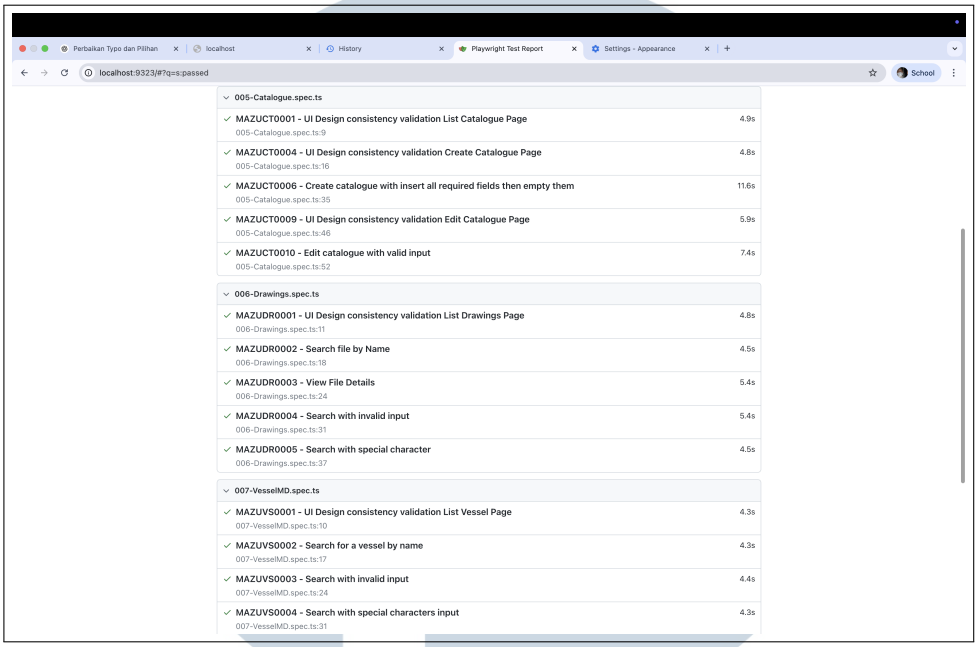
```
1  import type { Locator, Page, expect } from '@playwright/test'
2  import { randomNumber, randomString } from '../utils/
       randomGenerator'
3  import { BasePage } from './basePage'
4
5  export class ProductListPage extends BasePage {
6    private tempProductListName: string
7
8    constructor(page: Page) {
9      super(page)
10   }
11 }
```

Kode 3.8: Kode TypeScript untuk Halaman ProductListPage



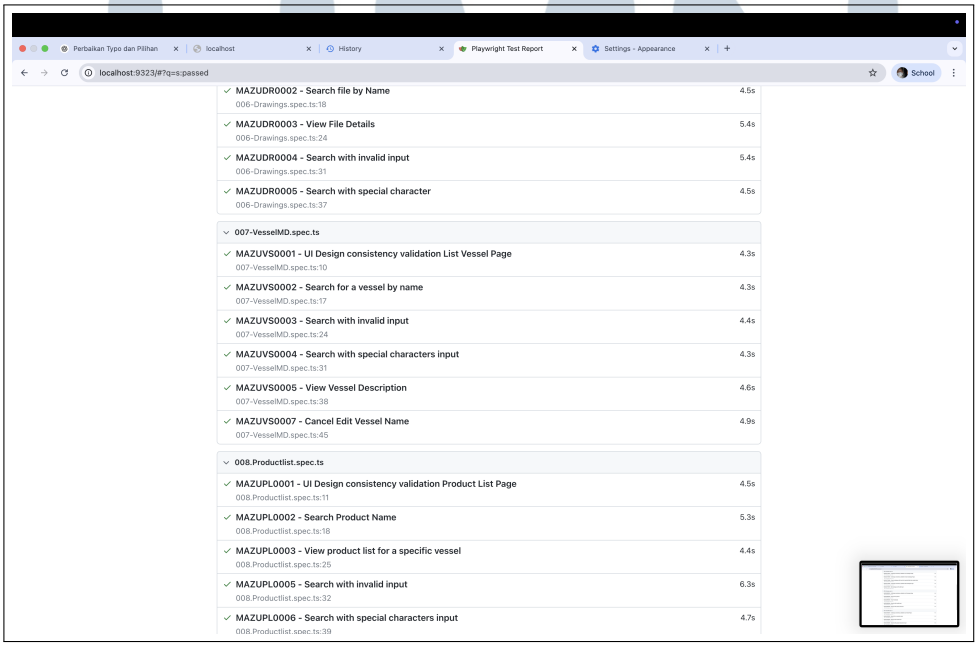
Gambar 3.22. Hasil Report setelah melakukan testing secara automation

Gambar 3.22 Menunjukan bahwa *testing* yang dilakukan dijalankan lancar, dan tidak memiliki *failed*



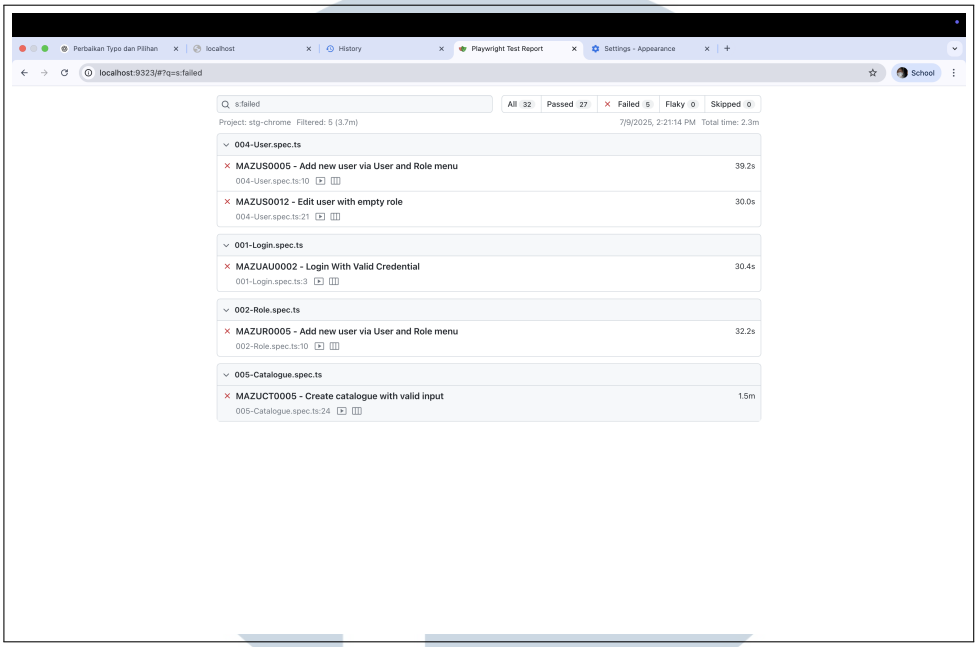
Gambar 3.23. Hasil Report setelah melakukan testing secara automation

Gambar 3.23 Menunjukan bahwa *testing* yang dilakukan dijalankan lancar, dan tidak memiliki *failed*



Gambar 3.24. Hasil Report setelah melakukan testing secara automation

Gambar 3.24 Menunjukan bahwa *testing* yang dilakukan dijalankan lancar, dan tidak memiliki *failed*



Gambar 3.25. Hasil Report setelah melakukan testing secara automation

Gambar 3.25 Menunjukan bahwa *testing* yang dilakukan dijalankan tidak lancar, dan memiliki *failed*

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala

Selama proses pengerjaan website dan aplikasi di PT. Ganda Jaya Tama, terdapat beberapa kendala yang dihadapi, yaitu

- Dalam code automation, masih mengunakan bahasa pemograman Javascript, sedangkan bahasa yang sekarang digunakan TypeScript.
- Masih terdapat hasil test yang kurang mendetail berdasarkan dari code Javascript sebelumnya

3.4.2 Solusi

Beberapa solusi yang diterapkan untuk mengatasi kendala-kendala tersebut adalah:

- Mengintegrasikan website tersebut mengunakan bahasa pemograman Typescript
- Membuat automation yang mendetail, sehingga meminimalisir terjadinya bug