BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Selama menjalani program magang di PT Emos Global Digital, peserta magang ditempatkan sebagai *front-end developer* di divisi *Information Technology (IT)*, dengan bimbingan dari Saudari Laras Nur Putri Arini sebagai pembimbing lapangan dan Bapak Heru Setiawan sebagai Manajer IT. Dalam peran tersebut, peserta magang bertanggung jawab untuk merancang tampilan situs web sesuai dengan kebutuhan perusahaan. Selain itu, mereka juga terlibat dalam pengembangan situs web internal untuk manajemen proyek atau bisnis perusahaan, serta situs web eksternal untuk manajemen bisnis klien.

3.2 Tugas yang Dilakukan

Selama melaksanakan program magang di PT EMOS Global Digital, peserta magang memiliki tugas sebagai *front-end developer*, yang bertugas dalam membuat tampilan pada *website* internal maupun eksternal yang dimiliki oleh perusahaan. Fitur Retrospective menjadi salah satu tugas yang dikerjakan oleh peserta magang, fitur ini merupakan fitur baru dari aplikasi MOSHELP yang memiliki fungsi utama yaitu sebagai media penyampaian kritik antar divisi internal perusahaan.

Setiap tugas yang dikerjakan merupakan permintaan langsung dari manajer IT ataupun *supervisor developer*. Tugas yang dikerjakan meliputi pengembangan *website* seperti penambahan fitur baru, peningkatan kerja sistem, dan perbaikan *bug*. Selain itu, peserta magang juga diberikan waktu untuk mempelajari lebih lanjut terkait teknologi terbaru yang memungkinkan untuk diimplementasikan oleh perusahaan.

3.3 Uraian Pelaksanaan Magang

Tabel 3.1 menunjukkan *timeline* program kerja magang di PT Emos Global Digital setiap minggunya.

Tabel 3.1. *Timeline* kerja magang setiap minggunya

Minggu Ke -	Pekerjaan yang dilakukan
1	Melaksanakan *meeting* dan desain awal fitur Retrospective
2	Menerapkan desain antarmuka tampilan *dashboard* utama fitur Retrospective ke dalam kode ReactJS
3	Menerapkan desain antarmuka tampilan detail Retrospective ke dalam kode ReactJS
4	Mengintegrasikan API dari *backend* ke dalam fitur Retrospective
5	Melakukan *testing* dan *bug fixing* pada fitur Retrospective
6	*Meeting* dan desain awal *event* EMOS Sewindu
7	Membuat tampilan *dashboard* untuk EMOS Sewindu *gamification*
8	Membuat tampilan *list both*, *leaderboard*, dan *promotion link* pada *website* EMOS Sewindu *gamification*
9	Mengintegrasikan API dari *backend* ke dalam *website event* EMOS Sewindu *gamification*
10	Melakukan *testing* dan *bug fixing* pada *website* EMOS Sewindu *gamification*
11	Membuat *dashboard report* pada *website* admin MOSLY
12	Melakukan *testing*, *bug fixing*, dan *refactor* pada fitur *dashboard report* MOSLY
13	Mempelajari *NextJS framework* dan desain 3D *website*
14	Mengintegrasikan desain Figma untuk tampilan ubah *password*, ubah PIN, dan ubah nomor *handphone* pada EMOS *marketplace*
15	Mengintegrasikan API untuk tampilan ubah *password*, ubah PIN, dan ubah nomor *handphone* pada EMOS *marketplace*
16	Mempelajari *NextJS* Authentication

3.3.1 Perangkat Pendukung Pengembangan Fitur Retrospective

Selama proses pengambangan fitur Retrospective perserta magang memanfaatkan perangkat keras dan perangkat lunak yang handal dalam pengembangan *website*.

Perangkat Lunak yang digunakan yaitu:

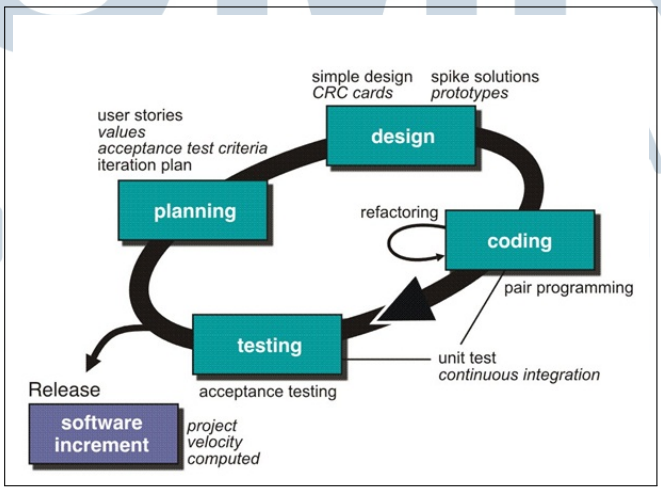
1. Google Chrome
2. Visual Studio Code
3. Postman
4. Figma

Perangkat keras berupa laptop memiliki spesifikasi sebagai berikut:

1. Tipe Laptop : Asus-Tuf 505DD
2. Prosesor : AMD Ryzen 5 3550H
3. RAM : 16 GB
4. Sistem Operasi : Windows 11 64-bit
5. Penyimpanan : SSD 512 GB
6. GPU : Nvidia Geforce GTX 1050

3.3.2 Metode Pengembangan Sistem

Selama pengembangan fitur Retrospective di PT Emos Global Digital, peserta magang menggunakan *agile development methods* dengan model *extreme programming*. *Extreme programming* merupakan sistem pengembangan perangkat lunak yang berfokus pada penyederhanaan tahapan pengembangan sistem sehingga menjadi lebih efisien, adaptif, dan fleksibel[4].



Gambar 3.1. Tahapan proses *extreme programming*

3.3.3 Proses Pengembangan Fitur Retrospective

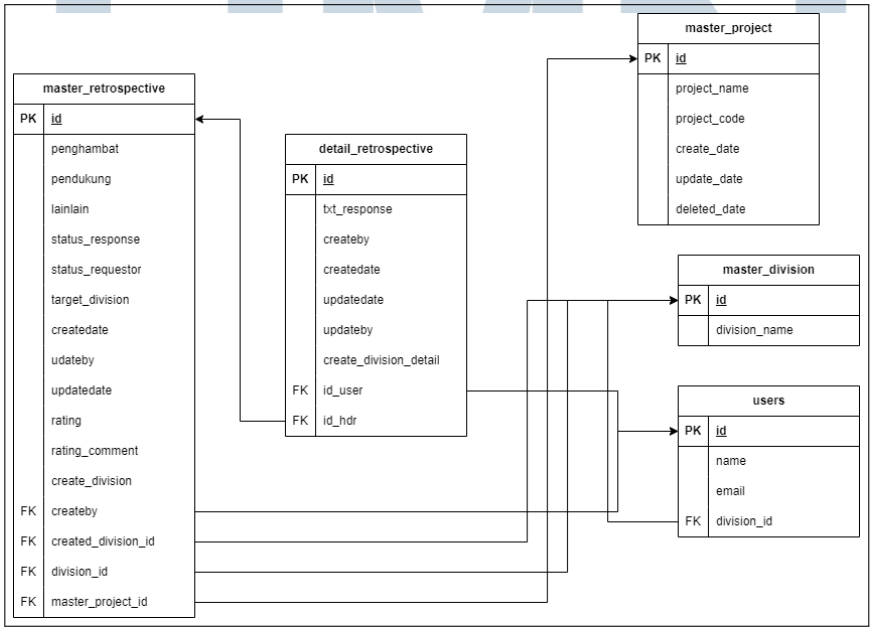
Pada bagian ini menjelaskan proses pengembangan fitur Retrospective. Proses Pengembangan meliputi *planning*, *desain*, *coding*, dan *testing*.

A. Planning

Pada tahap *planning*, dilakukan analisis kebutuhan sistem, termasuk fitur-fitur yang diperlukan, keluaran (*output*) yang diharapkan, serta fungsionalitas aplikasi yang dikembangkan. Tahap ini bertujuan untuk memahami dengan jelas apa yang harus dicapai oleh sistem. Pada tahap ini peserta magang membuat *Entity Relation Diagram* ERD dan *Flowchart* sebagai alat bantu untuk memvisualisasikan struktur data dan alur proses dalam sistem yang dikembangkan.

A.1 Entity Relation Diagram

Gambar 3.2 menampilkan *Entity Relationship Diagram* (ERD) untuk fitur Retrospective. Tabel *master_retrospective* berfungsi sebagai tabel utama yang menyimpan data terkait penyampaian kritik, seperti faktor penghambat, faktor pendukung, divisi target, divisi pengirim, serta kritik yang disampaikan. Tabel ini terhubung dengan beberapa tabel lain, yaitu *master_project*, *master_division*, *detail_retrospective*, dan *users*.

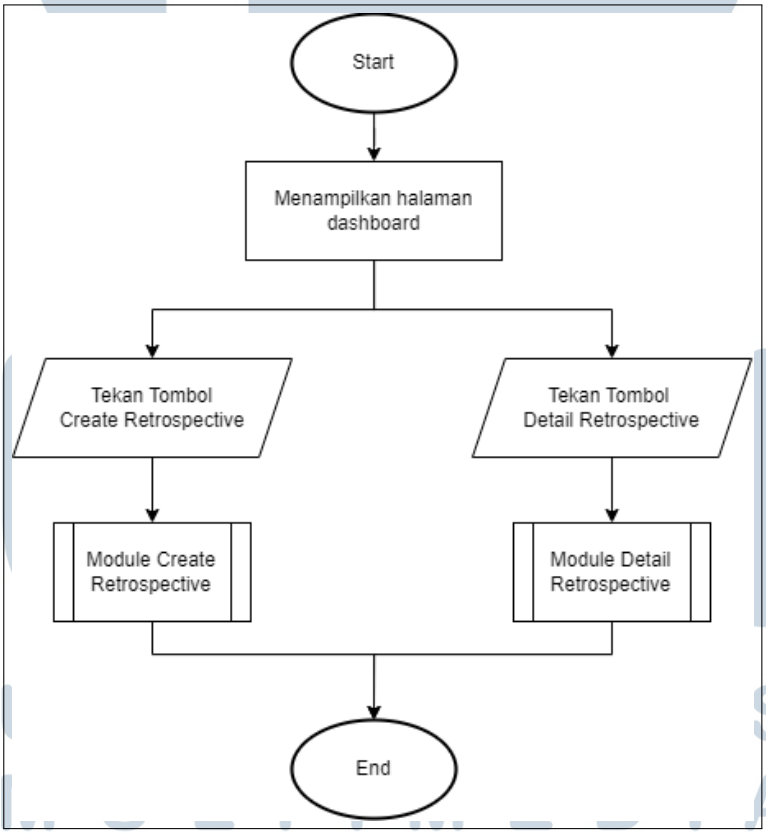


Gambar 3.2. Entity Relation Diagram module Retrospective

Setiap tabel tersebut memiliki fungsi masing-masing: *master_project* digunakan untuk menyimpan data proyek perusahaan, *master_division* berfungsi menyimpan informasi tentang divisi-divisi yang ada, *users* menampung data karyawan, dan *detail_retrospective* menyimpan detail retrospektif seperti umpan balik dari karyawan. Hubungan antar tabel ini memungkinkan pengelolaan dan pelacakan kritik serta saran dalam fitur Retrospective secara lebih efektif.

A.2 Flowchart

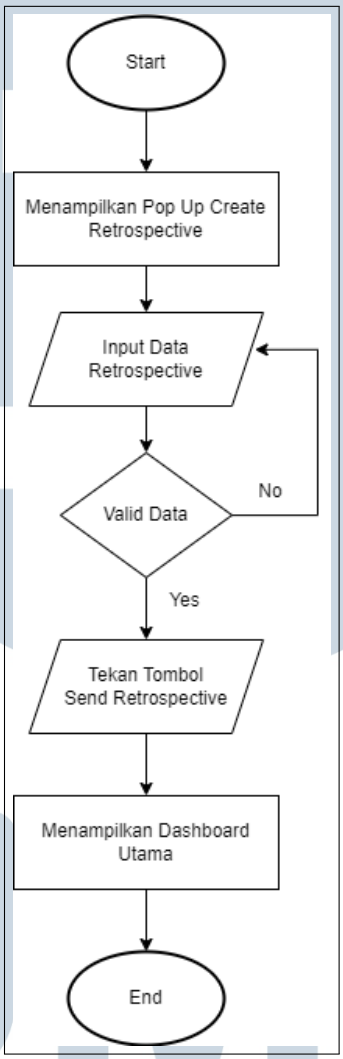
Gambar 3.3 menunjukkan bahwa ketika pengguna mengakses fitur Retrospective, yang pertama kali dikeluarkan oleh sistem adalah tampilan halaman *dashboard*. Pada halaman *dashboard* terdapat dua modul yang dapat diakses oleh pengguna, yaitu *Create Retrospective* dan *Detail Retrospective*.



Gambar 3.3. Flowchart *dashboard* Retrospective

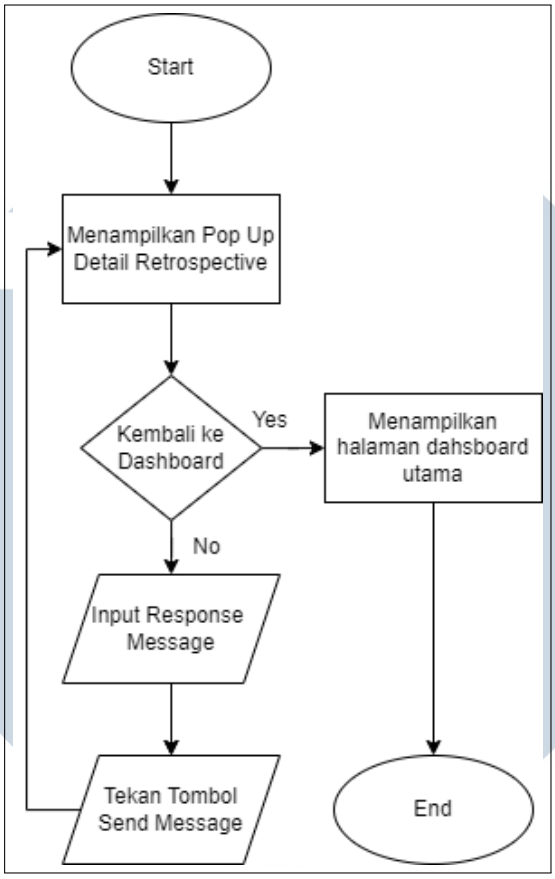
Gambar 3.4 menunjukkan bahwa ketika pengguna mengakses *Create Retrospective*, maka sistem mengeluarkan *pop up* berupa tampilan form untuk menambahkan Retrospective baru. Ketika *pop up* muncul, pengguna dapat memasukkan data Retrospective. Data yang dimasukkan akan divalidasi;

ketika data valid, pengguna berhasil mengirim Retrospective, namun jika data yang dimasukkan tidak valid, sistem memunculkan pesan *error* dan data yang dimasukkan gagal dikirimkan.



Gambar 3.4. Flowchart halaman tambah Retrospective

Gambar 3.5 menunjukkan bahwa ketika pengguna mengakses *Detail Retrospective*, maka sistem akan mengeluarkan *pop up* berupa detail dari Retrospective tersebut. Pada bagian ini, pengguna dapat mengirimkan *message*/pesan sebagai *feedback* dari Retrospective yang dibuat. Selain itu, pengguna juga dapat kembali ke halaman *dashboard* utama.



Gambar 3.5. Flowchart detail dan *feedback* Retrospective

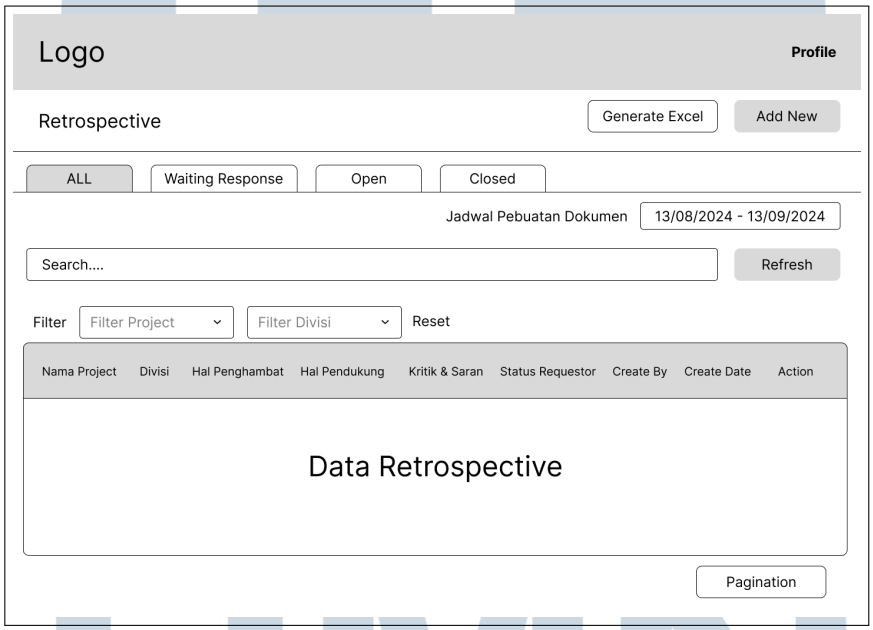
B. Desain

Dalam tahap perancangan tampilan fitur Retrospective, peserta magang memfokuskan pada pembuatan desain aplikasi yang tidak hanya menarik secara visual, tetapi juga intuitif dan responsif terhadap berbagai perangkat. Desain ini bertujuan untuk memastikan pengguna dapat dengan mudah dan nyaman berinteraksi dengan fitur Retrospective tanpa mengalami kesulitan dalam navigasi. Pemahaman mendalam mengenai pengalaman pengguna (UX) menjadi kunci utama dalam menentukan setiap elemen yang digunakan.

Peserta magang juga melakukan proses identifikasi yang teliti terhadap layout serta elemen-elemen yang diperlukan, seperti tata letak, komponen, dan ikon, yang tidak hanya sesuai dengan estetika aplikasi, tetapi juga mendukung fungsionalitasnya. Tampilan UI seperti font, warna, dan ikon dibuat menyesuaikan dengan website MOSHELP yang merupakan aplikasi utama dalam fitur ini. Proses desain juga melibatkan umpan balik dari atasan, dan setiap elemen pada desain juga disesuaikan untuk memastikan pengalaman pengguna yang maksimal.

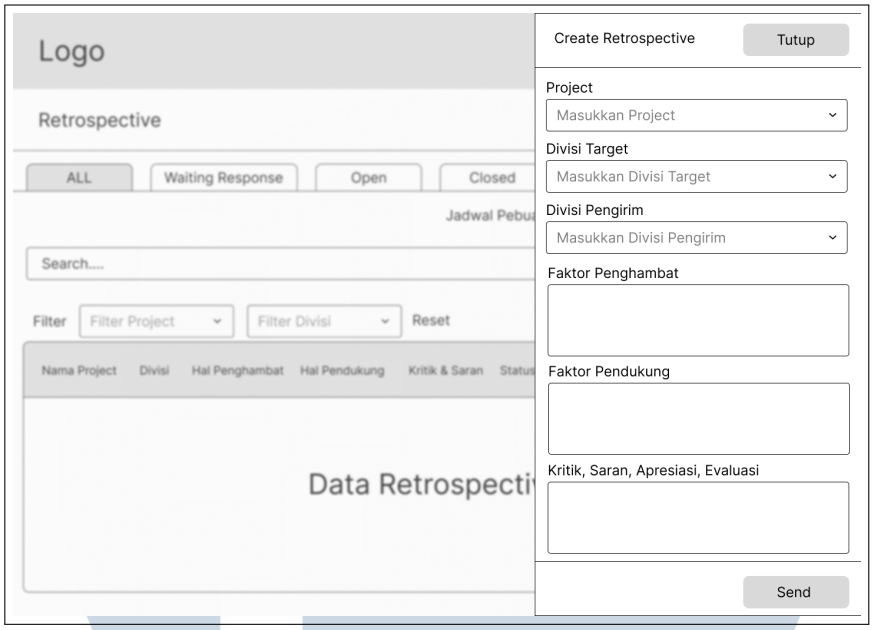
B.1 Mockup Website

Gambar 3.6 merupakan mockup halaman *dashboard* Retrospective. Halaman ini menjadi halaman utama yang diakses oleh pengguna ketika menggunakan fitur Retrospective. Pada halaman ini, pengguna dapat melihat list data Retrospective yang dimiliki. List data tersebut dilengkapi dengan fitur *searching*, *tab pagination*, dan *filter*. Di halaman ini juga terdapat tombol untuk menambah Retrospective baru yang menavigasikan pengguna ke halaman tambah Retrospective.



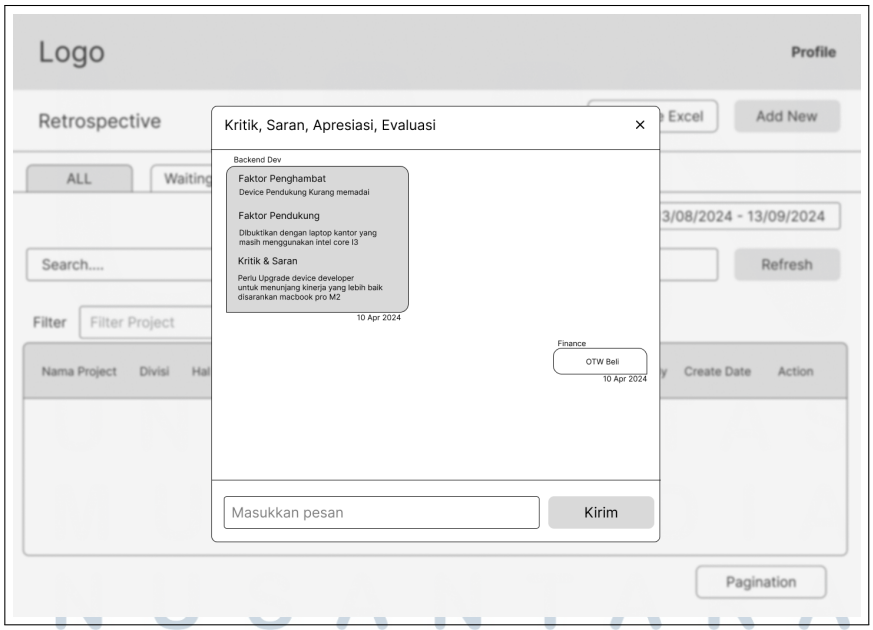
Gambar 3.6. Mockup halaman *dashboard* Retrospective

Gambar 3.7 merupakan mockup halaman tambah Retrospective. *Form* yang dikeluarkan melalui *pop up* memiliki beberapa *field* yang dapat diisi oleh pengguna.



Gambar 3.7. Mockup halaman tambah Retrospective

Gambar 3.8 merupakan mockup dari halaman detail Retrospective. Halaman ini berfungsi untuk menampilkan detail dan memberikan *feedback* dari Retrospective yang dibuat. Di halaman ini juga dilengkapi dengan *field* pesan yang digunakan untuk mengirim *feedback* antar pengguna.



Gambar 3.8. Mockup halaman detail Retrospective

C. Coding

Selama proses pengembangan fitur Retrospective, terdapat beberapa teknologi yang digunakan baik dari sisi *frontend* maupun *backend*, antara lain:

C.1 Frontend

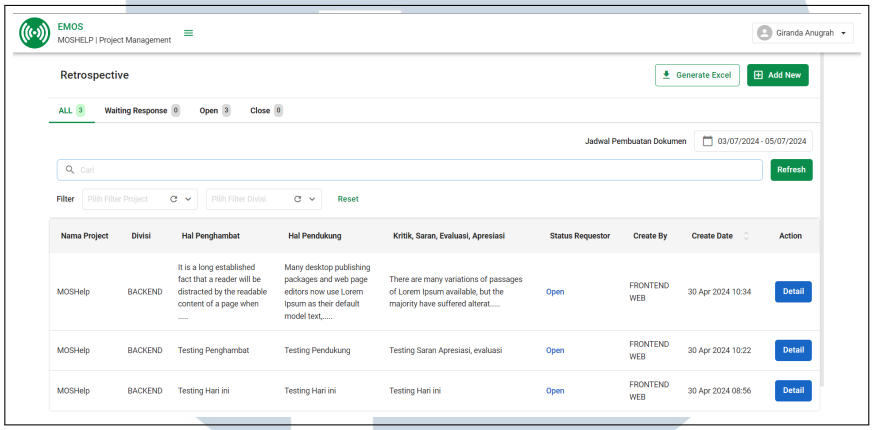
Dari sisi *frontend*, peserta magang menerapkan ReactJS (https://react.dev) sebagai framework utama dalam pengembangan website, yang dipilih karena kesesuaiannya dengan template website MOSHELP yang telah dimiliki oleh perusahaan. Hal ini memudahkan proses integrasi dan pengembangan, serta memastikan konsistensi dalam tampilan. Untuk mendukung antarmuka yang responsif, digunakan dua library CSS, yaitu Bootstrap (https://getbootstrap.com) dan Material UI (https://mui.com), yang keduanya menawarkan komponen UI modern dan fleksibel, sehingga mempercepat pengembangan tampilan yang menarik dan fungsional. Selain itu, dalam manajemen state aplikasi ReactJS, peserta magang menggunakan RecoilJS (https://recoiljs.org), yang disesuaikan dengan struktur dan teknologi template MOSHELP. RecoilJS memungkinkan pengelolaan state secara global dengan cara yang lebih sederhana. Dan untuk komunikasi antara *frontend* dan *backend*, peserta magang memanfaatkan Apollo GraphQL (https://www.apollographql.com) sebagai media penghubung API.

C.2 Backend

Di sisi *backend*, peserta magang tidak terlibat langsung dalam pengembangan API, namun peserta magang tetap memahami teknologi yang digunakan, yaitu Node.js (https://nodejs.org/en), PostgreSQL (https://www.postgresql.org/), dan Apollo GraphQL (https://www.apollographql.com). Pemilihan teknologi-teknologi ini disesuaikan dengan arsitektur dan template kode yang telah digunakan pada *backend* MOSHELP. Node.js dipilih karena keandalannya dalam menangani aplikasi berbasis JavaScript, sementara PostgreSQL menjadi pilihan sebagai database relasional yang kuat dan scalable. Apollo GraphQL digunakan untuk memastikan komunikasi API antara *frontend* dan *backend* berjalan secara efisien, memungkinkan query data yang lebih terarah dan hemat sumber daya. Teknologi-teknologi ini bekerja bersama-sama untuk mendukung performa dan skalabilitas aplikasi MOSHELP secara keseluruhan.

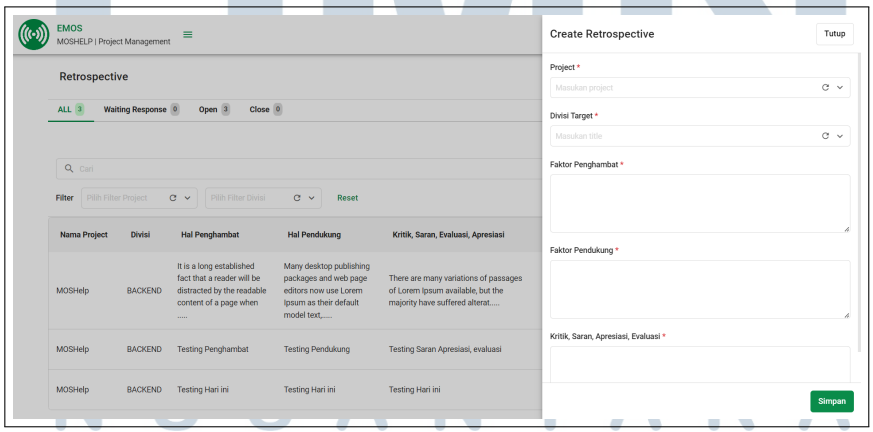
C.3 Hasil Implementasi

Gambar 3.9 merupakan tampilan *dashboard* dari fitur Retrospective. Halaman ini memiliki *data table* yang berfungsi untuk menampilkan *list* Retrospective. *Data table* juga dilengkapi dengan fitur *searching*, *filtering*, dan *tab pagination*. Di bagian kanan atas terdapat dua tombol untuk *generate excel* dan menambahkan Retrospective baru.



Gambar 3.9. Tampilan halaman *dashboard* Retrospective

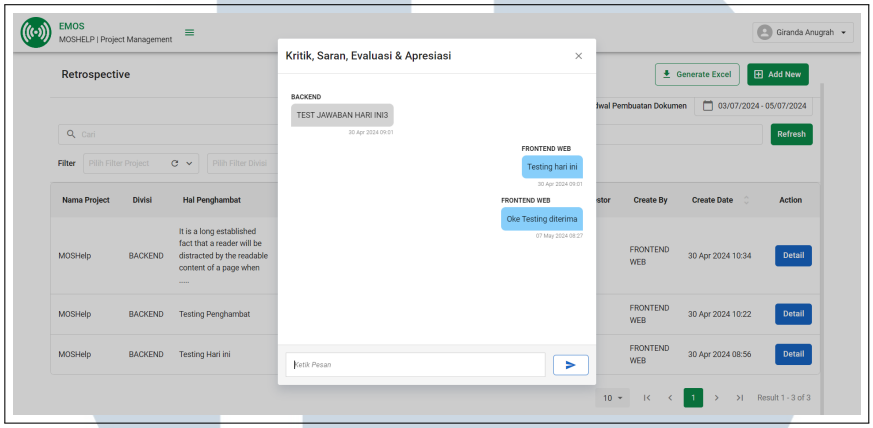
Gambar 3.10 merupakan tampilan tambah Retrospective. *Form* data Retrospective ditunjukkan melalui modal sidebar. Pada *form* ini terdapat beberapa *field* yang dapat diisi oleh pengguna, antara lain nama proyek, divisi target, divisi pengirim, faktor penghambat, faktor pendukung, dan kritik serta saran dari divisi pengirim.



Gambar 3.10. Tampilan halaman tambah Retrospective

Gambar 3.11 merupakan tampilan detail Retrospective. Tampilan detail dikeluarkan dalam bentuk *pop up* yang di dalamnya terdapat kritik dari divisi

pengirim serta *feedback* dari divisi penerima. Tampilan kritik dan *feedback* dibuat layaknya *chat bubble*, dan terdapat *field* pesan atau *feedback* yang dapat digunakan pengguna untuk saling menyampaikan kritik maupun *feedback* dari masing-masing divisi.



Gambar 3.11. Tampilan halaman detail dan *feedback* Retrospective

D. Testing

Setelah pengerjaan fitur Retrospective diselesaikan, dilakukan juga pengujian sistem menggunakan *black box testing* dan *white box testing* guna mengukur kesalahan dan kekurangan yang dimiliki oleh aplikasi sedini mungkin. *Black box testing* adalah metode pengujian sistem yang berfokus pada fungsionalitasnya tanpa mempertimbangkan struktur internal maupun kinerja aplikasi tersebut[5]. Sedangkan *white box testing* adalah metode pengujian aplikasi atau perangkat lunak dengan memanfaatkan modul untuk mengkaji dan menganalisis kode program yang telah dibuat[6].

Pengujian sistem ini dilakukan oleh peserta magang dan pembimbing lapangan, yang berkolaborasi dalam menilai efektivitas serta kestabilan setiap komponen dan fungsi aplikasi. Dengan demikian, hasil pengujian ini dapat memberikan umpan balik berharga untuk proses pemeliharaan dan pengembangan aplikasi ke depannya.

D.1 Black Box Testing

Dalam pengujian menggunakan *black box testing*, berfokus pada fungsionalitas aplikasi berdasarkan spesifikasi dan kebutuhan dari *software*. Tabel 3.3 menunjukkan hasil dari pengujian *black box testing*.

Tabel 3.2. Hasil pengujian *Black box* testing

No.	Fitur yang diuji	Skenario	Hasil yang diharapkan	Hasil pengujian
1	Melihat halaman *dashboard* Retrospective	Pengguna klik menu Retrospective pada *sidebar*	Pengguna berhasil masuk ke halaman *dashboard* Retrospective	Berhasil
2	Melihat *pop up* buat Retrospective	Pengguna klik tombol *Add New*	Pengguna berhasil menampilkan *Form new Retrospective*	Berhasil
3	Menambahkan Retrospective	Pengguna memasukkan data pada form Retrospective, klik tombol simpan	Pengguna dapat menambahkan Retrospective	Berhasil
4	Melihat *pop up detail* Retrospective	Pengguna klik tombol *detail*	Pengguna dapat menampilkan *pop up detail* Retrospective	Berhasil
5	Mengirimkan *feedback* Retrospective	Pengguna mengisi *field* kirim pesan, klik tombol kirim	Pengguna berhasil memberikan *feedback*	Berhasil
6	*Searching* data Retrospective	Pengguna memasukkan kata kunci pada *field searching* di halaman dashbord	Pengguna dapat melakukan s*earching* berdasarkan kata kunci	Berhasil
Lanjut di halaman berikutnya				

Tabel 3.3. Hasil pengujian *Black box* testing

No.	Fitur yang diuji	Skenario	Hasil yang diharapkan	Hasil pengujian
7	*Generate Excel*	Pengguna klik tombol *generate* di halaman dashboard	Pengguna dapat *generate* dan *download* fileexcel	Berhasil
8	*Filtering data Retrospective*	Pengguna memilih filter *dropdown* di halaman dashboard	Pengguna dapat melakukan filtering pada data Retrospective	Berhasil

D.2 White Box Testing

Pengujian menggunakan *whitebox testing* berfokus pada kegunaan fitur yang terdiri dari keberhasilan suatu module dan ketidak berhasilan sutu module. Pengujian white-box testing dilakukan dengan menyusun flowgraph berdasarkan flowchart yang telah dibuat sebelumnya. Selanjutnya, dilakukan perhitungan Cyclomatic Complexity untuk menentukan jumlah jalur independen yang terdapat pada flowgraph tersebut. Adapun rumus untuk menghitung Cyclomatic Complexity adalah sebagai berikut:

Cyclomatic Complexity:

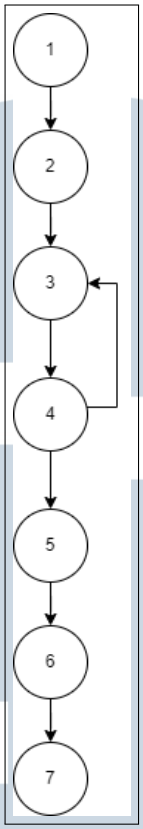
V(G) = E - N + 2

Keterangan:

V(G) : Jumlah jalur independen dalam flowgraph

E : Jumlah edge (anak panah) yang menggambarkan aliran kontrol

N : Jumlah node (simpul) yang merepresentasikan proses atau keputusan



Gambar 3.12. Flowgraph fitur tambah Retrospective

Gambar 3.12 merupakan *flowgraph* dari fitur tambah Retrospective. Berdasarkan *flowgraph* tersebut dapat dilakukan perhitungan menggunakan teknik basis *path* pada fitur tambah Retrospective:

Cyclomatic Complexity:
V(G) = E - N + 2
V(G) = 7 - 7 + 2
V(G) = 2

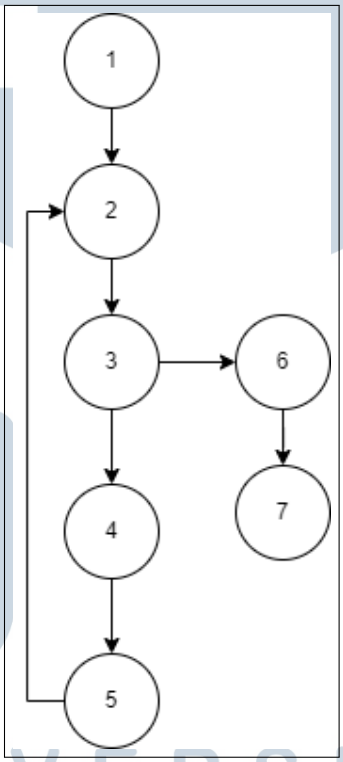
Berdasarkan hasil perhitungan tersebut didapatkan 2 jalur independen yaitu:

1. 1-2-3-4-5-6-7 (Skenario pengguna memasukkan data yang benar)
2. 1-2-3-4-3 (Skenario pengguna memasukkan data yang salah)

Setelah didapatkan jalur independen selanjutnya dilakukan *test case* dan membandingkan hasil yang didapat dengan hasil yang diharapkan. *Test case* tersebut dapat dilihat pada Tabel 3.4

Tabel 3.4. Hasil pengujian *test case* jalur independen

Path	Jalur	Skenario	Hasil pengujian
1	1-2-3-4-5-6-7	Pengguna memasukkan data yang benar, pengguna menekan tombol kirim, data berhasil dikirim dan kembali ke halaman *dashboard*	Berhasil
2	1-2-3-4-3	Pengguna memasukkan data yang salah, sistem menampikan pesan *error*, sistem melakukan *disable* pada tombol *submit*	Berhasil



Gambar 3.13. Flowgraph fitur *detail* Retrospective

Gambar 3.13 merupakan flow graph dari fitur detail Retrospective. Berdasarkan flowgraph tersebut dapat dilakukan perhitungan menggunakan teknik basis path pada fitur detail Retrospective:

Cyclomatic Complexity:

V(G) = E - N + 2

V(G) = 7 - 7 + 2

V(G) = 2

1. 1-2-3-4-5 (Skenario pengguna mengirimkan *feedback*)
2. 1-2-3-6-7 (Skenario pengguna keluar dari *pop up* detail)

Setelah didapatkan jalur independen selanjutnya dilakukan test case dan membandingkan hasil yang didapat dengan hasil yang diharapkan. Test case tersebut dapat dilihat pada Tabel 3.5

Tabel 3.5. Hasil pengujian *test case* jalur independen

Path	Jalur	Skenario	Hasil pengujian
1	1-2-3-4-5	Pengguna memasukkan *feedback*, pengguna menekan tombol kirim, data berhasil dikirim, data muncul pada halaman detail	Berhasil
2	1-2-3-6-7	Pengguna masuk ke *pop up* detail, pengguna menekan tombol keluar, pengguna kembali ke halaman *dashboard*	Berhasil

3.3.4 System Usability Scale

System Usability Scale (SUS) adalah metode penilaian *usability* secara *global* yang mencakup aspek efektivitas, efisiensi, dan kepuasan pengguna berdasarkan persepsi subjektif pribadi[7]. Pengujian menggunakan metode ini dilakukan dengan menyebarkan kusioner kepada 16 pengguna fitur Retrospective. Kusioner terdiri dari 10 pertanyaan yaitu:

1. Saya merasa fitur Retrospective bermanfaat.
2. Saya merasa fitur Retrospective rumit.
3. Saya merasa fitur Retrospective mudah digunakan.
4. Saya membutuhkan bantuan teknis untuk menggunakan fitur Retrospective.
5. Fungsi yang terdapat di dalam fitur Retrospective sudah sesuai dengan yang saya bayangkan.

6. Ada banyak ketidakkonsistenan dalam fitur Retrospective.
7. Saya merasa sebagian besar orang akan cepat memahami bagaimana cara menggunakan fitur Retrospective.
8. Saya merasa fitur Retrospective sangat membingungkan untuk digunakan.
9. Saya merasa percaya diri saat menggunakan fitur Retrospective.
10. Saya harus mempelajari banyak hal sebelum bisa menggunakan fitur Retrospective.

Perhitungan *System Usability Scale* (SUS) dilakukan dengan beberapa langkah. Pertama, untuk setiap pertanyaan ganjil, skor responden dikurangi dengan 1. Selanjutnya, untuk setiap pertanyaan genap, skor responden dikurangi 5. Setelah itu, hasil dari kedua langkah tersebut dijumlahkan. Total skor yang diperoleh kemudian dikalikan dengan 2,5 untuk mendapatkan nilai akhir SUS. Gambar 3.14 menampilkan hasil dari penyebaran kuesioner SUS beserta skor rata-rata yang diperoleh dari pengujian sistem.

Responden	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Jumlah	Jmlah*2,5
1	4	2	5	4	5	2	5	2	5	3	31	77,5
2	4	1	5	1	4	1	4	1	5	1	37	92,5
3	5	2	5	1	4	1	5	1	4	2	36	90
4	5	1	4	3	4	2	4	1	5	1	34	85
5	4	4	4	4	4	4	4	4	4	4	20	50
6	4	4	5	2	4	2	4	2	4	2	29	72,5
7	4	2	4	2	4	2	4	2	4	2	30	75
8	5	2	4	2	4	2	4	2	4	3	30	75
9	5	5	5	5	5	5	5	5	5	5	20	50
10	4	1	5	2	4	1	4	2	4	1	34	85
11	5	4	4	4	5	1	4	2	4	3	28	70
12	4	2	5	3	5	1	4	1	3	3	31	77,5
13	5	1	4	2	5	1	5	1	5	1	38	95
14	4	3	5	2	5	2	1	1	5	1	31	77,5
15	5	2	4	1	4	1	5	2	5	1	36	90
16	5	1	5	3	4	2	5	1	5	2	35	87,5
Rata-Rata												78,125

Gambar 3.14. Hasil Kusioner *System Usability Scale*

Skor rata-rata *System Usability Scale* (SUS) yang diperoleh adalah 78,125. Berdasarkan interpretasi standar SUS, skor ini berada di atas nilai rata-rata 68, yang menunjukkan bahwa sistem memiliki tingkat *usability* yang baik. Skor ini mengindikasikan bahwa sistem memenuhi kriteria efektivitas, efisiensi, dan kepuasan pengguna dengan cukup baik. Dengan nilai tersebut, sistem dapat dianggap unggul dalam mendukung pengalaman pengguna.

3.4 Kendala dan Solusi yang Ditemukan

Adapun kendala dan solusi yang ditemukan selama program magang adalah sebagai berikut:

3.4.1 Kendala

1. Keterlambatan dalam penyelesaian API dari tim *backend*, sehingga waktu pengerjaan *website* sedikit terhambat.
2. Project Moshelp masih menggunakan javascript sehingga sulit untuk pemeliharaan kode.
3. Kurangnya pemahaman terkait dengan penggunaan css *framework* bootsrap.

3.4.2 Solusi

1. Berkomunikasi secara teratur dengan tim *backend* untuk memahami perkembangan dan hambatan yang dihadapi.
2. Mempelajari struktur kode Moshelp yang masih menggunakan javascript secara menyeluruh sehingga mudah untuk pemeliharaan kode yang dibuat.
3. Manfaatkan dokumentasi resmi, video tutorial, serta sumber belajar lainnya untuk memahami css framework bootstrap.