BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama menjalani masa magang di PT Kalbe Farma Tbk, dan ditempatkan sebagai VR/XR Developer di bawah arahan langsung Bapak Miftahul Tirta Irawan selaku supervisor. Tanggung jawab utamanya adalah merancang dan membangun aplikasi Virtual Reality (VR) untuk mensimulasikan proses pemasangan dan pembongkaran mesin tablet pressing yang digunakan di pabrik Saka Farma. Sebelum masuk ke tahap pengembangan penuh, dilakukan eksplorasi terlebih dahulu terhadap kebutuhan teknis dan alur kerja mesin melalui studi pustaka, diskusi dengan tim, serta observasi lapangan. Dari hasil eksplorasi tersebut, dibuatlah *Proof of Concept* (PoC) sebagai validasi awal terhadap pendekatan teknis yang akan digunakan. Pengembangan selanjutnya dilakukan secara kolaboratif dalam tim VR, yang terdiri dari dua orang intern.

Dalam pelaksanaannya, proyek ini melibatkan koordinasi lintas divisi yang cukup intensif. Tim VR bertanggung jawab penuh terhadap proses pengembangan aplikasi mulai dari perancangan, implementasi, hingga pengujian. Tim 3D Model bertugas membuat aset-aset mesin dalam bentuk 3D yang nantinya diintegrasikan ke dalam lingkungan VR. Tim Project Management (PM) banyak berperan dalam merancang alur simulasi (*flow*) berdasarkan masukan dari pihak Saka Farma, serta mengoordinasikan kebutuhan teknis dan fungsional antar tim.

Koordinasi antar tim dilakukan secara daring melalui platform komunikasi seperti WhatsApp dan Microsoft Teams, serta menggunakan GitHub, SharePoint, Asana, dan Notion untuk dokumentasi dan pelacakan progres proyek. Selain itu, kami juga menggunakan Draw.io untuk visualisasi *flow* dan Unity sebagai platform utama pengembangan aplikasi VR. Setiap minggu, tim mengadakan sesi meeting untuk melaporkan progres, mendiskusikan kendala teknis, serta menentukan langkah tindak lanjut. Selain itu, terlibat langsung juga dalam sesi pengujian aplikasi ke pihak Saka Farma serta kegiatan genba (observasi langsung) ke pabrik Saka Farma untuk mempelajari proses *Clean-Up Set-Up* (CUSU) mesin yang disimulasikan.

Di luar tanggung jawab utama pengembangan aplikasi, juga turut membantu tim PM dalam merancang flow aplikasi VR agar sesuai dengan urutan proses

kerja mesin di dunia nyata. Selain itu, tim VR juga menyusun sebuah *internal framework* berupa panduan umum dan alur kerja standar yang dapat digunakan oleh tim lain dalam mengembangkan proyek VR serupa di masa mendatang. *Framework* ini bertujuan untuk meningkatkan efisiensi dan konsistensi dalam pengembangan aplikasi *immersive* di lingkungan perusahaan.

3.2 Tugas yang Dilakukan

Selama pelaksanaan magang di PT Kalbe Farma Tbk sebagai VR/XR Developer, beberapa tugas utama yang dilakukan antara lain sebagai berikut.

1. Merancang dan mengembangkan aplikasi Virtual Reality (VR) menggunakan Unity untuk simulasi perakitan (*assembly*) dan pembongkaran (*disassembly*) mesin industri.
2. Mengimplementasikan fitur interaksi pengguna berbasis XR Interaction Toolkit dan Unity Input System untuk perangkat Meta Quest 3.
3. Membuat *Proof of Concept* (PoC) sebagai purwarupa awal untuk validasi alur interaksi dan performa sistem.
4. Melakukan integrasi aset 3D ke dalam lingkungan Unity serta pengaturan *collider*, *rigidbody*, dan *socket*.
5. Melaksanakan observasi lapangan (*genba*) untuk memahami prosedur kerja aktual dan merancang alur simulasi yang sesuai.
6. Menyusun alur simulasi dalam bentuk diagram menggunakan Draw.io untuk mendukung proses pengembangan dan validasi.
7. Melakukan pengujian aplikasi melalui demo internal dan evaluasi bersama tim Project Management (PM) dan klien.
8. Merumuskan framework pengembangan proyek VR untuk mendukung pengembangan berkelanjutan dan dokumentasi proses.

Proyek utama yang dikerjakan berfokus pada pembuatan aplikasi VR yang mensimulasikan proses perakitan (assembly) dan pembongkaran (disassembly) komponen mesin, yang digunakan sebagai media pelatihan bagi operator di lingkungan industri. Pengembangan dilakukan menggunakan Unity sebagai *game*

engine utama, dengan memanfaatkan fitur-fitur seperti XR Interaction Toolkit dan Unity Input System untuk membangun interaksi pengguna secara imersif melalui perangkat *Head-Mounted Display* (HMD) Meta Quest 3. Tugas mencakup pembuatan skrip interaksi berbasis bahasa pemrograman C#, konfigurasi *collider* dan *rigidbody* untuk interaksi fisik, serta integrasi aset 3D dari tim desainer ke dalam lingkungan virtual.

Salah satu tahapan penting dalam proses pengembangan adalah pembuatan *Proof of Concept* (PoC) sebagai bentuk purwarupa awal dari sistem yang akan dikembangkan. PoC ini bertujuan untuk memvalidasi konsep interaksi, alur simulasi, serta performa sistem secara teknis sebelum masuk ke tahap produksi penuh. Pada fase ini, dikembangkan fitur-fitur dasar seperti *grab* dan *release* objek, interaksi tombol, serta sistem validasi urutan perakitan yang sederhana. Pembuatan PoC memungkinkan tim untuk mengidentifikasi potensi kendala sejak awal dan memberikan gambaran konkret kepada stakeholder mengenai arah pengembangan proyek. Selain itu, PoC juga berperan sebagai bahan presentasi awal kepada klien untuk mendapatkan umpan balik sebelum aplikasi dilanjutkan ke tahap simulasi lengkap.

Untuk mendukung ketepatan desain simulasi, kegiatan *genba* atau observasi langsung ke lapangan juga dilakukan sebagai bagian integral dari proses pengembangan. *Genba* memungkinkan tim pengembang memahami secara menyeluruh proses kerja nyata yang akan disimulasikan, termasuk posisi operator, urutan kerja, peralatan yang digunakan, serta potensi kesalahan yang umum terjadi. Dari hasil observasi ini, alur simulasi kemudian dirancang agar sesuai dengan kondisi aktual dan tetap ergonomis ketika dijalankan dalam lingkungan VR. Selain itu, *genba* juga menjadi wadah untuk berkoordinasi langsung dengan teknisi atau operator berpengalaman yang memberikan masukan praktis mengenai langkah kerja yang penting untuk direpresentasikan secara akurat.

Selain pengembangan teknis, proses penyusunan alur simulasi juga menjadi bagian penting dari tanggung jawab selama magang. Bersama tim Project Management (PM), dilakukan diskusi mendalam untuk memahami kebutuhan pengguna dan menyusun skenario simulasi yang sesuai dengan prosedur nyata di lapangan. *Flow* simulasi ini divisualisasikan dalam bentuk diagram alur menggunakan Draw.io, yang memuat urutan interaksi, kondisi benar/salah, serta transisi antar tahap. Seluruh alur kemudian dijadikan acuan dalam proses pengembangan dan validasi oleh tim internal maupun pihak klien.

Tahapan pengujian juga menjadi bagian penting dari rangkaian kegiatan

magang, saat ini karena masih dalam tahap pengembangan dan belum memungkinkan untuk di *build* ke perangkat VR, maka testing dilakukan dengan meeting dalam bentuk presentasi atau demo. Namun jika sudah memungkinkan untuk di *build* ke aplikasi VR, maka akan dilakukan proses testing aplikasi secara langsung menggunakan perangkat HMD untuk memastikan seluruh fungsi berjalan sesuai skenario. Kegiatan uji coba tidak hanya dilakukan di lingkungan pengembangan, tetapi juga secara langsung di lokasi pengguna akhir (genba) bersama tim PM dan pihak klien. Observasi ini memberikan masukan berharga untuk menyempurnakan sistem, termasuk dari segi kenyamanan pengguna, ketepatan prosedur, hingga efektivitas skenario pelatihan.

Sebagai upaya mendukung keberlanjutan pengembangan proyek serupa di masa depan, dilakukan pula penyusunan framework atau kerangka acuan proyek VR. Framework ini mencakup tahapan mulai dari perencanaan awal, alur pengembangan, manajemen aset, hingga proses testing dan dokumentasi. Tujuannya adalah untuk memberikan pedoman yang jelas bagi tim pengembang lainnya, agar proyek-proyek VR ke depannya dapat dilakukan secara lebih terstruktur dan efisien.

Secara keseluruhan, tugas-tugas yang dilaksanakan mencerminkan keterlibatan dalam seluruh siklus pengembangan aplikasi VR, mulai dari perencanaan, produksi, implementasi, evaluasi, hingga perumusan dokumentasi pendukung. Kegiatan ini tidak hanya memperluas pemahaman teknis dalam pengembangan XR, tetapi juga memperkuat keterampilan kolaborasi, dokumentasi, dan adaptasi terhadap kebutuhan industri yang dinamis.

3.3 Uraian Pelaksanaan Magang

Tabel 3.1 menyajikan rincian lengkap mengenai linimasa kegiatan yang telah dilaksanakan selama masa magang di PT Kalbe Farma Tbk. Tabel ini memuat aktivitas mingguan yang mencerminkan progres bertahap dari setiap tahapan pengembangan aplikasi Virtual Reality (VR), mulai dari tahap perencanaan awal, pembuatan *Proof of Concept* (PoC), integrasi aset dan interaksi, hingga proses pengujian dan dokumentasi. Dengan adanya linimasa ini, pembaca dapat memperoleh gambaran menyeluruh mengenai alur kerja dan kontribusi yang diberikan selama program magang berlangsung, serta bagaimana setiap kegiatan saling terintegrasi dalam mendukung pencapaian tujuan proyek secara keseluruhan.

Tabel 3.1. Pekerjaan yang dilaksanakan setiap minggu

Minggu Ke -	Pekerjaan yang dilakukan
1	Kegiatan yang dilakukan antara lain meliputi pengenalan divisi dan struktur perusahaan, penjelasan umum mengenai proyek, sesi perkenalan tim dan onboarding, brainstorming ide pengembangan model VR, serta pengaturan awal perangkat kerja VR.
2	Kegiatan berfokus pada pembuatan awal model 3D kabinet secara berulang dan pengaturan VR toolkit, serta diakhiri dengan rapat tim untuk membahas detail proyek lebih lanjut.
3	Dilakukan pembaruan dokumentasi pada Notion, pengembangan model perakitan dan pembongkaran meja, serta eksplorasi awal terhadap alur kerja perakitan dan pembongkaran.
4	Kegiatan difokuskan pada eksplorasi lebih lanjut terhadap proses perakitan dan pembongkaran secara berulang, disertai dengan rapat koordinasi tim.
5	Kegiatan masih berfokus pada eksplorasi praktik perakitan dan pembongkaran, dan ditutup dengan observasi lapangan (genba) ke Saka Farma Laboratories.
6	Dilakukan finalisasi eksplorasi sebelumnya dan dimulainya proyek Cabinet Assemble and Disassemble secara lebih terstruktur.
7	Pengembangan proyek cabinet dilanjutkan dengan fokus pada penghubungan antar komponen dan penyusunan urutan logika assembly.
8	Kegiatan meliputi dokumentasi proses, serta pengembangan dan pengujian interaksi pada objek hammer dan nail dalam simulasi perakitan cabinet.
9	Dilakukan perancangan dan pengujian antarmuka pengguna (UI), termasuk menu, highlight objek, dan konfigurasi kontroler.
10	Dilanjutkan integrasi antar bagian dan interaksi dua tangan (2-hand grab), serta koneksi proyek ke repositori GitHub dan dokumentasi pendukung.
11	Kegiatan mencakup perbaikan bug berdasarkan hasil pengujian POC serta pemenuhan skenario test case dan integrasi efek suara (SFX).
	Lanjut pada halaman berikutnya

Tabel 3.1. Pekerjaan yang dilaksanakan setiap minggu (Lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
12	Pengujian test case dilanjutkan dan dokumentasi disempurnakan, disertai diskusi bersama tim terkait kendala dan perbaikan.
13	Dilakukan penyusunan ulang kode (refactor), pengaturan ulang perangkat VR, serta pengujian terhadap pembongkaran objek cabinet.
14	Dilakukan bug fixing pada sistem socket dan placeholder, serta rapat kickoff untuk proyek lanjutan.
15	Kegiatan meliputi penyempurnaan sistem perakitan, pengujian akhir, serta perancangan awal skenario pembongkaran.
16	Dikembangkan skenario pembongkaran termasuk menu utama, proses melepaskan nail, dan pembuatan flowchart kerja mesin.
17	Dimulai proyek baru untuk simulasi mesin Fill-O-Matic, mencakup tahap awal perakitan dan pembongkaran serta dokumentasinya.
18	Dilakukan pembongkaran menyeluruh pada mesin Fill-O-Matic dan pengujian untuk semua komponen bagian.
19	Kegiatan ditujukan untuk menyempurnakan dan mendokumentasikan semua bagian dari pembongkaran mesin Fill-O-Matic.

Kegiatan yang tercantum dalam Tabel 3.1 menggambarkan bahwa pelaksanaan magang berjalan secara bertahap dan sistematis, dengan fokus utama pada pengembangan aplikasi VR yang fungsional dan sesuai kebutuhan industri. Setiap minggu diisi dengan aktivitas yang saling mendukung, mulai dari observasi langsung hingga implementasi teknis. Hal ini menunjukkan keterlibatan aktif peserta magang dalam setiap proses pengembangan, sekaligus mencerminkan pendekatan kerja kolaboratif dan iteratif yang diterapkan dalam proyek simulasi ini.

3.4 Perangkat Penunjang Pelaksanaan Magang

Dalam melaksanakan kegiatan magang, dibutuhkan berbagai perangkat lunak (software), perangkat keras (hardware), serta kerangka kerja (framework) dan pustaka (library) yang masing-masing memiliki peran penting dalam mendukung

proses pengembangan VR. Uraian mengenai perangkat-perangkat tersebut akan dijelaskan lebih lanjut dalam beberapa subbab berikut.

3.4.1 Perangkat Lunak yang Digunakan

Selama proses magang, berbagai perangkat lunak digunakan untuk menunjang kegiatan pengembangan aplikasi Virtual Reality serta koordinasi tim. Perangkat lunak utama yang digunakan antara lain:

Tabel 3.2. Daftar Perangkat Lunak dan spesifikasinya (versi)

Nama Perangkat Lunak	Spesifikasi (Versi)
Unity	Digunakan sebagai game engine utama untuk membangun simulasi VR. Mendukung fitur seperti Physics Interaction, XR Interaction Toolkit, dan integrasi 3D model. Versi 2022.3.43f1 LTS
Side Quest	Digunakan untuk menginstall aplikasi VR ke dalam perangkat Meta Quest 3
Draw.io	Digunakan untuk membuat diagram alur proses simulasi, termasuk urutan interaksi pengguna dan transisi tahap simulasi.
Notion	Digunakan untuk dokumentasi proyek: ide, catatan meeting, tugas mingguan, dan referensi teknis.
SharePoint	Digunakan untuk penyimpanan dokumen dan aset proyek secara terorganisir dan kolaboratif.
GitHub	Digunakan sebagai sistem version control dan kolaborasi kode melalui branch, pull request, dan review.
Microsoft Teams	Digunakan untuk komunikasi tim dan rapat daring bersama klien dari Saka Farma.
WhatsApp	Digunakan untuk komunikasi informal dan koordinasi cepat antar anggota tim.
Asana	Digunakan untuk task tracking.

3.4.2 Perangkat Keras yang Digunakan

Dalam menunjang kegiatan pengembangan aplikasi Virtual Reality, perangkat keras yang digunakan harus mendukung kebutuhan rendering 3D serta kompatibilitas dengan platform XR. Perangkat keras utama yang digunakan antara lain:

Tabel 3.3. Daftar perangkat keras dan spesifikasinya

Nama Perangkat	Spesifikasi
Laptop Pengembangan	Digunakan untuk menjalankan Unity Editor, melakukan coding, serta mengelola aset proyek. Spesifikasi utama: Intel Core i3, RAM 12 GB. Cukup mumpuni untuk kebutuhan pengembangan dan testing VR dasar.
PC Pengembangan	Digunakan untuk menjalankan Unity Editor, melakukan coding, serta mengelola aset proyek. Spesifikasi utama: Intel Core i7, RAM 12 GB. Cocok untuk pengembangan dan pengujian VR yang lebih berat.
Head-Mounted Display (HMD) dan Controller	Meta Quest 3. Digunakan untuk menguji aplikasi VR secara real-time. Memungkinkan pengguna berinteraksi secara imersif dengan lingkungan virtual.

3.4.3 Kerangka Kerja dan Library yang Digunakan

Pengembangan aplikasi Virtual Reality selama masa magang menggunakan berbagai framework dan library untuk mendukung proses pembangunan sistem yang modular dan interaktif. Beberapa kerangka kerja dan pustaka utama yang digunakan antara lain:

Tabel 3.4. Daftar framework / library dan spesifikasinya (versi)

Nama Framework / Library	Spesifikasi (Versi)
XR Interaction Toolkit	3.0.8
Unity Input System	1.14.0

Lanjut pada halaman berikutnya

Tabel 3.4. Daftar framework / library dan spesifikasinya (versi) (Lanjutan)

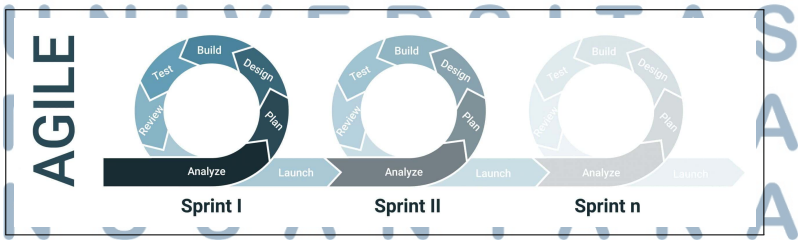
Nama Framework / Library	Spesifikasi (Versi)
Mock HMD (XR Simulator)	1.2.0-preview.1
GitHub Desktop	3.4.19
Draw.io (Desktop)	27.0.5
Unity Asset Store Package	1.6.250208

3.5 Proses Pelaksanaan Magang

Dalam pelaksanaan proyek pengembangan aplikasi *Virtual Reality* (VR) untuk pelatihan operator mesin di PT Kalbe Farma Tbk, digunakan pendekatan pengembangan perangkat lunak yang menggabungkan metodologi *Agile* dengan kerangka kerja perbaikan berkelanjutan *PDCA (Plan-Do-Check-Act)*. Pemilihan pendekatan ini didasarkan pada kebutuhan akan fleksibilitas, kecepatan adaptasi, dan efektivitas kolaborasi lintas tim dalam merespons dinamika kebutuhan industri.

3.5.1 Metodologi Agile

Agile merupakan pendekatan pengembangan perangkat lunak yang menekankan pada kolaborasi, fleksibilitas terhadap perubahan [15][16][17][18][19][20], serta pengiriman produk yang dapat diujicobakan secara iteratif. *Agile* dipilih karena sangat sesuai dengan karakteristik proyek ini, di mana pengembangan aplikasi dilakukan secara bertahap dan melibatkan umpan balik rutin dari stakeholder untuk memastikan kesesuaian simulasi dengan prosedur nyata di lapangan [21][22][23].



Gambar 3.1. Ilustrasi metode Agile dalam pengembangan proyek.

Proyek dibagi ke dalam beberapa *sprint* atau siklus pengembangan singkat

berdurasi 1–2 minggu. Setiap sprint akan menghasilkan feedback terhadap pengerjaan saat ini dan digunakan sebagai acuan untuk sprint berikutnya.

Penggunaan Agile memberikan sejumlah keuntungan berikut:

1. Adaptif terhadap perubahan kebutuhan user selama proses berlangsung.
2. Mengelola risiko teknis dengan membangun dan menguji sistem secara bertahap.
3. Melibatkan stakeholder secara aktif dalam siklus pengambilan keputusan.
4. Efisiensi kerja lintas divisi karena tugas dibagi modular per fitur.

3.5.2 Kerangka PDCA (Plan-Do-Check-Act)

Selain *Agile*, proyek juga menerapkan prinsip manajemen berkelanjutan PDCA, yang mendukung pengembangan sistematis dan terstruktur. PDCA digunakan untuk mengevaluasi dan menyempurnakan proses kerja dalam setiap fase.

Gambar 3.2. Siklus PDCA dalam pengembangan sistem.

Berikut adalah penerapan PDCA dalam konteks proyek ini:

1. Plan (Perencanaan): Analisis kebutuhan, penyusunan flowchart, perencanaan fitur, dan pembagian sprint.
2. Do (Pelaksanaan): Implementasi sistem menggunakan Unity, termasuk scripting, setup interaksi, dan integrasi aset.

3. Check (Pemeriksaan): Uji coba aplikasi VR menggunakan HMD, verifikasi alur kerja, dan pengumpulan umpan balik.

4. Act (Tindakan): Perbaikan sistem berdasarkan hasil pengujian dan masukan dari pengguna akhir.

3.5.3 Alasan Penggunaan Agile dan PDCA

Penggabungan Agile dan PDCA memberikan sinergi optimal antara fleksibilitas pengembangan dan pengendalian mutu. Alasannya antara lain:

1. Kompleksitas tinggi membutuhkan pendekatan iteratif dan adaptif.

2. Kebutuhan pengguna berkembang seiring berjalannya proyek.

3. Feedback pengguna harus segera ditindaklanjuti untuk meningkatkan efektivitas pelatihan.

4. Kolaborasi antardivisi memerlukan struktur kerja yang jelas dan responsif.

Pendekatan ini terbukti efektif dalam mendukung efisiensi kerja tim, mempercepat pengembangan fitur, serta menjaga kualitas akhir aplikasi sesuai dengan kebutuhan pelatihan industri.

3.5.4 Tahap 1: Analisis Kebutuhan

Analisis kebutuhan merupakan tahap awal yang sangat krusial dalam proses pengembangan aplikasi *Virtual Reality* (VR) untuk keperluan pelatihan operator mesin. Tahap ini bertujuan untuk memperoleh pemahaman yang komprehensif mengenai kebutuhan pengguna akhir, konteks lingkungan kerja nyata, serta proses bisnis yang hendak disimulasikan dalam aplikasi VR. Dalam konteks magang ini, analisis kebutuhan dilakukan dengan pendekatan sistematis dan kolaboratif bersama tim Project Management (PM), klien dari Saka Farma, serta divisi-divisi teknis terkait di Kalbe Farma.

A Tahap 1.1: Memahami Kebutuhan User/Klien

Langkah pertama dalam analisis kebutuhan adalah memahami secara menyeluruh kebutuhan pengguna dan ekspektasi dari pihak klien. Kegiatan ini

dilakukan melalui beberapa sesi diskusi bersama tim Project Management dan perwakilan dari pihak Saka Farma sebagai klien utama. Diskusi dilakukan secara daring maupun luring untuk memperoleh kejelasan terkait tujuan utama pengembangan aplikasi VR, target pengguna (yaitu operator mesin), serta prosedur kerja yang menjadi fokus pelatihan.

Hasil dari sesi ini menunjukkan bahwa kebutuhan utama adalah adanya media pelatihan interaktif yang dapat menggantikan metode konvensional yang selama ini mengandalkan mesin fisik di area produksi. Klien mengharapkan aplikasi VR dapat mensimulasikan proses perakitan dan pembongkaran mesin secara realistis, lengkap dengan validasi langkah kerja serta antarmuka yang intuitif bagi pengguna. Selain itu, pelatihan juga diharapkan dapat meningkatkan efisiensi, mengurangi risiko kerusakan mesin akibat pelatihan langsung, dan memberikan fleksibilitas waktu belajar bagi operator baru.

B Tahap 1.2: Pelaksanaan Genba / Survey Lokasi

Setelah memperoleh kebutuhan fungsional dan non-fungsional dari sisi klien, dilakukan kegiatan *genba* atau observasi langsung ke area kerja (lapangan) di pabrik Saka Farma. Tujuan dari genba ini adalah untuk memahami kondisi nyata operasional mesin, termasuk tata letak fisik, urutan kerja, alat yang digunakan, serta potensi tantangan yang dihadapi oleh operator dalam proses perakitan dan pembongkaran mesin.

Selama proses observasi, tim pengembang melakukan dokumentasi visual dan pencatatan rinci terhadap seluruh prosedur teknis yang dijalankan oleh operator berpengalaman. Tim juga berinteraksi langsung dengan teknisi dan operator senior guna memperoleh insight praktis terkait langkah-langkah kritikal, waktu pengerjaan rata-rata, serta kesalahan umum yang kerap terjadi di lapangan.

Hasil dari kegiatan genba ini menjadi fondasi penting dalam merancang alur simulasi yang realistis dan ergonomis dalam lingkungan VR. Setiap pergerakan, posisi, dan urutan kerja operator disesuaikan agar bisa diadaptasi secara optimal ke dalam antarmuka dan mekanisme interaksi VR.

C Tahap 1.3: Pembuatan dan Pemahaman Workflow (Flowchart)

Langkah krusial dalam proses perancangan sistem adalah penyusunan alur kerja (workflow) berdasarkan hasil wawancara dengan *stakeholder* dan observasi

langsung (genba) di area produksi. *Workflow* ini divisualisasikan dalam bentuk diagram alur (flowchart) yang merepresentasikan urutan prosedural kerja operator dalam menjalankan proses perakitan (assembly) dan pembongkaran (disassembly) komponen mesin.

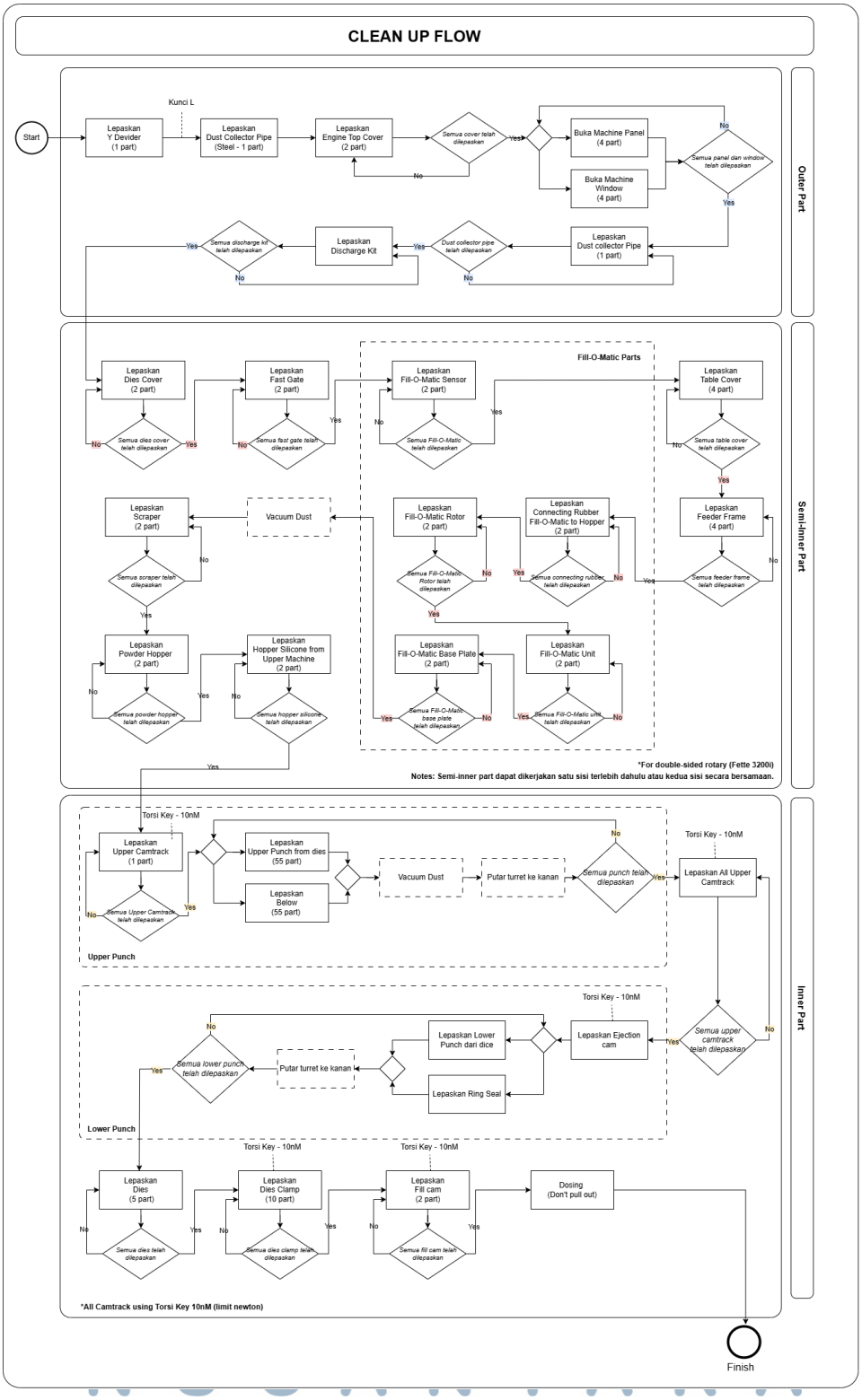
Diagram alur dirancang untuk mencakup:

1. Urutan langkah-langkah kerja dari awal hingga akhir.
2. Percabangan logika seperti kondisi benar/salah.
3. Pengulangan langkah akibat kesalahan prosedural.

Dalam proses pengembangan simulasi pelatihan berbasis Virtual Reality (VR), diperlukan pemahaman yang menyeluruh terhadap prosedur kerja nyata yang dijalankan oleh operator di lapangan. Simulasi yang baik tidak hanya dibangun berdasarkan asumsi alur kerja, tetapi harus didasarkan pada proses yang benar-benar terjadi di dunia nyata. Oleh karena itu, dilakukan observasi langsung atau genba terhadap aktivitas pemasangan (assembly) dan pembongkaran (disassembly) komponen mesin di area produksi. Observasi ini dilakukan secara sistematis untuk menangkap setiap langkah teknis, urutan pengerjaan, serta alat dan komponen yang digunakan oleh operator selama proses berlangsung.

Dari hasil observasi tersebut, disusunlah dua buah diagram alur (flowchart) yang merepresentasikan tahapan-tahapan kerja secara logis dan berurutan, sebagaimana dilakukan oleh operator pada kondisi aktual. Flowchart ini bukan merupakan alur pengembangan aplikasi VR, melainkan dokumentasi visual dari prosedur kerja nyata yang kemudian dijadikan referensi dalam merancang skenario simulasi interaktif

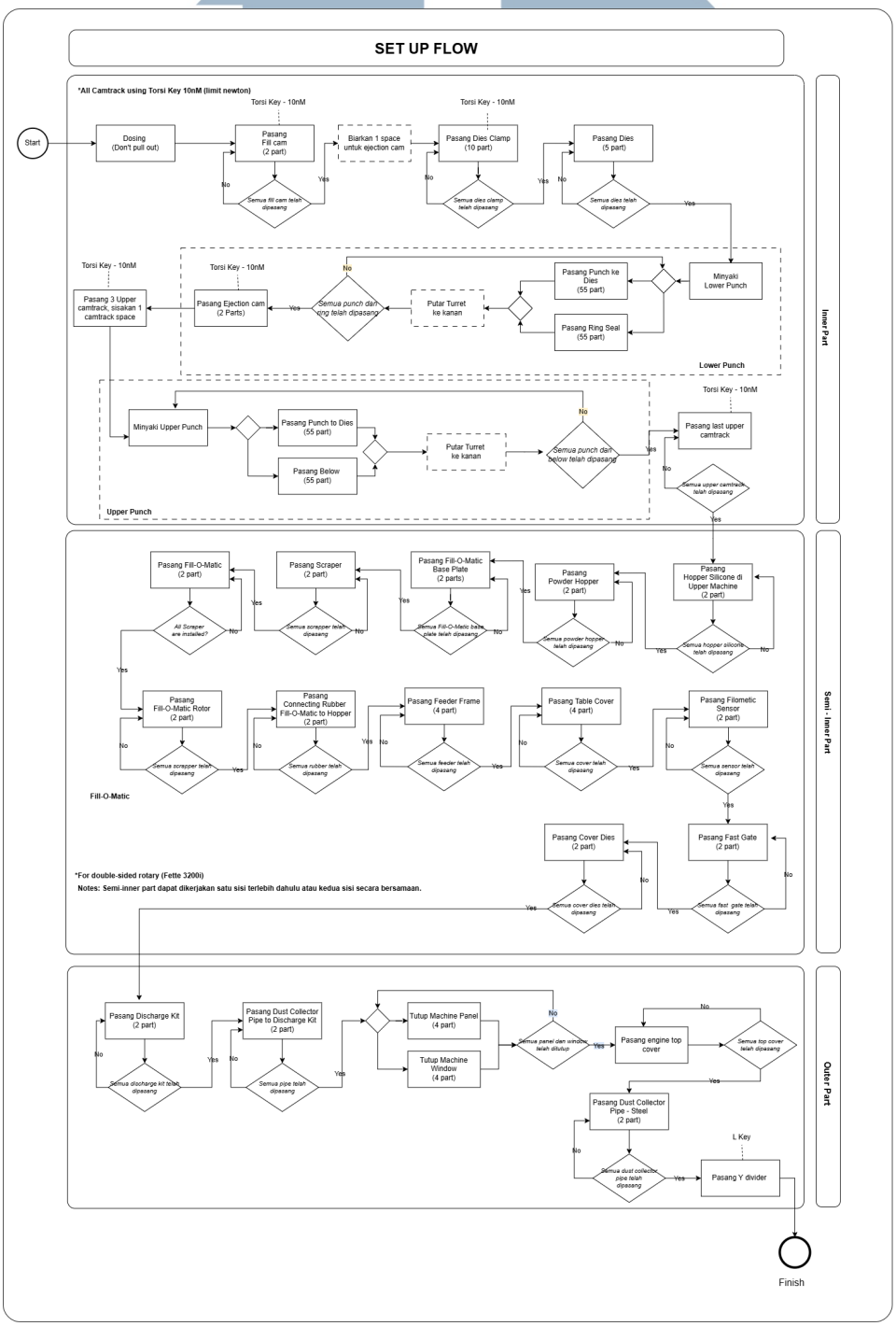
Kedua flowchart yang telah disusun tersebut menjadi referensi krusial dalam tahap implementasi sistem simulasi VR, baik dari sisi urutan kerja secara logis maupun penggambaran interaksi antar objek secara interaktif. Dengan mengacu pada hasil dokumentasi ini, aplikasi simulasi yang dikembangkan tidak hanya mampu meniru kondisi kerja secara visual, tetapi juga memberikan pengalaman pelatihan yang efektif, aman, dan selaras dengan standar operasional yang berlaku di industri farmasi.



Gambar 3.3. Diagram alur prosedur pembongkaran (*disassembly*) mesin berdasarkan observasi lapangan.

Gambar 3.3 memperlihatkan urutan proses pembongkaran mesin, mulai dari

pelepasan bagian luar seperti *Y Divider* dan *Cover Panel*, hingga pelepasan bagian dalam seperti *Camtrack*, *Punch*, dan *Dies*. Prosedur ini mencakup langkah-langkah pembersihan seperti *Vacuum Dust*, serta penggunaan alat bantu untuk memastikan pelepasan komponen dilakukan sesuai standar.



Gambar 3.4. Diagram alur prosedur pemasangan (*assembly*) mesin berdasarkan observasi lapangan.

Gambar 3.4 menyajikan tahapan pemasangan komponen mesin secara menyeluruh, dimulai dari proses dasar seperti pemasangan *Dosing* dan *Dies*, hingga tahapan akhir seperti pemasangan *Feeder Frame*, *Cover Dies*, dan *Fat Gate*. Setiap langkah dalam diagram ini merepresentasikan urutan aktual yang dilaksanakan operator, termasuk penggunaan alat ukur torsi untuk memastikan kekencangan yang tepat.

Penyusunan flowchart dilakukan secara kolaboratif antara tim pengembang, tim Project Management, serta pihak teknis dari lapangan. Hal ini bertujuan agar diagram alur yang dihasilkan dapat mencerminkan proses aktual secara presisi dan mengakomodasi kompleksitas sistem yang disimulasikan. Keterlibatan tim lintas fungsi juga memastikan bahwa *flowchart* dapat dimengerti baik oleh pihak teknis maupun non-teknis, serta meminimalkan potensi miskomunikasi selama proses implementasi.

Flowchart ini tidak hanya menjadi dokumen visualisasi proses, tetapi juga berfungsi sebagai referensi utama bagi tim pengembang dalam menyusun *Proof of Concept (PoC)* dan membangun struktur logika interaktif dalam Unity. Setiap elemen dalam flowchart dikaji ulang untuk memastikan kelayakan teknis, efisiensi interaksi, dan kesesuaian ergonomis dalam lingkungan Virtual Reality.

Setelah disusun, flowchart divalidasi secara internal oleh tim dan diserahkan kepada pihak klien untuk mendapatkan umpan balik. Validasi ini dilakukan agar seluruh langkah dan urutan interaksi dalam simulasi nantinya sesuai dengan prosedur operasi standar (SOP) yang berlaku di lapangan.

Flowchart ini menjadi fondasi penting dalam pembangunan sistem karena menjamin konsistensi antara kebutuhan awal dan aplikasi akhir yang dihasilkan, serta mempermudah proses debugging dan pengujian selama tahap pengembangan dan implementasi.

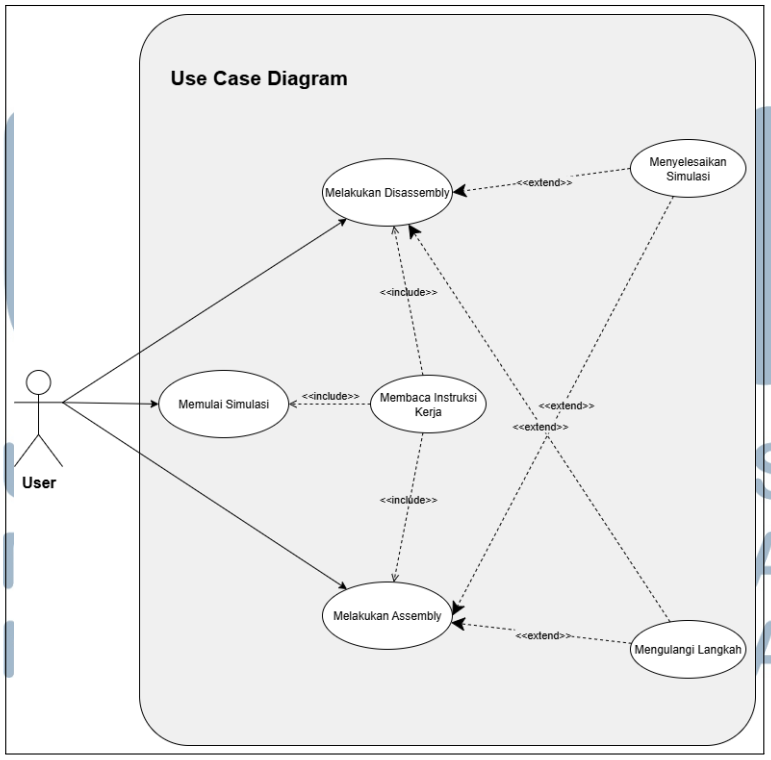
3.5.5 Tahap 2: Perancangan Sistem

Tahap perancangan sistem merupakan tahap transisi antara analisis kebutuhan dan implementasi teknis. Pada tahap ini, alur interaksi pengguna dan struktur logika sistem divisualisasikan dalam bentuk diagram dan skenario interaktif, untuk memastikan bahwa pengembangan akan berjalan sesuai dengan kebutuhan pengguna akhir.

A Tahap 2.1: UML Diagram

Dalam proses perancangan sistem, penggunaan *Unified Modeling Language* (UML) menjadi pendekatan yang efektif untuk memvisualisasikan struktur dan perilaku sistem secara sistematis. UML menyediakan berbagai jenis diagram yang membantu pengembang, desainer, dan stakeholder dalam memahami serta mendokumentasikan kebutuhan dan alur kerja sistem. Pada pengembangan aplikasi simulasi perakitan dan pembongkaran berbasis VR ini, dibutuhkan pemodelan visual yang mampu menjelaskan interaksi kompleks antara pengguna dan sistem secara jelas dan terstruktur.

Sebagai langkah awal, disusunlah *Use Case Diagram* untuk mengidentifikasi aktor utama serta kasus penggunaan (use case) yang tersedia dalam sistem. Diagram ini dipilih karena mampu memberikan gambaran makro mengenai fungsi-fungsi utama yang dapat diakses oleh pengguna, serta hubungan antara aktor dan sistem secara konseptual. Use case diagram efektif digunakan pada tahap awal perancangan karena dapat mengomunikasikan ruang lingkup sistem secara ringkas kepada seluruh pihak yang terlibat dalam pengembangan, termasuk tim teknis maupun non-teknis.



Gambar 3.5. Use case diagram aplikasi.

Setelah use case teridentifikasi, masing-masing dijabarkan lebih lanjut menggunakan *Activity Diagram*. Diagram ini dipilih karena paling tepat untuk menggambarkan urutan tindakan yang melibatkan keputusan logika, percabangan, serta peran ganda antara pengguna dan sistem. Dibandingkan dengan use case diagram yang bersifat konseptual atau state diagram yang fokus pada perubahan status objek tunggal, activity diagram mampu memberikan gambaran menyeluruh terhadap proses bisnis dan logika interaksi sistem secara prosedural [24][25][26][27][28][29]. Hal ini penting agar ekspektasi pengembangan dapat selaras dengan realisasi implementasi teknis dalam sistem VR berbasis skenario.

Diagram ini menunjukkan pembagian peran antara pengguna sebagai pemicu interaksi awal, dan sistem sebagai eksekutor proses backend. Alur ini penting untuk memastikan bahwa setiap sesi simulasi dimulai dari titik yang konsisten dan terkontrol, sehingga pengguna mendapatkan pengalaman pelatihan yang terstruktur. Selain itu, diagram ini juga berfungsi sebagai referensi penting untuk memvalidasi kesiapan teknis sistem dalam menyediakan konteks dan aset virtual sebelum interaksi lebih lanjut dimulai.

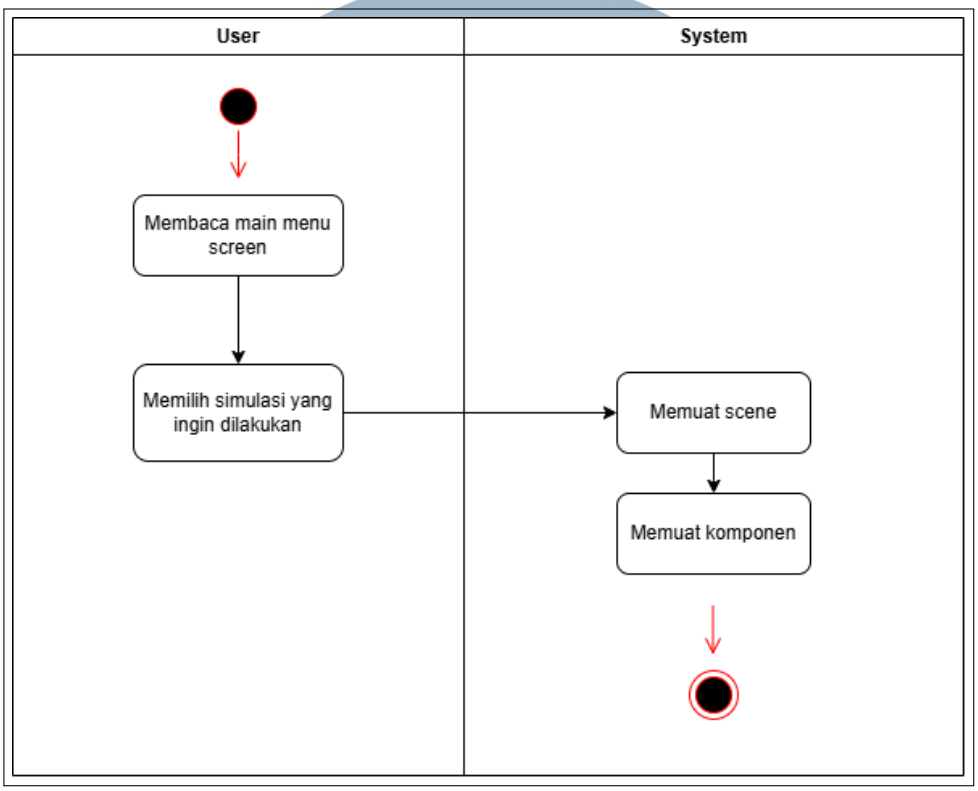
A.1 Activity Diagram: Memulai Simulasi

Activity diagram pada Gambar 3.6 menggambarkan langkah awal dari alur kerja simulasi pelatihan berbasis Virtual Reality (VR), dimulai dari interaksi awal pengguna hingga pemuatan skenario simulasi yang dipilih. Diagram ini berfungsi untuk memodelkan urutan aktivitas secara sistematis, yang merepresentasikan bagaimana pengguna berinteraksi dengan sistem sejak pertama kali memasuki aplikasi. Tahapan dimulai dari pengguna yang melihat tampilan awal berupa main menu screen, di mana tersedia berbagai pilihan skenario simulasi yang dapat dijalankan.

Setelah pengguna menentukan pilihan terhadap salah satu skenario, sistem akan memproses input tersebut dan memuat scene yang sesuai. Proses ini mencakup pengaturan ulang posisi awal pengguna di lingkungan virtual, inisialisasi komponen interaktif seperti alat, objek mesin, dan instruksi visual, serta aktivasi logika interaksi yang relevan untuk skenario tersebut. Dengan kata lain, sistem akan menyesuaikan seluruh konteks simulasi, baik dari sisi lingkungan maupun alur kerja, sesuai dengan kebutuhan skenario yang dipilih.

Penggunaan activity diagram dalam tahap ini bertujuan untuk membantu pengembang memahami keterkaitan antar aksi pengguna dan reaksi sistem secara

visual dan terstruktur, sehingga dapat menjadi dasar perancangan implementasi logika dalam aplikasi VR.



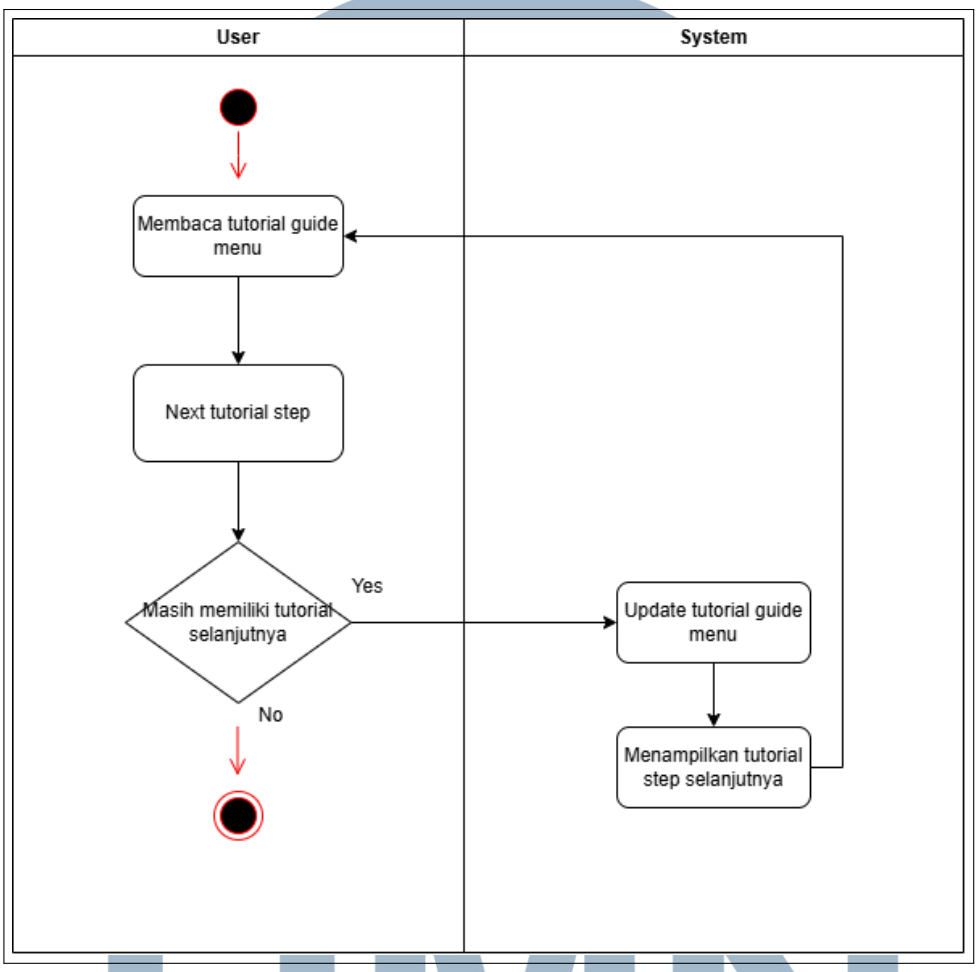
Gambar 3.6. Activity diagram start.

A.2 Activity Diagram: Membaca Tutorial

Activity diagram pada Gambar 3.7 merepresentasikan proses pembacaan instruksi atau langkah kerja oleh pengguna melalui tampilan tutorial guide menu. Diagram ini menggambarkan pola interaksi iteratif, di mana pengguna membaca satu langkah instruksi, kemudian memutuskan apakah masih terdapat langkah berikutnya. Jika ya, sistem akan memperbarui tampilan dan menampilkan konten instruksi tahap selanjutnya. Proses ini akan terus berulang hingga seluruh langkah dalam tutorial selesai dibaca.

Penggunaan pola iteratif dalam diagram ini mencerminkan dukungan sistem terhadap proses pembelajaran yang bertahap dan mandiri, sehingga pengguna dapat memahami urutan kerja tanpa tekanan waktu atau alur yang dipaksakan. Dengan demikian, diagram ini tidak hanya menunjukkan alur teknis sistem, tetapi juga mencerminkan pendekatan desain instruksional yang berpusat pada pengguna. Visualisasi ini sangat penting untuk memastikan bahwa pengguna

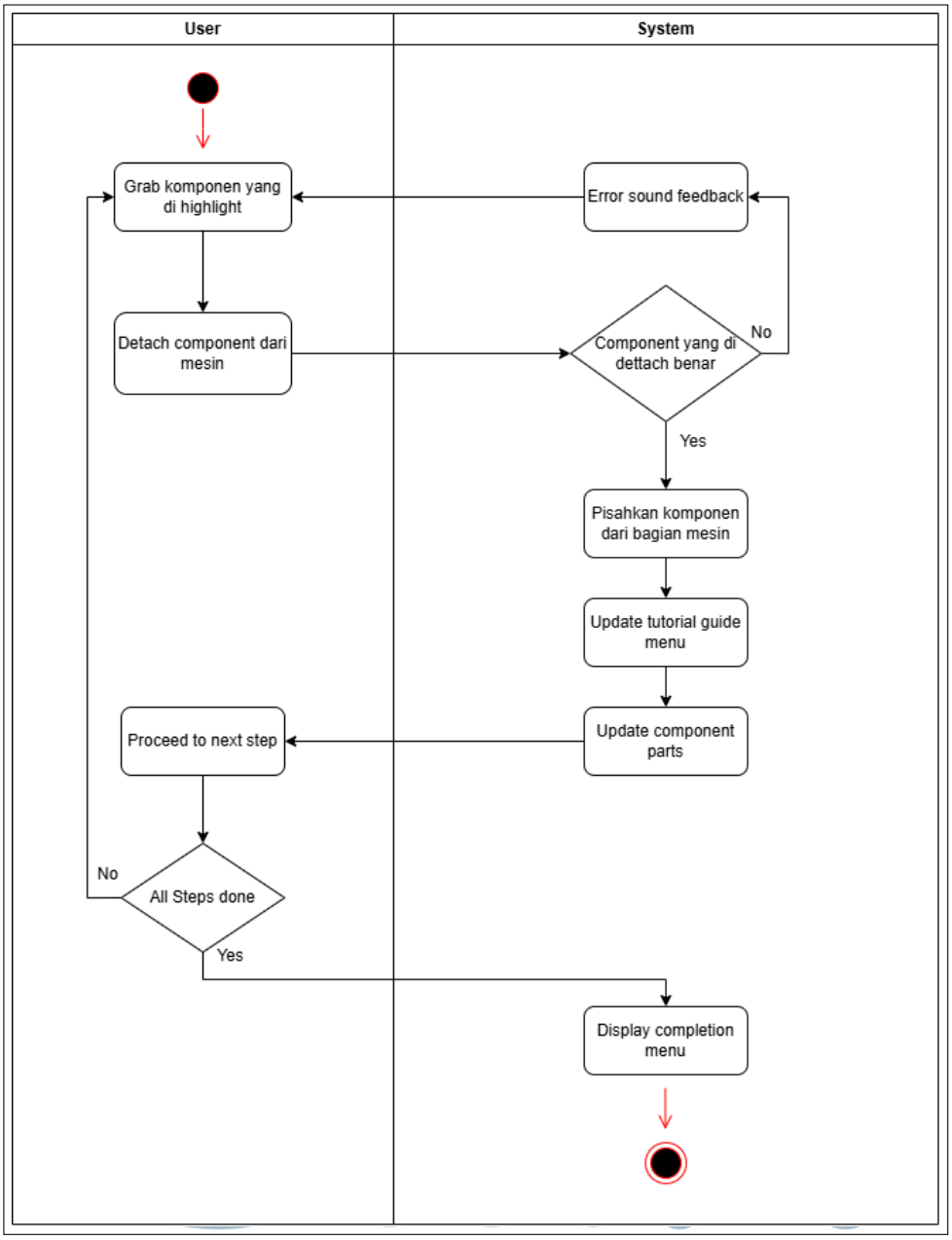
memiliki pemahaman yang memadai sebelum melakukan simulasi perakitan atau pembongkaran secara langsung.



Gambar 3.7. Activity diagram tutorial.

A.3 Activity Diagram: Melakukan Disassembly

Gambar 3.8 memperlihatkan alur kerja pengguna dalam proses pembongkaran komponen mesin. Diagram ini dimulai dari tindakan pengguna untuk mengambil (grab) komponen yang disorot, lalu melepaskannya dari bagian mesin. Sistem akan memvalidasi apakah komponen yang dilepas sudah benar, dan apabila tidak sesuai, sistem memberikan umpan balik berupa suara kesalahan. Jika berhasil, sistem akan memperbarui status komponen dan tutorial. Alur akan berulang hingga semua bagian selesai dilepas. Diagram ini menekankan pentingnya urutan kerja yang benar dan kontrol prosedural oleh sistem.

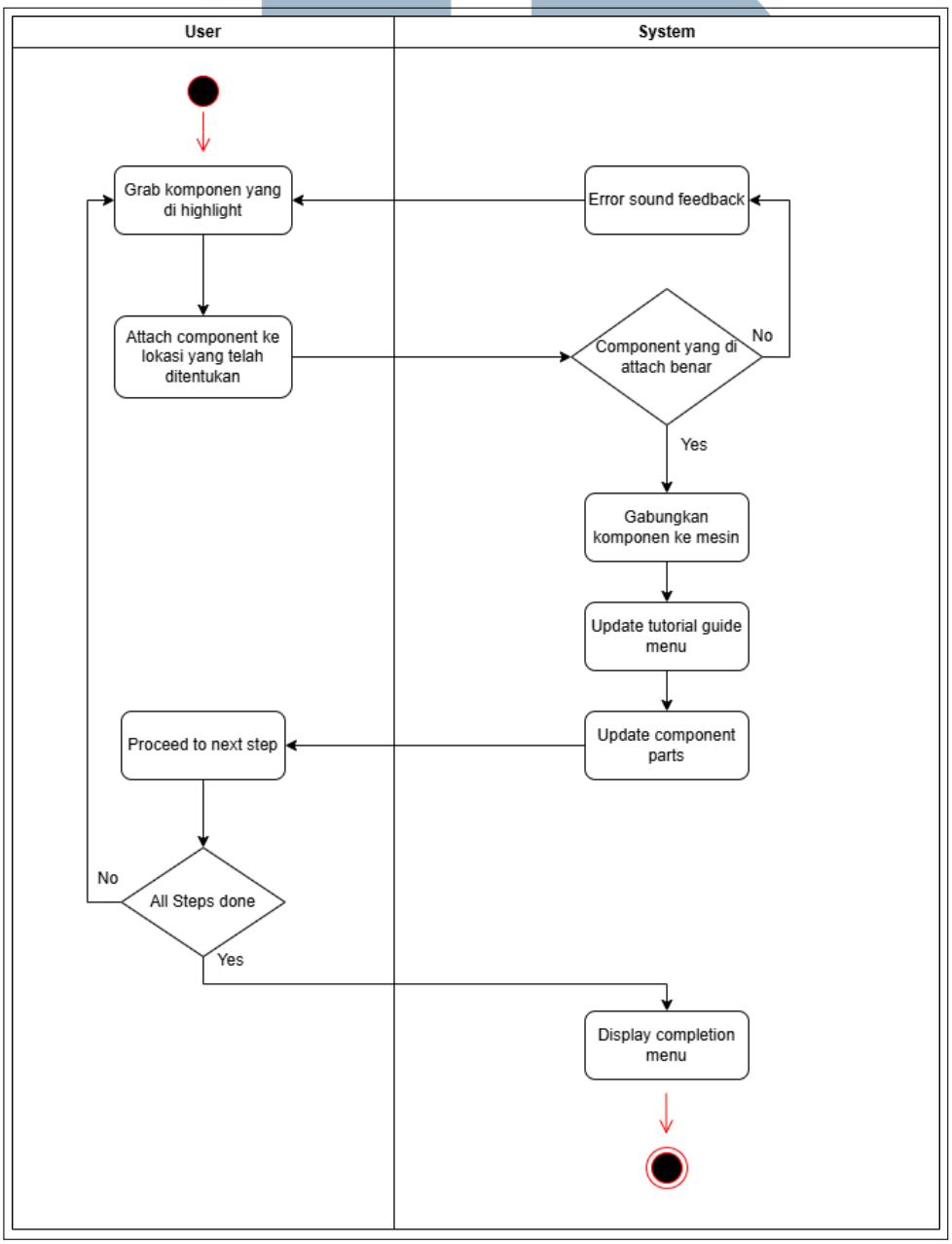


Gambar 3.8. Activity diagram disassembly.

A.4 Activity Diagram: Melakukan Assembly

Activity diagram pada Gambar 3.9 menggambarkan alur kerja pengguna saat melakukan perakitan komponen mesin. Dimulai dari pengambilan komponen yang di-highlight, pengguna menempatkan komponen tersebut ke lokasi yang telah ditentukan. Sistem kemudian memvalidasi posisi pemasangan. Jika salah, sistem memberikan umpan balik kesalahan dan pengguna harus mengulang. Jika benar,

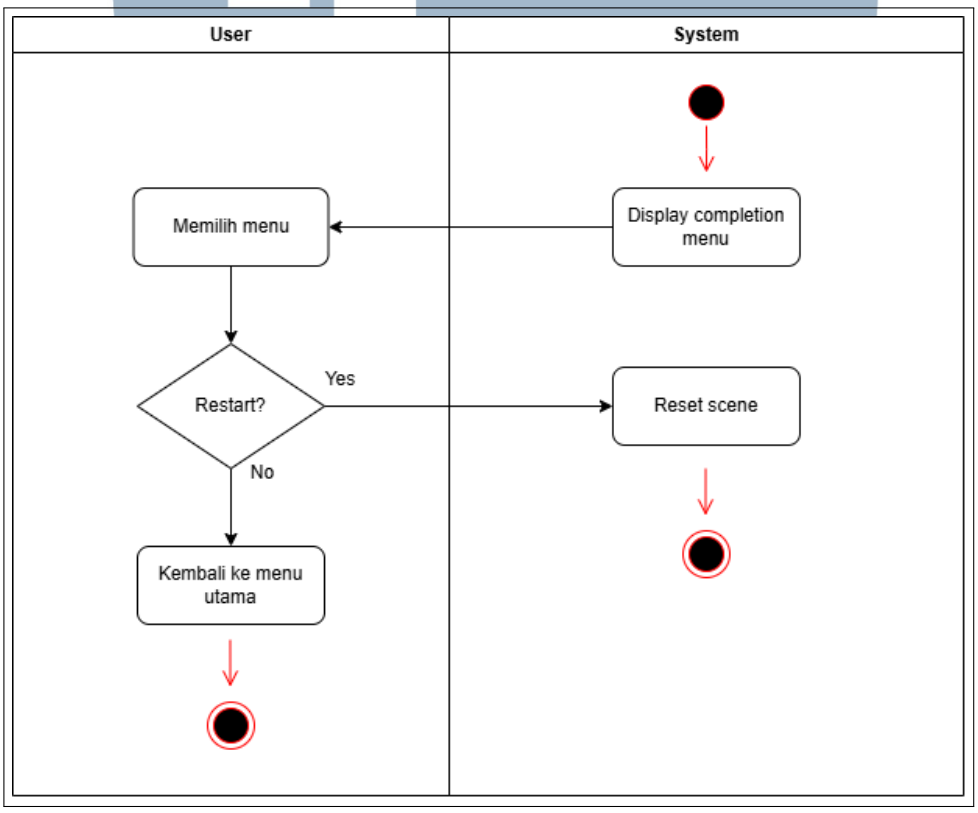
sistem menggabungkan komponen ke mesin dan memperbarui status tutorial serta komponen. Proses ini berulang hingga seluruh langkah perakitan selesai, kemudian sistem menampilkan menu penyelesaian. Diagram ini memperlihatkan perpaduan antara interaksi langsung dan logika validasi otomatis oleh sistem.



Gambar 3.9. Activity diagram assembly.

A.5 Activity Diagram: Menyelesaikan Simulasi

Gambar 3.10 menggambarkan akhir dari sesi simulasi. Setelah seluruh langkah selesai, sistem menampilkan menu penyelesaian. Pengguna kemudian diberikan opsi untuk mengulang simulasi atau kembali ke menu utama. Jika pengguna memilih untuk restart, sistem akan mereset ulang scene dan memulai kembali sesi. Jika memilih keluar, pengguna diarahkan kembali ke menu utama. Diagram ini menggunakan dua jalur akhir yang berbeda untuk merepresentasikan kondisi akhir yang bergantung pada keputusan pengguna, sesuai dengan prinsip percabangan logika dalam interaksi prosedural.

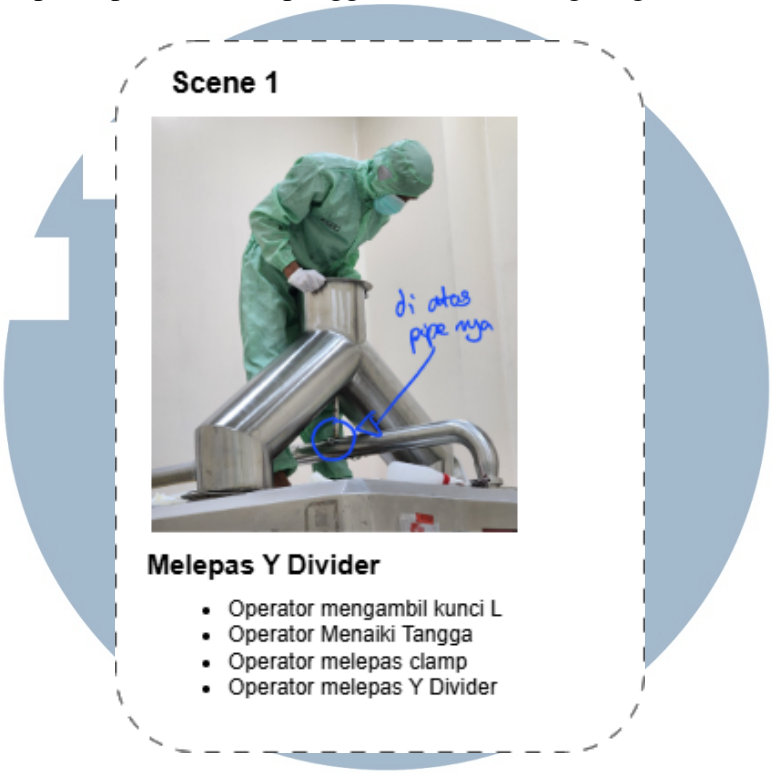


Gambar 3.10. Activity diagram end.

B Tahap 2.2: Pembuatan Storyboard

Dalam pengembangan aplikasi *Virtual Reality* (VR) untuk pelatihan operator mesin, perancangan storyboard menjadi langkah penting dalam menjembatani kebutuhan fungsional dengan pengalaman pengguna (user experience). *Storyboard* digunakan sebagai alat bantu visual untuk menggambarkan

alur simulasi secara berurutan melalui serangkaian sketsa atau tampilan yang mewakili setiap tahapan interaksi pengguna di dalam lingkungan virtual.



Gambar 3.11. Contoh storyboard skenario interaksi disassembly komponen.

Storyboard pada Gambar 3.11 menunjukkan skenario interaksi dalam tahap *disassembly*, di mana pengguna melakukan pembongkaran komponen mesin secara bertahap sesuai urutan prosedural. Visualisasi ini diambil berdasarkan hasil observasi lapangan serta diagram alur yang telah disusun sebelumnya. Tujuan dari storyboard ini adalah untuk menyusun kerangka visual yang dapat memandu proses implementasi simulasi VR secara lebih terarah, serta memastikan setiap langkah interaksi terekam secara sistematis dan konsisten dengan standar operasional.

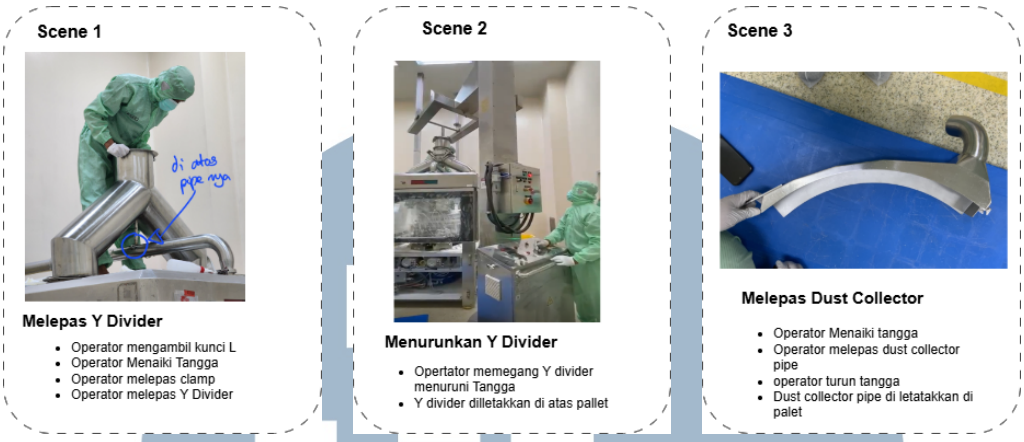
Setelah tahapan *disassembly* divisualisasikan, proses *assembly* turut didokumentasikan melalui storyboard untuk memastikan kedua skenario utama pelatihan tercakup secara menyeluruh. Pendokumentasian ini penting untuk menjaga keseimbangan antara proses pelepasan dan pemasangan komponen, sehingga pelatihan simulasi yang dikembangkan dapat merepresentasikan siklus kerja nyata secara utuh.

Gambar 3.12. Contoh storyboard skenario interaksi assembly komponen.

Storyboard dirancang berdasarkan hasil observasi lapangan, wawancara dengan teknisi berpengalaman, serta *flowchart* prosedur kerja yang telah divalidasi sebelumnya. Setiap *frame* dalam storyboard merepresentasikan satu momen penting dalam simulasi, seperti:

1. Pengambilan alat atau komponen.
2. Pemasangan bagian mesin secara berurutan.
3. Interaksi dengan tombol atau sistem validasi.
4. Respons terhadap kesalahan prosedural.
5. Transisi antar tahap pelatihan.

Pembuatan storyboard dilakukan secara kolaboratif antara tim pengembang dan tim Project Management (PM) untuk memastikan bahwa semua skenario kerja yang disimulasikan sesuai dengan kenyataan di lapangan dan mudah dipahami oleh pengguna akhir. Selain itu, storyboard juga digunakan sebagai bahan komunikasi visual kepada stakeholder untuk menyelaraskan ekspektasi sebelum proses pengembangan teknis dimulai.



Gambar 3.13. Storyboard awal skenario disassembly mesin.

Storyboard juga berperan penting dalam proses iterasi awal, karena memungkinkan tim untuk:

1. Mengidentifikasi celah atau ketidaksesuaian alur sebelum implementasi teknis,
2. Menyampaikan ide secara visual kepada anggota tim yang berbeda latar belakang teknis,
3. Menyusun logika interaksi yang dapat diubah dengan cepat berdasarkan masukan klien,
4. Mempersiapkan konten visual untuk PoC (Proof of Concept).

Dalam implementasinya, storyboard menjadi salah satu dokumen acuan utama yang diterjemahkan ke dalam interaksi fungsional di dalam Unity. Setiap frame dalam storyboard dikonversi menjadi adegan interaktif yang dilengkapi dengan objek 3D, kontrol pengguna, serta sistem validasi berbasis logika pemrograman. Proses ini dilakukan secara bertahap dan divalidasi secara langsung menggunakan perangkat Head-Mounted Display (HMD) untuk memastikan kesesuaian antara rancangan dan hasil implementasi.

C Tahap 2.3: Pembuatan Proof of Concept (PoC)

Proof of Concept (PoC) merupakan prototipe awal yang dikembangkan untuk memvalidasi kelayakan teknis dan konsep interaksi dari aplikasi. PoC

ini mencakup elemen-elemen dasar seperti pengenalan objek, interaksi *grab-release*, tombol virtual, dan sistem validasi sederhana. PoC dikembangkan segera setelah tahapan observasi lapangan (genba) selesai dilakukan, dan secara strategis juga dimanfaatkan sebagai aktivitas pengembangan paralel sembari menunggu penyediaan aset 3D dari tim desainer.

Pada tahap PoC ini, fokus simulasi ditujukan pada proses perakitan (assembly) dan pembongkaran (disassembly) komponen kabinet, sebagai salah satu bagian penting dalam pelatihan operator. Komponen ini dipilih karena mewakili struktur prosedural yang umum dalam pekerjaan teknis, dan memungkinkan tim untuk menguji berbagai jenis interaksi dasar seperti *grab*, *release*, validasi urutan, dan tombol virtual.

Tujuan utama dari PoC adalah:

1. Memastikan alur kerja dasar dapat diimplementasikan secara fungsional di Unity.
2. Melakukan pengujian performa awal pada perangkat HMD (Meta Quest 3).
3. Menyampaikan demonstrasi awal kepada klien dan tim PM untuk mendapatkan umpan balik.

PoC juga memainkan peran penting sebagai alat komunikasi visual dan teknis untuk memfasilitasi diskusi lintas tim. Hasil dari PoC ini digunakan untuk mengevaluasi konsep alur, kenyamanan pengguna, dan potensi tantangan teknis sejak dini.

Berikut ini ditampilkan dokumentasi visual dan teknis dari tahapan PoC, termasuk tangkapan layar dari tampilan Unity, serta potongan kode interaksi dasar yang digunakan untuk membangun logika perakitan.

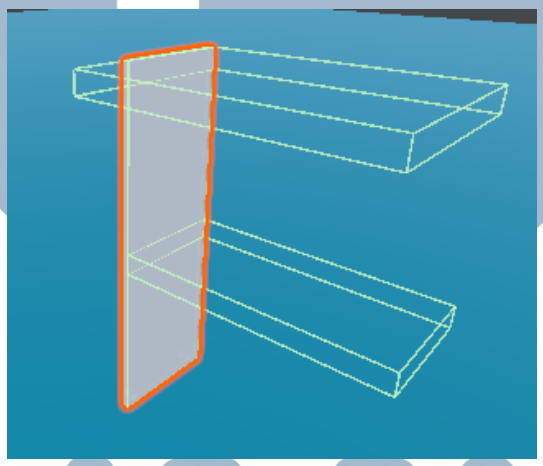
Pengembangan PoC ini mencakup beberapa skrip utama yang mendasari logika interaksi dalam simulasi. Salah satu skrip yang paling sentral adalah `ScSocketActive.cs`. Skrip ini berfungsi untuk mengelola proses ketika sebuah objek utama (seperti bagian kabinet) dipasangkan ke dalam *socket*. Saat pengguna berhasil menempatkan objek tersebut ke socket, sistem akan menonaktifkan deteksi tumbukan antar objek agar simulasi tetap stabil secara fisik. Proses ini dilakukan melalui iterasi *collider* yang dikendalikan dengan fungsi berikut:

```
foreach (GameObject socket in sockets)
{
```

```
    Collider[] colliders = socket.GetComponentsInChildren<Collider
    >();
    foreach (Collider col in colliders)
    {
        Physics.IgnoreCollision(currentCollider, col);
    }
}
```

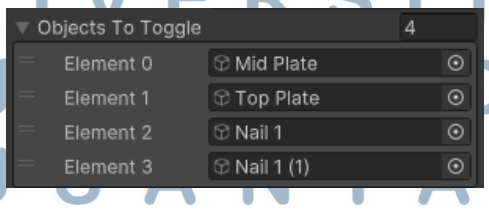
Kode 3.1: Menonaktifkan tumbukan antar collider pada objek socket

Setelah proses penonaktifan tumbukan diterapkan melalui kode 3.1, konfigurasi socket dan *collider* di dalam Unity divisualisasikan seperti ditunjukkan pada Gambar 3.14.



Gambar 3.14. Tampilan socket dan collider sebagai lokasi pemasangan objek bagian kabinet.

Untuk mendukung pengaturan logika interaktif, Unity Inspector digunakan untuk menetapkan daftar objek yang dikontrol, seperti *Mid Plate*, *Top Plate*, dan beberapa objek *nail*. Konfigurasi ini dapat dilihat pada pengaturan *array* `Objects To Toggle` di bawah ini:



Gambar 3.15. Inspector Unity: konfigurasi daftar objek yang akan diaktifkan setelah pemasangan plank.

Pengaturan ini memungkinkan sistem untuk mengaktifkan objek baru

seperti nail ketika bagian kabinet utama sudah dipasang. Hal ini menciptakan alur simulasi bertahap yang mengikuti prosedur kerja asli.

Setelah objek terpasang, sistem akan memicu perubahan visual dengan mengganti material objek menjadi warna hijau, memberikan umpan balik visual bahwa langkah tersebut berhasil. Selain itu, skrip ini juga mengaktifkan objek-objek nail untuk tahap berikutnya.

```
Renderer objRenderer = obj.GetComponent<Renderer>();
if (objRenderer != null)
{
objRenderer.material = snapGreenMaterial; // Highlight hijau
obj.SetActive(true);
}
```

Kode 3.2: Perubahan material saat objek terpasang ke socket

Penggunaan material *snapGreenMaterial* atau *whiteMaterial* dilakukan secara kondisional, tergantung pada konteks, apakah objek sedang aktif, dilepas, atau berhasil digabungkan. Visualisasi ini sangat penting dalam lingkungan VR untuk memberikan petunjuk tanpa perlu antarmuka tambahan.

Langkah selanjutnya melibatkan pemasangan *nail* ke dalam *socket*. Untuk ini digunakan skrip `ScNail.cs`, yang mencatat jumlah pukulan palu yang diterima oleh setiap paku. Saat palu mengenai paku, metode `HitNail()` akan dijalankan. Metode ini akan menambahkan hitungan ke variabel hammerCount, menggeser posisi *nail* agar tampak tertanam, serta memainkan efek suara.

```
public void HitNail()
{
    if (!IsFullyHammered && currentSocket != null)
    {
        hammerCount++;
        currentSocket.transform.localPosition += movementDirection
   ;
        source.PlayOneShot(hit);
        OnHammered?.Invoke();
    }
}
```

Kode 3.3: Menangani logika pemukulan paku

Pemanggilan `OnHammered.Invoke()` digunakan untuk memberi tahu sistem bahwa status paku telah berubah. Ketika semua paku dalam suatu tahap telah berhasil dipukul hingga penuh, sistem akan mengeksekusi fungsi

`CheckAllNailsHammered()` yang berasal dari skrip `ScSocketActive`. Fungsi ini akan memverifikasi apakah semua paku telah dipasang dengan benar dan kemudian menggabungkan objek kabinet secara fisik dengan menggunakan `FixedJoint`.

```
fixedJoint = gameObject.AddComponent<FixedJoint>();
fixedJoint.connectedBody = parentRigidbody;
fixedJoint.breakForce = Mathf.Infinity;
fixedJoint.breakTorque = Mathf.Infinity;
```

Kode 3.4: Penggabungan objek kabinet menggunakan FixedJoint

Dengan menambahkan `FixedJoint`, sistem mensimulasikan bahwa dua bagian objek telah terhubung secara permanen. Penggunaan `breakForce` dan `breakTorque` yang sangat tinggi memastikan bahwa sambungan tidak akan terlepas secara tidak sengaja karena interaksi pengguna.

Deteksi tumbukan antara palu dan paku dikendalikan oleh skrip `Hammer.cs`. Skrip ini secara selektif memproses hanya tumbukan yang terjadi pada bagian kepala palu, untuk menghindari pemrosesan interaksi palsu. Saat deteksi valid terjadi, sistem akan langsung memanggil fungsi `HitNail()` dari objek yang dikenai.

```
if (contact.thisCollider == hammerHeadCollider && hitObject.
    CompareTag("Nail"))
{
    ScNail nail = hitObject.GetComponent<ScNail>();
    if (nail != null) nail.HitNail();
}
```

Kode 3.5: Deteksi tumbukan valid antara kepala palu dan paku

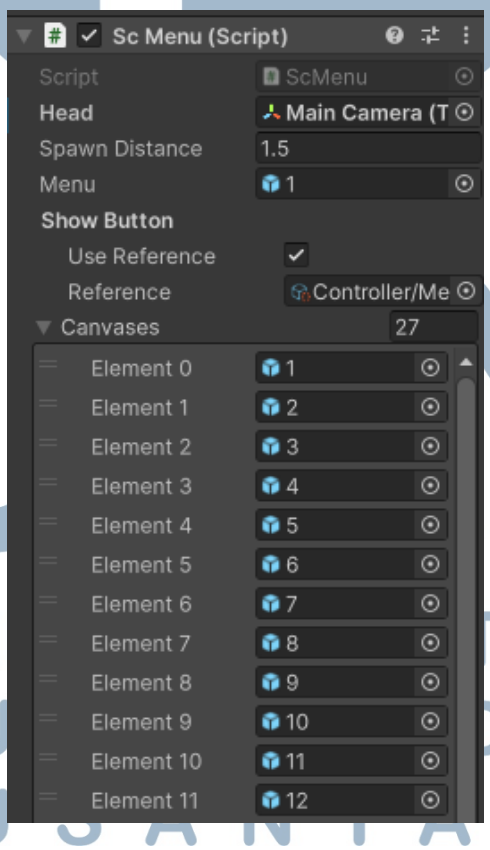
Selain logika interaksi fisik, PoC juga mengimplementasikan antarmuka pengguna menggunakan kanvas. Navigasi antar langkah simulasi dilakukan melalui skrip `ScMenu.cs`, yang memungkinkan pengguna untuk berpindah dari satu kanvas ke kanvas lain menggunakan input tertentu. Menu juga secara otomatis ditempatkan di depan pengguna berdasarkan posisi headset, menggunakan logika sebagai berikut:

```
menu.transform.position = head.position + new Vector3(head.forward
    .x, 0, head.forward.z).normalized * spawnDistance;
menu.transform.LookAt(new Vector3(head.position.x, menu.transform.
    position.y, head.position.z));
menu.transform.forward *= -1;
```

Kode 3.6: Penempatan menu dinamis berdasarkan posisi HMD

Gambar 3.16. Tampilan sistem menu dalam simulasi VR. Menu akan selalu muncul menghadap pengguna.

Konfigurasi kanvas ini ditampilkan dalam Unity Inspector sebagai daftar Canvas berurutan yang merepresentasikan tahapan-tahapan pelatihan. Dalam proyek ini terdapat lebih dari 20 kanvas, masing-masing mewakili satu langkah atau status simulasi tertentu, seperti tahap pemasangan, pemukulan, validasi, dan sebagainya.



Gambar 3.17. Inspector Unity: daftar canvas pada script ScMenu yang mewakili langkah-langkah pelatihan.

Terakhir, untuk meningkatkan pengalaman interaksi, digunakan skrip

`DynamicHoverMaterial.cs` yang bertugas mengubah warna objek saat pengguna melakukan hover, attach, atau detach. Misalnya, ketika kontroler pengguna berada di atas suatu objek, material objek tersebut akan diubah ke material hover yang telah ditentukan.

```
if (args.interactorObject.transform.CompareTag("Controller"))
{
    originalMaterial = objectRenderer.material;
    objectRenderer.material = hoverMaterial;
}
```

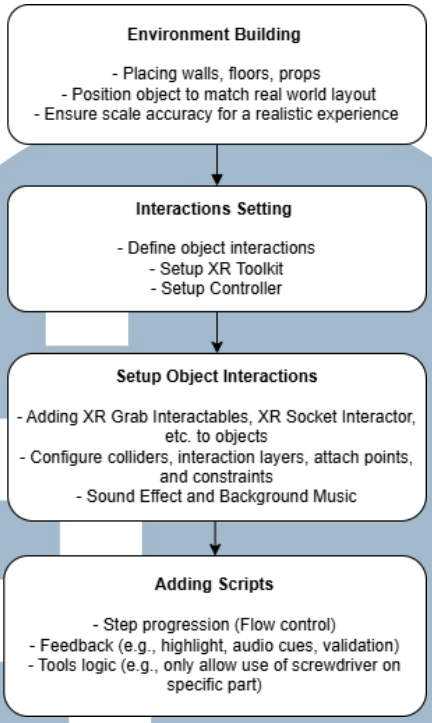
Kode 3.7: Perubahan material saat objek di-hover oleh pengguna

Keseluruhan sistem dalam PoC ini berhasil menunjukkan bahwa interaksi perakitan secara prosedural dapat disimulasikan secara imersif dan logis dalam lingkungan VR. Pengujian dilakukan langsung di perangkat *Meta Quest 3* dan menunjukkan bahwa seluruh interaksi berjalan dengan stabil. PoC ini juga memberikan landasan teknis yang kuat untuk dikembangkan lebih lanjut dalam simulasi pelatihan yang lebih kompleks dan realistis.

3.5.6 Tahap 3: Development

Tahap pengembangan (development) merupakan inti dari pelaksanaan proyek, di mana seluruh fitur dan interaksi virtual dibangun berdasarkan hasil perancangan sebelumnya. Proses ini dilakukan secara iteratif menggunakan pendekatan *Agile*, yang memungkinkan tim untuk melakukan evaluasi dan perbaikan secara bertahap selama masa pengembangan. Aktivitas utama dalam fase ini mencakup pengaturan lingkungan virtual, pemrograman interaksi, serta penyesuaian terhadap aspek pengalaman pengguna (user experience).

Untuk menjaga konsistensi, efisiensi, dan skalabilitas selama implementasi di Unity, disusunlah alur kerja teknis yang menggambarkan urutan tahapan utama dalam proses pengembangan. Alur ini juga menjadi panduan utama struktur kerja yang dimulai dari inisialisasi proyek, penyiapan pustaka (library) dan perangkat lunak pendukung, hingga pembuatan interaksi dan validasi logika sistem. Setiap tahap dirancang agar saling terintegrasi, sehingga memudahkan tim pengembang dalam melacak progres serta mengidentifikasi potensi hambatan sejak dini. Diagram berikut menunjukkan struktur proses pengembangan aplikasi VR secara umum dan dapat dijadikan referensi dalam pengembangan proyek-proyek serupa di masa depan.



Gambar 3.18. Sub-flowchart tahapan teknis dalam pengembangan aplikasi VR di Unity.

A Tahap 3.1: Setup VR Project (Environment, SDK, Library)

Langkah awal dari tahap pengembangan adalah melakukan inisialisasi proyek Unity dengan konfigurasi yang kompatibel dengan perangkat *standalone VR* seperti *Meta Quest 3*. Tujuan dari tahap ini adalah memastikan bahwa seluruh komponen yang dibutuhkan untuk pengembangan aplikasi berbasis *Virtual Reality* telah tersedia dan berjalan dengan baik dalam lingkungan Unity.

Beberapa komponen penting yang dikonfigurasi pada proyek ini antara lain:

Langkah awal pengembangan dimulai dengan melakukan inisialisasi proyek Unity yang kompatibel dengan perangkat *Meta Quest 3*. Untuk menjamin kompatibilitas serta kelancaran proses pengembangan dan kompilasi ke dalam perangkat VR, beberapa konfigurasi dasar dan instalasi *dependency* wajib dilakukan sejak awal. Konfigurasi dilakukan pada Unity Hub dan Unity Editor dengan memperhatikan beberapa komponen berikut:

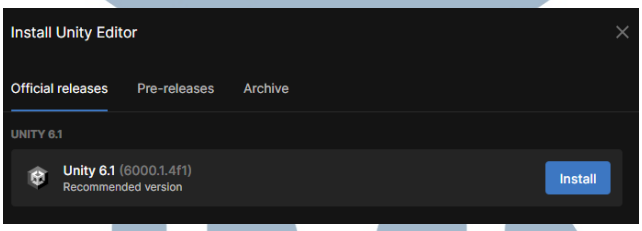
1. Unity Editor versi terbaru.
2. Android Build Support beserta dependensinya, yaitu:
    - Android SDK NDK Tools

- OpenJDK

3. XR Plugin Management (untuk mendukung perangkat VR).
4. XR Interaction Toolkit versi 3.0.8 (melalui Unity Package Manager).
5. Unity Input System versi 1.4.0 atau lebih baru (untuk kompatibilitas kontroler dan input VR).
6. Mock HMD XR Plugin (untuk pengujian dalam editor tanpa HMD).

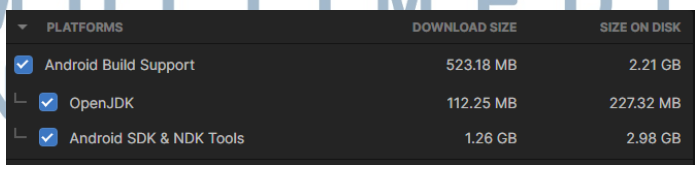
Pengaturan Unity Editor dan dukungan platform Android sangat penting karena build akhir aplikasi VR akan dijalankan langsung pada perangkat Meta Quest yang menggunakan sistem operasi berbasis Android.

Untuk memastikan kompatibilitas maksimal, Unity Editor yang digunakan adalah versi terbaru yang direkomendasikan oleh Unity, yaitu 2022.3.43f1 LTS atau Unity 6.1.0f1 untuk sekarang. Versi ini dapat diinstal melalui tab *Official Releases* seperti tampak pada Gambar 3.19:



Gambar 3.19. Instalasi Unity Editor.

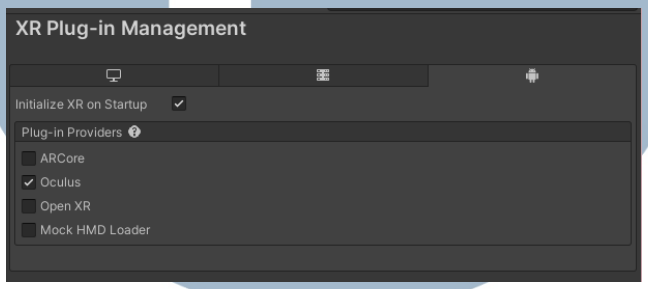
Kemudian, pastikan bahwa komponen Android Build Support telah diaktifkan pada Unity Hub. Komponen ini mencakup Android SDK NDK Tools serta OpenJDK, yang merupakan bagian penting dalam proses build aplikasi ke perangkat Android seperti *Meta Quest 3*. Ketiga komponen ini harus dicentang agar Unity dapat melakukan kompilasi dan deployment dengan benar ke platform Android. Tampilan pemilihan komponen tersebut ditunjukkan pada Gambar 3.20.



Gambar 3.20. Pengaturan modul Android Build Support pada Unity Hub untuk kompilasi ke perangkat Meta Quest.

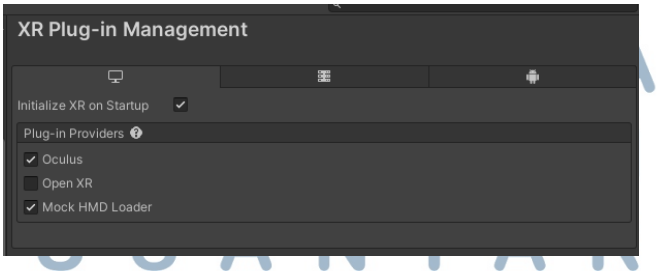
Setelah instalasi komponen dasar selesai, tahap krusial berikutnya adalah konfigurasi XR Plug-in Management. Fitur ini dapat diakses melalui menu `Edit > Project Settings > XR Plug-in Management`, dan berfungsi sebagai pengatur integrasi antara Unity dan perangkat XR seperti Meta Quest 3. Plugin ini menyediakan antarmuka untuk memilih penyedia XR yang digunakan dalam proyek, baik untuk kebutuhan runtime maupun pengujian dalam editor.

Untuk platform Android, yang merupakan target utama Meta Quest, menggunakan opsi `Oculus` sebagai XR provider utama. Hal ini memungkinkan Unity untuk mengenali Meta Quest sebagai perangkat Oculus dan mengaktifkan semua layanan yang dibutuhkan untuk rendering stereoskopik dan pelacakan kontroler:



Gambar 3.21. Pengaturan XR Plug-in Management untuk platform Android menggunakan Oculus sebagai penyedia utama.

Untuk platform PC (standalone), terutama saat pengujian dilakukan langsung di dalam editor tanpa menggunakan perangkat headset, aktifkan plugin `Mock HMD Loader`. Plugin ini mensimulasikan perangkat HMD dan kontroler virtual, sehingga pengembang tetap dapat melakukan pengujian dan debugging tanpa harus menjalankan build langsung ke headset. Pada konfigurasi ini, `Oculus` juga tetap diaktifkan sebagai *fallback* saat HMD sesungguhnya tersedia:



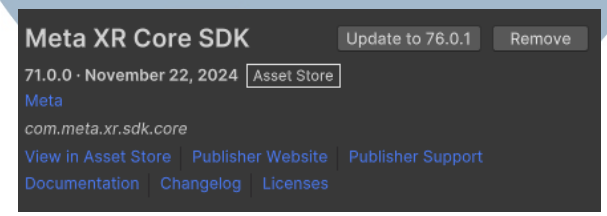
Gambar 3.22. Pengaturan XR Plug-in Management untuk platform PC dengan Mock HMD dan Oculus diaktifkan.

Dengan kombinasi pengaturan seperti di atas, proses pengembangan

menjadi jauh lebih fleksibel. Dapat digunakan Mock HMD untuk pengujian di awal, dan dengan cepat berpindah ke pengujian langsung di perangkat Meta Quest hanya dengan mengganti target platform menjadi Android.

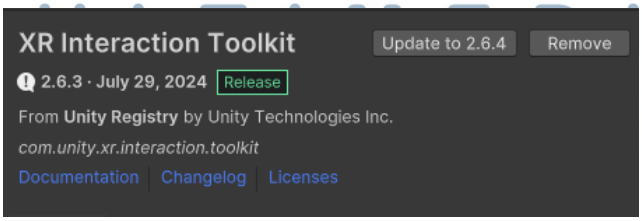
Setelah konfigurasi XR Plug-in Management selesai, tahap berikutnya adalah melakukan instalasi dan integrasi SDK serta pustaka interaksi yang dibutuhkan untuk mendukung pengembangan antarmuka pengguna berbasis XR. Hal ini dilakukan melalui *Unity Package Manager* maupun *Asset Store*.

Komponen utama yang perlu diimpor adalah *Meta XR Core SDK*. SDK ini dikembangkan langsung oleh pihak *Meta (Facebook)* dan menyediakan berbagai modul penting untuk perangkat *Oculus/Meta Quest*, termasuk *tracking system*, *hand input*, *stereo rendering system*, dan integrasi dengan layanan *Oculus Runtime*. Paket ini dapat diunduh melalui *Unity Asset Store* dan kemudian ditambahkan ke dalam proyek Unity. Versi yang digunakan pada proyek ini adalah versi `71.0.0` seperti ditunjukkan pada Gambar 3.23:



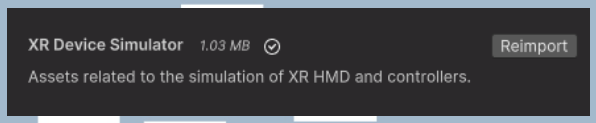
Gambar 3.23. Meta XR Core SDK dari Unity Asset Store digunakan untuk mendukung interaksi dan tracking di perangkat Meta Quest.

Langkah selanjutnya adalah menginstal *XR Interaction Toolkit*. Toolkit ini merupakan pustaka standar dari Unity untuk menangani interaksi di lingkungan XR (Extended Reality), baik untuk VR maupun AR. Dengan *XR Interaction Toolkit*, pengembang dapat dengan mudah menerapkan sistem grab, hover, socket, teleporasi, dan lainnya tanpa harus membangun semuanya dari awal. Versi yang digunakan dalam proyek ini adalah `2.6.3`, yang diunduh langsung dari *Unity Registry*:



Gambar 3.24. XR Interaction Toolkit digunakan sebagai basis sistem interaksi dalam simulasi VR.

Untuk mendukung proses pengembangan tanpa memerlukan headset atau kontroler fisik, Unity juga menyediakan *XR Device Simulator*. Fitur ini merupakan bagian dari *XR Interaction Toolkit* dan memungkinkan pengguna untuk mengendalikan simulasi XR hanya dengan *keyboard* dan *mouse*. Hal ini sangat membantu saat pengujian awal sebelum dilakukan deploy ke perangkat *Meta Quest*. *Simulator* ini juga dapat digunakan untuk navigasi, interaksi *grab*, dan validasi logika interaksi di dalam Unity Editor:



Gambar 3.25. XR Device Simulator diaktifkan untuk mendukung pengembangan dan pengujian tanpa HMD dan kontroler fisik.

Ketiga komponen tersebut—*Meta XR Core SDK, XR Interaction Toolkit, dan XR Device Simulator*—membentuk fondasi dari sistem interaksi yang dikembangkan dalam proyek ini. Penggabungan antara pustaka resmi dari Meta dan toolkit native Unity memastikan stabilitas, kompatibilitas, dan efisiensi dalam proses pengembangan serta pengujian aplikasi VR.

Jika terdapat pembaruan versi dari *Meta SDK* maupun *XR Toolkit*, maka Unity juga menyediakan fitur update langsung melalui *Unity Package Manager* untuk menjaga kompatibilitas terhadap sistem dan perangkat terbaru. Kombinasi pengaturan ini memungkinkan tim pengembang untuk fokus pada logika bisnis dan interaksi pengguna tanpa harus mengkhawatirkan aspek teknis *low-level* terkait integrasi perangkat XR.

B Tahap 3.2: Setup Player and Controls

Setelah lingkungan proyek berhasil dikonfigurasi, tahap selanjutnya adalah mengatur sistem *Player* yang merepresentasikan pengguna dalam simulasi VR. Pada Unity, representasi ini difasilitasi oleh *XR Rig*, yaitu prefab yang menyediakan struktur dasar untuk kamera dan kontroler berbasis XR. Pengaturan *XR Rig* sangat krusial karena elemen inilah yang memungkinkan pengguna berinteraksi secara langsung dengan lingkungan virtual, baik melalui pergerakan kepala (head tracking) maupun tangan (controller input).

XR Rig terdiri dari beberapa komponen utama berikut:

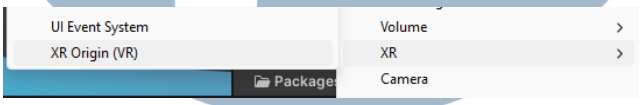
1. Main Camera: bertindak sebagai titik pandang pengguna (HMD / headset)

dalam ruang 3D. Kamera ini akan bergerak mengikuti posisi dan rotasi kepala pengguna secara real-time.

2. Left Controller dan Right Controller: masing-masing mewakili tangan kiri dan kanan pengguna. Posisi, rotasi, serta input tombol pada controller dapat digunakan untuk mengendalikan objek, melakukan grab, teleportasi, ataupun interaksi spesifik lainnya.

3. XR Origin atau XRRig Object: objek induk yang menampung semua elemen XR, termasuk kamera dan kontroler. Posisi awal dari XR Origin menentukan posisi spawn pengguna dalam scene ketika aplikasi dijalankan.

Pengaturan awal dilakukan dengan menambahkan prefab `XR Origin (Action-based)` dari XR Interaction Toolkit ke dalam scene. Setelah itu, setiap bagian seperti kamera dan controller dikonfigurasi agar merespons input sistem dari Unity Input System yang telah diaktifkan sebelumnya.



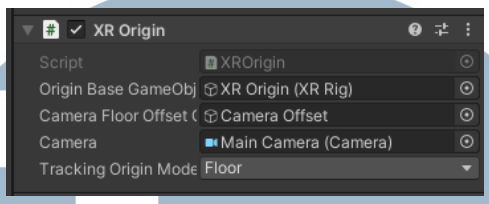
Gambar 3.26. Menambahkan XR Rig ke dalam scene.

Penyesuaian juga dilakukan untuk memastikan *XR Rig* berada pada posisi awal yang sesuai di dalam scene, terutama agar pengguna tidak langsung berada terlalu dekat dengan objek atau collider ketika simulasi dimulai. Posisi dan orientasi kontroler pun diperiksa agar selaras dengan orientasi tangan yang realistis di dunia nyata.

Untuk memastikan kamera mengikuti tinggi kepala pengguna dengan akurat, parameter Tracking Origin Mode pada komponen `XR Origin` perlu disesuaikan. Nilai yang digunakan adalah *Floor*, yang berarti sistem akan menganggap lantai sebagai referensi nol (origin). Dengan demikian, posisi kamera secara otomatis akan disesuaikan berdasarkan posisi vertikal kepala pengguna dalam ruang nyata, dan bukan hanya berdasarkan posisi relatif terhadap perangkat.

Pengaturan ini sangat penting untuk mendukung pengalaman imersif di VR karena memungkinkan sistem menyesuaikan perspektif secara alami. Sebagai contoh, ketika pengguna berdiri, duduk, atau membungkuk, kamera akan mengikuti perubahan tinggi tersebut dan memperbarui sudut pandang dengan real-time sesuai pelacakan HMD (Head-Mounted Display).

Pengaturan ini dapat dilakukan langsung melalui inspector Unity seperti ditunjukkan pada Gambar 3.27:



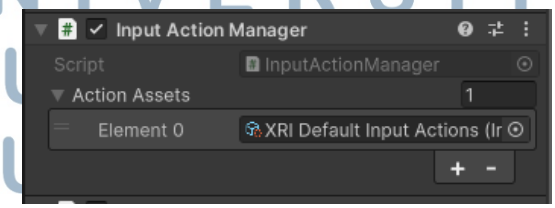
Gambar 3.27. Pengaturan Tracking Origin Mode pada komponen XR Origin ke mode `Floor`.

Dengan mengaktifkan mode ini, sistem XR akan memberikan pengalaman visual yang konsisten dengan tinggi fisik pengguna, serta mendukung head tracking secara penuh. Ini adalah salah satu komponen fundamental untuk menciptakan pengalaman pengguna VR yang realistis dan ergonomis.

Langkah selanjutnya adalah menambahkan komponen `Input Action Manager`. Komponen ini bertugas sebagai penghubung antara sistem input (Unity Input System) dengan aksi-aksi XR yang telah didefinisikan di dalam *action asset*. Tanpa komponen ini, input dari perangkat seperti kontroler atau headset tidak akan bisa dikenali dengan benar oleh Unity.

Untuk mengaktifkan sistem input secara penuh, tambahkan `Input Action Manager` ke dalam XR Rig dan set field `Action Assets` dengan `XRI Default Input Actions`. File ini merupakan konfigurasi default yang disediakan oleh Unity dan sudah mencakup seluruh aksi umum dalam aplikasi XR, seperti:

1. `Select`, `Activate`, dan `UI Press`
2. `Move`, `Turn`, dan `Teleport`
3. `HMD tracking`, `hand pose`, dan sebagainya



Gambar 3.28. Menambahkan komponen Input Action Manager dan mengatur XRI Default Input Actions sebagai aset aksi.

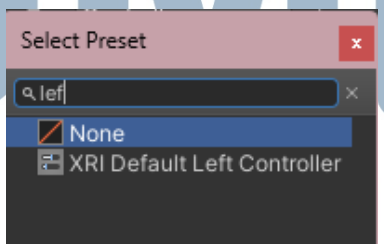
Penerapan konfigurasi ini akan secara otomatis menghubungkan seluruh komponen dalam *XR Rig* ke sistem input, baik itu kamera (untuk pelacakan kepala), maupun kontroler (untuk interaksi dan navigasi). Setelah tahap ini selesai, kontroler akan mampu merespons aksi seperti *grab*, tekan tombol, dan gestur lainnya sesuai konfigurasi default yang telah ditentukan.

Selain konfigurasi *XR Origin* dan *Tracking Origin Mode*, pengaturan input pada kontroler juga merupakan aspek penting dalam simulasi VR. Untuk mempercepat proses setup dan menghindari konfigurasi manual yang kompleks, Unity menyediakan fitur *Preset* pada XR Controller, yaitu kumpulan konfigurasi standar untuk input yang dapat langsung diterapkan ke objek kontroler.

Preset ini dapat diakses melalui komponen `XR Controller (Action-based)` baik untuk tangan kiri maupun tangan kanan. Unity menyediakan dua preset default utama, yaitu:

1. `XRI Default Left Controller` – Digunakan untuk mengatur input kontroler kiri.
2. `XRI Default Right Controller` – Digunakan untuk mengatur input kontroler kanan.

Dengan memilih preset ini, seluruh mapping input seperti `select`, `activate`, `ui press`, dan lain-lain akan langsung terisi berdasarkan konfigurasi standar dari *XR Interaction Toolkit*. Hal ini sangat membantu untuk memastikan bahwa interaksi dasar seperti grab, teleport, dan interaksi UI dapat berjalan tanpa perlu mengatur aksi satu per satu.



Gambar 3.29. Pemilihan preset `XRI Default Left Controller` pada kontroler kiri.

Penggunaan preset tidak hanya mempercepat proses setup, tetapi juga memastikan bahwa proyek mengikuti praktik standar Unity untuk XR, sehingga kompatibilitas dan performa tetap optimal. Setelah preset diterapkan, kontroler akan langsung merespons input seperti trigger, grip, joystick movement, dan lainnya, sesuai dengan pengaturan Unity Input System.

Agar pengguna dapat melihat representasi tangan mereka di dalam simulasi, tahap berikutnya adalah menambahkan model tangan 3D ke masing-masing kontroler. Model tangan ini berfungsi sebagai umpan balik visual langsung atas interaksi yang dilakukan, seperti mengambil objek, menekan tombol, atau sekadar menggerakkan tangan di dunia virtual.

Untuk menampilkan model tangan, cukup tambahkan prefab atau objek 3D tangan sebagai *child* dari `Left Controller` dan `Right Controller`. Unity menyediakan beberapa model tangan default yang dapat diimpor dari `XR Interaction Toolkit Samples` atau dari `Asset Store`. Selain itu, pengguna juga bisa menggunakan model tangan khusus yang diunduh dari internet atau dibuat sendiri agar tampilan lebih realistis dan sesuai kebutuhan.

Gambar 3.30. Struktur hierarki XR Rig dengan penambahan model tangan sebagai anak dari kontroler kiri dan kanan.

Model tangan ini akan mengikuti rotasi dan posisi kontroler secara real-time, sehingga menciptakan kesan imersif bagi pengguna. Model juga bisa dihubungkan dengan animasi tambahan (misalnya animasi jari menggenggam) menggunakan sistem input yang telah dikonfigurasi pada tahap sebelumnya, untuk menunjukkan status grab atau pose interaksi lainnya.

Dengan demikian, pengalaman pengguna akan terasa lebih natural dan intuitif, karena mereka dapat melihat representasi visual dari gerakan tangan mereka di dunia virtual.

Untuk meningkatkan realisme interaksi dalam simulasi VR, disertakan juga animasi dinamis pada tangan virtual, seperti menggenggam atau menekan. Animasi ini memungkinkan model tangan untuk merespons input pengguna secara real-time, sehingga menciptakan pengalaman visual yang lebih natural saat berinteraksi dengan objek di dunia virtual.

Implementasinya dilakukan dengan menambahkan skrip bernama ScHandAnimator ke masing-masing kontroler. Skrip ini membaca nilai dari input *trigger* (pinch) dan *grip* (genggaman) pada kontroler VR menggunakan InputActionProperty, lalu meneruskan nilai tersebut ke parameter animator.

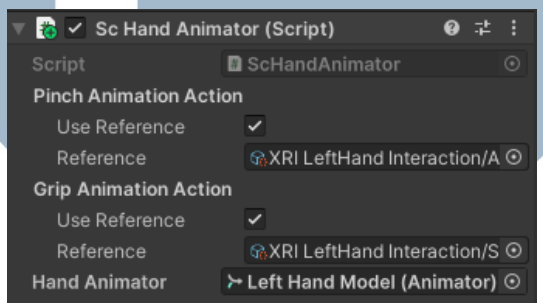
Animator kemudian mengatur transisi animasi berdasarkan nilai input tersebut.

```
float triggerValue = pinchAnimationAction.action.ReadValue<float
    >();
handAnimator.SetFloat("Trigger", triggerValue);

float gripValue = gripAnimationAction.action.ReadValue<float>();
handAnimator.SetFloat("Grip", gripValue);
```

Kode 3.8: Logika animasi tangan dari input VR controller

Untuk dapat bekerja dengan baik, referensi Animator pada model tangan juga harus dikaitkan, dan parameter animator seperti Trigger dan Grip perlu sudah tersedia di dalam animasi model tangan. Gambar 3.31 menunjukkan konfigurasi skrip animator yang telah dikaitkan dengan model tangan kiri:



Gambar 3.31. Konfigurasi skrip animasi tangan untuk mengontrol pose berdasarkan input controller.

Dengan begitu, pengguna akan melihat tangan mereka menggenggam ketika tombol grip ditekan, dan mencubit saat trigger ditekan, yang mencerminkan gerakan sebenarnya saat berinteraksi dengan objek fisik. Hal ini penting terutama pada simulasi grab-and-release serta manipulasi komponen mesin, karena memberikan feedback visual yang presisi dan intuitif.

Selain konfigurasi model tangan dan sistem animasinya, perlu dikonfigurasi juga deteksi area interaksi menggunakan *collider*. *Collider* ini berfungsi sebagai volume deteksi di sekitar tangan virtual, yang memungkinkan objek di sekitarnya terdeteksi saat pengguna melakukan aksi seperti grab atau touch.

Implementasinya dilakukan dengan menambahkan `Collider` (umumnya berupa `Sphere Collider` atau `Box Collider`) pada GameObject tangan—baik tangan kiri maupun kanan. *Collider* tersebut tidak harus terlihat secara visual, namun menjadi komponen penting dalam sistem deteksi proximity objek interaktif.

Gambar 3.32. Contoh penambahan `Sphere Collider` pada model tangan untuk mendeteksi objek di sekitarnya.

Collider ini biasanya diatur sebagai `Trigger` agar tidak menyebabkan benturan fisik, namun tetap dapat digunakan untuk mendeteksi apakah objek berada dalam jangkauan. Ketika pengguna menekan tombol grab (biasanya grip atau trigger), sistem akan mencari objek yang berada dalam area *collider* dan memiliki komponen interaktif seperti `XRGrabInteractable`.

Dengan demikian, pengguna tidak dapat melakukan grab terhadap objek yang berada terlalu jauh dari tangan, dan hanya objek yang benar-benar berada dalam jangkauan yang bisa dipilih. Hal ini menciptakan pengalaman interaksi yang lebih imersif dan konsisten secara fisik dalam dunia virtual.

C Tahap 3.3: Setup Machine Parts (Import, Colliders, Physics)

Setelah sistem interaksi dasar seperti XR Rig dan input telah dikonfigurasi, langkah berikutnya adalah melakukan pengaturan terhadap aset-aset 3D dari bagian mesin yang akan digunakan dalam simulasi. Aset-aset ini dikembangkan oleh tim desainer eksternal dan diberikan dalam format 3D yang siap diimpor ke dalam Unity. Proses impor dilakukan secara manual melalui metode *drag-and-drop* ke dalam folder `Assets` pada Unity Project, dan kemudian ditempatkan ke dalam `Scene Hierarchy` sesuai dengan struktur perakitan yang diinginkan.

Setiap bagian dari mesin dikonfigurasi secara individual untuk memastikan dapat berinteraksi secara realistis dalam dunia virtual. Komponen-komponen penting yang ditambahkan atau disesuaikan pada tiap objek meliputi:

1. Collider: Penggunaan `Box Collider`, `Capsule Collider`, atau `Mesh Collider` ditentukan berdasarkan bentuk fisik masing-masing objek. Komponen ini memungkinkan deteksi tumbukan dan interaksi dengan tangan virtual pengguna.

2. Rigidbody: Ditambahkan untuk memungkinkan interaksi fisik seperti jatuh

akibat gravitasi, tergeser, atau tertarik saat di-grab. Parameter seperti massa, drag, dan use gravity disesuaikan agar menyerupai perilaku objek nyata.

3. XR Grab Interactable: Komponen ini memungkinkan objek untuk bisa diambil, dilepaskan, dan diletakkan kembali menggunakan kontroler. Interaksi seperti hover dan grab secara otomatis ditangani oleh komponen ini melalui XR Interaction Toolkit.

4. XR Socket Interactor: Pada objek mesin yang menjadi tempat pemasangan bagian lain, socket disiapkan sebagai tempat docking untuk menerima komponen yang di-grab oleh pengguna. Socket akan mencocokkan bentuk dan tag dari objek interaktif yang mendekat.

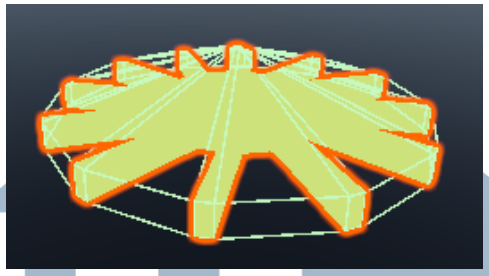
5. Layer dan Tag: Seluruh objek interaktif diberikan `Layer` khusus seperti `Parts`, `Grabbable`, atau `Socket` agar dapat difilter dalam sistem event Unity. Penamaan `Tag` juga dilakukan untuk mendukung skrip validasi dan interaksi khusus berdasarkan jenis objek.

Pengaturan-pengaturan ini bertujuan untuk mensimulasikan pengalaman kerja nyata secara mendekati kondisi sebenarnya, baik dalam hal fisika maupun alur kerja perakitan. Konfigurasi ini juga penting agar seluruh sistem interaksi berbasis XR dapat mengenali objek dengan benar dan memfasilitasi alur prosedural yang telah ditentukan sebelumnya.

Selain penempatan dan strukturisasi objek 3D ke dalam scene Unity, setiap bagian mesin perlu diberi komponen fisika untuk memastikan dapat berinteraksi dengan sistem XR. Salah satu tahap paling krusial adalah pemberian Collider.

Sebagian besar objek dalam proyek ini memiliki bentuk geometri yang kompleks (tidak sekadar kubus atau silinder), sehingga penggunaan komponen `Mesh Collider` menjadi pilihan utama. Dibandingkan `Box Collider` atau `Capsule Collider`, `Mesh Collider` mampu mengikuti kontur objek secara akurat, memungkinkan interaksi fisik yang lebih presisi dan realistis.

Namun karena `Mesh Collider` tidak dapat digunakan secara default untuk interaksi fisika dinamis seperti grabbing atau pengaplikasian `Rigidbody`, maka perlu dilakukan aktivasi opsi *Convex*. Dengan mencentang properti `Convex`, Unity akan mengkonversi collider menjadi bentuk konveks yang aman untuk digunakan dalam interaksi dinamis, sekaligus kompatibel dengan sistem XR Interaction Toolkit.



Gambar 3.33. Contoh objek mesin dengan `Mesh Collider` dan parameter `Convex` aktif.

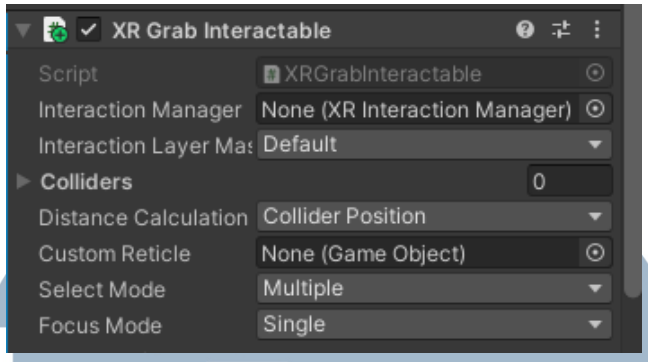
Setelah menambahkan `Mesh Collider`, langkah selanjutnya adalah menyisipkan komponen `Rigidbody` ke setiap objek mesin yang dapat digerakkan atau berinteraksi secara fisik. Komponen `Rigidbody` memungkinkan objek untuk merespons gaya seperti gravitasi, tumbukan, serta penggerakan oleh pengguna (misalnya saat objek di-grab atau dijatuhkan).

Tanpa `Rigidbody`, objek tidak akan berpartisipasi dalam sistem fisika Unity seperti dipengaruhi gravitasi, sehingga tidak dapat terdeteksi sebagai bagian dari interaksi dinamis seperti grab, snap, atau reaksi tumbukan. Dengan kata lain, objek tanpa `Rigidbody` akan bersifat statis dan tidak bisa didorong atau dijatuhkan oleh pengguna VR.

Setelah objek memiliki collider dan komponen fisika (Rigidbody), langkah selanjutnya adalah memungkinkan objek tersebut untuk bisa diinteraksi seperti diambil (grab) oleh pengguna. Hal ini dilakukan dengan menambahkan komponen XR Grab Interactable pada setiap bagian mesin yang bisa diinteraksikan.

Komponen ini memungkinkan objek untuk dikenali oleh sistem XR sebagai objek interaktif. Dalam implementasi proyek ini, beberapa pengaturan khusus diterapkan untuk meningkatkan realisme dan fleksibilitas interaksi:

1. Select Mode diatur ke `Multiple`, agar objek dapat di-grab menggunakan kedua tangan secara bersamaan. Hal ini berguna terutama untuk objek besar atau berat.
2. Use Dynamic Attach dicentang, agar posisi dan orientasi objek akan mengikuti posisi tangan pengguna secara natural saat diambil. Ini menciptakan pengalaman interaksi yang lebih realistis dan intuitif.



Gambar 3.34. Pengaturan komponen XR Grab Interactable pada objek mesin. Select Mode diset ke Multiple.

Setelah pengaturan Select Mode diterapkan pada komponen XR Grab Interactable, diaktifkan juga opsi Dynamic Attach. Pengaturan ini memungkinkan objek yang di-grab menyesuaikan posisi dan rotasinya secara dinamis terhadap titik interaksi pengguna, sehingga menghasilkan pengalaman interaksi yang lebih natural dan realistis. Fitur ini sangat berguna terutama untuk simulasi kerja manual seperti pemasangan atau pembongkaran komponen mesin, karena memberikan efek seolah-olah pengguna benar-benar memegang dan mengarahkan objek secara langsung. Tampilan antarmuka pengaturan ini dapat dilihat pada Gambar berikut.

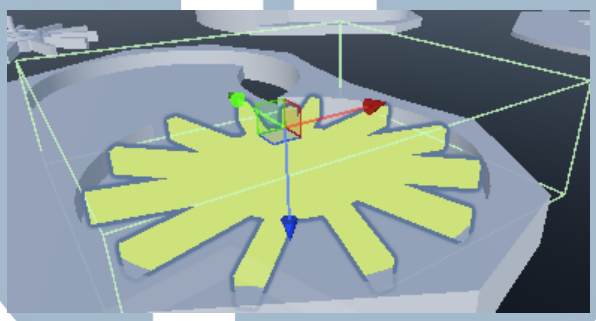
Gambar 3.35. Pengaturan komponen XR Grab Interactable pada objek mesin. Dynamic attach diaktifkan.

Dengan konfigurasi ini, seluruh bagian mesin yang relevan kini dapat diambil dan diposisikan ulang oleh pengguna menggunakan kontroler VR. Pengaturan ini merupakan dasar dari seluruh sistem interaksi manual dalam aplikasi, termasuk proses perakitan dan pembongkaran bagian mesin secara prosedural.

Langkah selanjutnya dalam proses setup interaksi adalah menambahkan

XR Socket Interactor pada lokasi-lokasi tertentu di dalam scene sebagai titik atau tempat penempatan objek. Komponen ini berfungsi sebagai socket atau ”tempat tujuan” bagi objek yang dapat di-grab oleh pengguna. Saat objek dibawa mendekati socket, sistem akan secara otomatis mendeteksi kompatibilitas dan memungkinkan pengguna untuk melepaskan objek ke dalam socket dengan presisi tinggi.

Socket Interactor sangat penting dalam simulasi prosedural seperti perakitan mesin, karena memastikan objek ditempatkan di lokasi yang benar dan terkunci secara visual maupun fungsional.



Gambar 3.36. Socket Interactor dengan Box Collider untuk mendeteksi objek yang akan ditempatkan. Socket ini menjadi titik referensi peletakan bagian mesin.

Dengan konfigurasi socket ini, sistem interaksi dapat mengenali bahwa suatu objek telah ditempatkan dengan benar di lokasi yang ditentukan. Selain itu, socket juga bisa digunakan untuk memicu event atau perubahan status objek dalam skrip, seperti aktivasi highlight objek berikutnya atau validasi tahap kerja.

Selain komponen-komponen diatas, pengaturan *Tag* dan *Layer Mask* juga merupakan aspek penting dalam pengelolaan interaksi objek-objek mesin di Unity. Fitur ini memungkinkan pemisahan logika dan kontrol interaksi antar berbagai jenis objek dalam simulasi.

Tag digunakan untuk mengidentifikasi jenis objek tertentu, seperti ’Parts’, ’Socket’, ’Tool’, atau ’Screw’. Penamaan tag ini kemudian digunakan dalam skrip untuk melakukan validasi atau pemrosesan kondisi tertentu, misalnya mendeteksi apakah objek yang di-grab merupakan komponen yang sesuai.

Layer Mask, di sisi lain, digunakan untuk mengatur interaksi antar layer melalui sistem fisika dan XR Interaction Toolkit. Dengan menetapkan layer seperti ’Grabable’, ’SocketOnly’, atau ’Screw’, dapat ditentukan objek mana yang boleh di-interaksikan atau dikenali oleh komponen seperti `XR Socket Interactor` dan `XR Ray Interactor`.

Dengan menggunakan kombinasi tag dan layer ini, dapat:

1. Menghindari interaksi antar objek yang tidak relevan.
2. Mengelompokkan logika tertentu dalam skrip tanpa memeriksa nama objek secara manual.
3. Menyederhanakan kontrol highlight, validasi urutan, dan sistem kombinasi objek.

D Tahap 3.4: Code Scripts

Pengembangan skrip interaksi dilakukan menggunakan bahasa pemrograman C#, yang diintegrasikan dalam Unity melalui komponen-komponen berbasis XR Interaction Toolkit. Skrip dirancang secara modular untuk menangani berbagai skenario interaksi pengguna dalam simulasi, mulai dari proses perakitan hingga validasi akhir.

Fitur-fitur utama yang diimplementasikan melalui skrip mencakup:

1. Sistem grab dan release dengan validasi urutan komponen.
2. Pemberian highlight pada objek yang sedang di-hover untuk umpan balik visual.
3. Logika interaksi berbasis event seperti `selectEntered`, `hoverEntered`, dan `selectExited`.
4. Sistem validasi yang memeriksa apakah seluruh komponen telah berhasil dirakit sebelum simulasi dilanjutkan.

Setiap jenis interaksi dalam simulasi ini dibangun dengan mempertimbangkan skenario kerja nyata yang terjadi di lapangan, seperti proses perakitan alat, pemasangan bagian modular, hingga pembongkaran komponen yang dikunci menggunakan skrup. Dalam pengembangannya, sistem menjadi tiga jenis interaksi utama, yaitu interaksi dasar (normal), interaksi gabungan (combine), dan interaksi dengan skrup (screw). Masing-masing jenis memiliki kompleksitas yang berbeda dan ditangani melalui pendekatan skrip yang spesifik.

Jenis interaksi pertama adalah *normal disassembly*, di mana pengguna hanya perlu melepaskan bagian mesin dari socket yang telah ditentukan tanpa ada proses tambahan seperti rotasi, pukulan, atau penggabungan ulang. Proses ini merepresentasikan skenario umum dalam pembongkaran perangkat industri

yang tidak membutuhkan alat bantu khusus. Ketika objek berhasil dilepaskan, sistem akan memberikan umpan balik visual berupa perubahan material untuk menunjukkan bahwa objek telah tidak aktif, serta mengaktifkan kembali collider untuk memungkinkan tumbukan fisik dengan objek lain di lingkungan VR.

Sistem kemudian memanggil fungsi coroutine yang berfungsi untuk menonaktifkan socket saat ini dan mengaktifkan objek serta interaksi berikutnya, termasuk pengaturan ulang event listener dan penyorotan (highlight) terhadap komponen yang akan dimanipulasi berikutnya. Semua logika ini ditangani oleh skrip `ScDiesDisassembly.cs`, dengan fungsi utama seperti ditunjukkan dalam potongan Kode 3.9:

```
public void OnDetached(SelectExitEventArgs args)
{
    if (args.interactorObject is XRSocketInteractor xrSocket)
    {
        gameObject.tag = "Parts";

        foreach (GameObject obj in currentHighlight)
        {
            Renderer renderer = obj.GetComponent<Renderer>();
            if (renderer != null)
            {
                renderer.material = defaultMaterial;
            }
        }

        Collider currentCollider = GetComponent<Collider>();
        GameObject[] sockets = GameObject.FindGameObjectsWithTag("Socket");

        foreach (GameObject socket in sockets)
        {
            Collider[] colliders = socket.GetComponentsInChildren<Collider>();
            foreach (Collider col in colliders)
            {
                Physics.IgnoreCollision(currentCollider, col, false);
            }
        }

        StartCoroutine(ProcessDisassembly());
```

```
    }
}
```

Kode 3.9: Disassembly object

Dalam coroutine `ProcessDisassembly()`, sistem mengatur alur logika berikutnya setelah bagian berhasil dilepas. Fungsi ini mematikan socket terkait, menyiapkan objek-objek selanjutnya agar dapat di-grab, serta melakukan registrasi ulang terhadap event interaksi seperti hover dan detach, agar alur simulasi tetap dapat berjalan secara berurutan.

```
ScDiesDisassembly disassemblyScript = obj.GetComponent<
    ScDiesDisassembly>();
if (disassemblyScript != null)
{
    objGrab.selectEntered.AddListener(disassemblyScript.OnAttached
    );
    objGrab.selectExited.AddListener(disassemblyScript.OnDetached)
    ;
}

ScHoverObject hoverScript = obj.GetComponent<ScHoverObject>();
if (hoverScript != null)
{
    objGrab.hoverEntered.AddListener(hoverScript.OnHoverEnter);
    objGrab.hoverExited.AddListener(hoverScript.OnHoverExit);
    objGrab.selectExited.AddListener(hoverScript.OnDetach);
}
```

Kode 3.10: Register ulang event

Jenis interaksi berikutnya adalah *normal assembly*, yaitu proses perakitan bagian mesin ke dalam socket tanpa perlu dilakukan penggabungan objek secara fisik. Interaksi ini sangat sesuai untuk skenario di mana objek cukup diposisikan secara presisi namun tidak memerlukan gerakan setelah pemasangan, seperti tutup panel, bracket, atau pengunci statis.

Skrip untuk assembly normal serupa dengan `ScFOMAssembly.cs` namun tanpa bagian `Combine()`. Setelah objek ditempatkan, sistem akan menonaktifkan placeholder, mengganti material untuk menghapus efek highlight, dan memicu aktivasi serta penyorotan objek berikutnya.

```
private void OnAttached(SelectEnterEventArgs args)
{
    if (args.interactorObject is XRSocketInteractor xrSocket)
```

```
    {
        gameObject.tag = "Socket";

        foreach (GameObject obj in currentHighlight)
        {
            Renderer renderer = obj.GetComponent<Renderer>();
            if (renderer != null)
            {
                renderer.material = defaultMaterial;
            }
        }

        foreach (GameObject obj in nextHighlight)
        {
            Renderer renderer = obj.GetComponent<Renderer>();
            if (renderer != null)
            {
                renderer.material = highlightMaterial;
            }
        }

        foreach (GameObject obj in objectToggle)
        {
            obj?.SetActive(true);
        }
    }
}
```

Kode 3.11: Assembly object

Jenis selanjutnya adalah *combine assembly*, yang digunakan untuk objek-objek yang perlu digabungkan menjadi satu kesatuan setelah ditempatkan, agar bisa digerakkan sebagai unit utuh. Proses combine ini dilakukan dengan membuat GameObject baru bernama `Combined` yang menjadi parent dari seluruh bagian yang digabung. Komponen fisika seperti `Rigidbody` dan interaksi seperti `XRGrabInteractable` kemudian ditambahkan pada objek gabungan ini. Implementasi teknik ini terlihat dalam skrip `ScFOMAssembly.cs`:

```
private IEnumerator Combine()
{
    yield return new WaitForSeconds(1f);
    GameObject combinedObject = new GameObject("Combined");

    foreach (GameObject part in allParts)
```

```
    {
        XRGrabInteractable grab = part.GetComponent<
    XRGrabInteractable>();
        if (grab != null) Destroy(grab);

        Rigidbody rb = part.GetComponent<Rigidbody>();
        if (rb != null) Destroy(rb);

        part.transform.SetParent(combinedObject.transform);
    }

    Rigidbody combinedRb = combinedObject.AddComponent<Rigidbody
    >();
    XRGrabInteractable combinedGrab = combinedObject.AddComponent<
    XRGrabInteractable>();
    combinedGrab.useDynamicAttach = true;
}
```

Kode 3.12: Combine method

Jenis interaksi selanjutnya adalah *combine disassembly*, yaitu proses pembongkaran terhadap objek yang sebelumnya telah digabungkan (combined) menjadi satu kesatuan rigid. Interaksi ini mencerminkan skenario teknis di mana beberapa komponen telah dipasang menjadi satu kesatuan fisik dan perlu dibongkar ulang agar dapat diinteraksikan kembali secara individu. Implementasi logika ini diatur melalui skrip `ScFOMDisassembly.cs`.

Ketika pengguna menarik atau melepaskan objek dari socket, fungsi coroutine `Uncombine()` akan dijalankan. Fungsi ini bertugas untuk membongkar objek gabungan dengan cara menghapus komponen `Rigidbody` dan `XRGrabInteractable` dari objek `Combined`, serta mengatur kembali hierarki objek agar masing-masing komponen dapat di-grab secara individu. Ini penting untuk memastikan bahwa setelah proses disassembly, pengguna dapat melanjutkan interaksi terhadap bagian-bagian mesin satu per satu, sesuai urutan yang diharapkan.

```
private IEnumerator Uncombine()
{
    yield return new WaitForSeconds(1f);

    GameObject combinedObject = GameObject.Find("Combined");

    if (combinedObject == null)
```

```
    {
        combinedObject = new GameObject("Combined");
    }

    XRGrabInteractable existingGrab = combinedObject.GetComponent<
    XRGrabInteractable>();
    if (existingGrab != null)
    {
        Destroy(existingGrab);
    }

    Rigidbody existingRb = combinedObject.GetComponent<Rigidbody
    >();
    if (existingRb != null)
    {
        Destroy(existingRb);
    }
```

Kode 3.13: Uncombine method

Selain menghapus properti gabungan, skrip ini juga melakukan aktivasi kembali pada objek-objek yang akan diinteraksikan berikutnya. Masing-masing objek akan dipastikan memiliki komponen `Rigidbody` untuk simulasi fisika dan `XRGrabInteractable` untuk memungkinkan interaksi grab. Event listener juga diatur ulang agar objek dapat merespon `hover` dan `detach`, serta fungsi interaksi yang relevan seperti berikut:

```
foreach (GameObject obj in nextHighlight)
{
    obj.transform.SetParent(null);
    Renderer renderer = obj.GetComponent<Renderer>();
    if (renderer != null)
    {
        renderer.material = highlightMaterial;
    }

    Rigidbody objRb = obj.GetComponent<Rigidbody>();
    if (objRb == null)
    {
        objRb = obj.AddComponent<Rigidbody>();
    }

    XRGrabInteractable objGrab = obj.GetComponent<
    XRGrabInteractable>();
```

```
    if (objGrab == null)
    {
        objGrab = obj.AddComponent<XRGrabInteractable>();
        objGrab.useDynamicAttach = true;
    }

    ScFOMDisassembly disassemblyScript = obj.GetComponent<
    ScFOMDisassembly>();
    if (disassemblyScript != null)
    {
        objGrab.selectEntered.AddListener(disassemblyScript.
    OnAttached);
        objGrab.selectExited.AddListener(disassemblyScript.
    OnDetached);
    }
}
```

Kode 3.14: Aktivasi objek dan register ulang event

Terakhir, socket-socket sebelumnya dinonaktifkan dan komponen baru dari objek hasil pembongkaran diaktifkan untuk memungkinkan kelanjutan simulasi secara sekuensial. Proses ini menjamin bahwa alur kerja tetap sesuai dengan urutan operasional di dunia nyata, sekaligus memastikan bahwa interaksi VR tetap terasa natural dan modular.

Jenis selanjutnya, yang paling kompleks, adalah *screw disassembly*. Jenis ini merepresentasikan situasi teknis di mana bagian mesin dikunci menggunakan skrup dan harus dilepaskan dengan alat bantu seperti obeng (wrench). Pengguna perlu memutar atau memukul skrup beberapa kali hingga sistem menganggapnya telah berhasil dilepas. Logika ini diatur dalam skrip `ScScrewRemove.cs`, yang mencakup perpindahan posisi dan rotasi setiap kali fungsi `HitNail()` dipanggil.

```
public void HitNail()
{
    if (!IsFullyHammered)
    {
        hammerCount++;
        transform.localPosition += movementDirection.normalized *
    movementPerHit;
        transform.Rotate(rotationAxis, rotationPerHit, Space.Self)
    ;
        source.PlayOneShot(hit);
        OnHammered?.Invoke();
    }
```

```
}
```

Kode 3.15: Method hit nail disassembly

Untuk memastikan bahwa semua skrup telah dilepas sebelum bagian mesin bisa dibongkar, digunakan skrip tambahan `ScCheckScrewRemove.cs`, yang memverifikasi status dari setiap skrup. Jika seluruh skrup telah dalam keadaan `IsFullyHammered` dan `IsDetached`, maka bagian mesin bisa dibongkar, objek lanjutan diaktifkan, dan interaksi berikutnya tersedia.

```
private void CheckAllScrews()
{
    foreach (var screw in screws)
    {
        if (!screw.IsFullyHammered || !screw.IsDetached)
        {
            return;
        }
    }

    disassemblyScript.enabled = true;
}
```

Kode 3.16: Method check screws disassembly

Proses deteksi interaksi alat dengan skrup dilakukan melalui skrip `WrenchRemove.cs`, yang menggunakan metode `OnTriggerEnter()` untuk memicu rotasi skrup saat alat bersentuhan dengannya. Fungsi ini mencegah pemicuan ganda dengan menggunakan cooldown time dan kontrol boolean.

```
private void OnTriggerEnter(Collider other)
{
    if (!canTrigger) return;

    if (other.CompareTag("Nail"))
    {
        ScScrewRemove nail = other.GetComponent<ScScrewRemove>();
        if (nail != null)
        {
            nail.HitNail();
        }

        StartCoroutine(ResetTriggerCooldown());
    }
```

```
}
```

Kode 3.17: Method on trigger

Jenis selanjutnya adalah *screw assembly*, yaitu jenis interaksi yang membutuhkan pengguna untuk memasang skrup secara bertahap menggunakan alat bantu seperti wrench. Skrup harus dipukul atau diputar beberapa kali sampai sistem mendeteksi bahwa ia telah terpasang penuh. Interaksi ini mencerminkan prosedur dunia nyata dalam perakitan industri yang memerlukan alat bantu mekanis.

Proses ini ditangani oleh skrip `ScScrewAttach.cs`, yang melacak jumlah pukulan melalui variabel `hammerCount`. Setiap kali fungsi `HitNail()` dipanggil, posisi dan rotasi skrup akan berubah untuk mensimulasikan proses pemasangan.

```
public void HitNail()
{
    if (!IsFullyHammered)
    {
        hammerCount++;
        transform.localPosition += movementDirection.normalized *
    movementPerHit;
        transform.Rotate(rotationAxis, rotationPerHit, Space.Self)
    ;
        source.PlayOneShot(hit);
        OnHammered?.Invoke();
    }
}
```

Kode 3.18: Method hit nail assembly

Setelah semua skrup pada suatu bagian telah terpasang penuh, sistem akan mengaktifkan objek atau tahap simulasi berikutnya. Logika ini diatur dalam skrip `ScScrewCheckerYDivider.cs`, yang memverifikasi seluruh skrup dan mengatur material highlight secara visual.

```
private void CheckAllScrews()
{
    foreach (var screw in screws)
    {
        if (!screw.IsFullyHammered)
            return;
    }

    foreach (var obj in objectsToActivate)
    {
```

```
        obj?.SetActive(true);
    }

    foreach (var obj in objectsCurrent)
    {
        Renderer renderer = obj.GetComponent<Renderer>();
        if (renderer != null)
            renderer.material = defaultMaterial;
    }

    foreach (var obj in objectsNext)
    {
        Renderer renderer = obj.GetComponent<Renderer>();
        if (renderer != null)
            renderer.material = highlightMaterial;
    }
}
```

Kode 3.19: Method check screws assembly

Interaksi jenis ini memungkinkan pengguna untuk merasakan sensasi prosedural yang lebih realistis dalam pemasangan, sekaligus menambah kompleksitas yang mencerminkan kondisi nyata di lapangan. Dengan bantuan efek suara dan perubahan visual, pengguna mendapatkan umpan balik menyeluruh terhadap proses pemasangan skrup secara bertahap.

Untuk memperkuat pengalaman pengguna, sistem ini juga dilengkapi dengan skrip hover bernama `ScHoverNew.cs`, yang mengatur perubahan material secara visual saat objek di-hover atau dilepas dari socket. Ini meningkatkan umpan balik visual dan membantu pengguna memahami objek mana yang aktif dalam tahap interaksi.

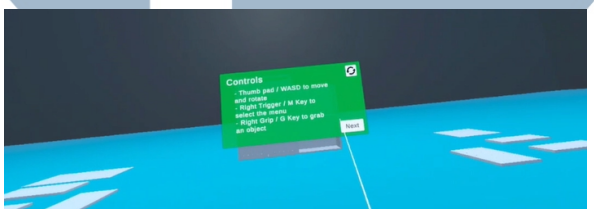
```
public void OnHoverEnter(HoverEnterEventArgs args)
{
    foreach (GameObject obj in targetObjects)
    {
        Renderer rend = obj?.GetComponent<Renderer>();
        if (rend != null)
        {
            rend.material = hoverMaterial;
        }
    }
}
```

Kode 3.20: Method hover

Keseluruhan sistem interaksi ini membentuk fondasi utama dari simulasi pelatihan VR yang dikembangkan, menggabungkan elemen logika prosedural, umpan balik visual, serta modularitas yang memungkinkan pengembangan tahap-tahap simulasi yang lebih kompleks ke depannya.

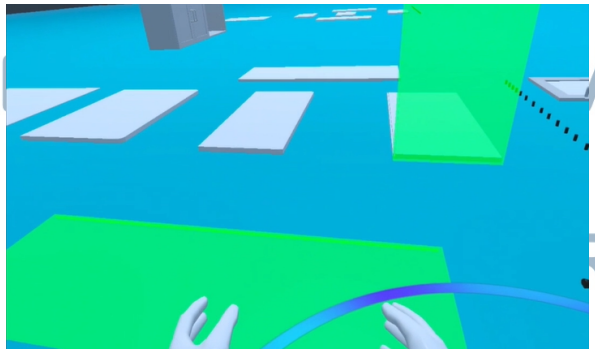
E Tahap 3.5: Contoh Hasil Implementasi

Pada bagian ini akan ditunjukkan hasil implementasi dari pengembangan aplikasi Virtual Reality (VR) yang telah dilakukan selama masa magang. Implementasi ini merupakan perwujudan dari seluruh tahapan pengembangan sebelumnya, mulai dari perancangan konsep, pembuatan alur kerja, konfigurasi lingkungan Unity, hingga integrasi interaksi menggunakan XR Interaction Toolkit. Simulasi yang dihasilkan dirancang untuk mensubstitusi proses pelatihan manual dengan pendekatan yang lebih aman, efisien, dan interaktif berbasis teknologi immersive.



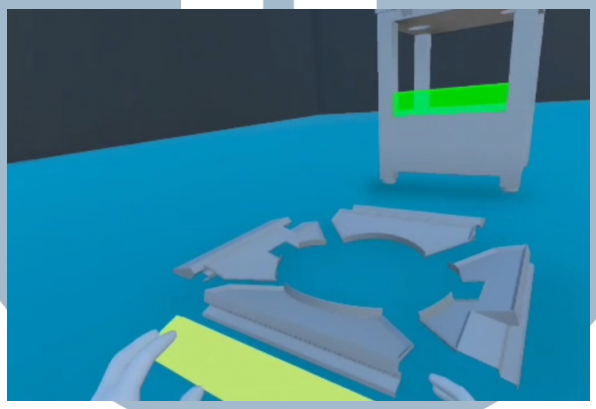
Gambar 3.37. Contoh hasil pembuatan Tutorial Menu PoC

Gambar 3.37 menunjukkan contoh tampilan awal dari menu tutorial yang dikembangkan dalam tahap *Proof of Concept* (PoC). Menu ini berfungsi sebagai panduan interaktif yang memberikan instruksi dasar kepada pengguna mengenai pengerjaan simulasi dan kontrol yang dapat digunakan selama simulasi.



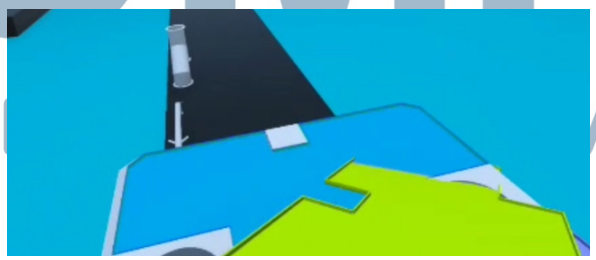
Gambar 3.38. Contoh hasil pembuatan PoC

Gambar 3.38 menunjukkan tampilan dari *Proof of Concept* (PoC) yang dibangun sebagai fondasi untuk menguji dan memvalidasi berbagai mekanisme interaksi dasar. Dalam tahap ini, pengguna dapat mencoba fungsi dasar seperti mengambil dan melepaskan objek (*grab* dan *release*), meletakkan objek pada *socket* yang sesuai, hingga melakukan tindakan spesifik seperti pemukulan paku ke dalam lubang. Meskipun PoC ini masih menggunakan aset sementara (*placeholder*), tahap ini sangat penting untuk memastikan sistem interaksi bekerja stabil dan logis sebelum masuk ke tahap implementasi final.



Gambar 3.39. Contoh hasil implementasi outer mesin

Gambar 3.39 memperlihatkan implementasi simulasi pada bagian luar mesin, dengan aset 3D yang disesuaikan secara akurat dengan desain asli di industri. Pengguna dapat melakukan perakitan dan pembongkaran sesuai prosedur standar melalui interaksi berbasis socket. Sistem validasi memberikan umpan balik terhadap langkah yang benar atau salah, sehingga mendukung pembelajaran yang mandiri dan terstruktur.



Gambar 3.40. Contoh hasil implementasi upper mesin

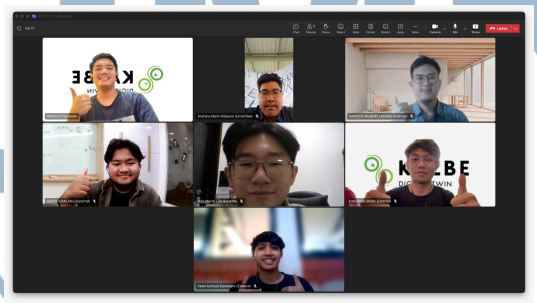
Gambar 3.40 menunjukkan bagian atas mesin yang memiliki kompleksitas tinggi karena terdiri dari komponen kecil dan padat. Interaksi pada bagian ini memerlukan ketelitian serta didukung oleh mekanisme penguncian otomatis dalam

simulasi. Bagian ini menjadi fokus penting karena sering diabaikan dalam pelatihan konvensional, meskipun berperan krusial dalam prosedur kerja. Urutan interaksi disusun berdasarkan hasil observasi lapangan dan masukan teknis dari engineer serta operator berpengalaman.

3.5.7 Tahap 4: Testing

Pada tahap ini, proses pengujian aplikasi dilakukan secara terbatas mengingat proyek masih berada pada fase pengembangan dan belum mencapai versi akhir yang siap untuk diuji secara menyeluruh. Meskipun belum dilakukan user testing secara formal, kegiatan pengujian tetap dilaksanakan secara bertahap dan iteratif melalui sesi evaluasi internal dalam lingkup tim proyek.

Pengujian dilakukan secara informal dalam bentuk sesi weekly meeting yang dilaksanakan setiap minggu bersama tim Project Management (PM), developer, serta desainer. Dalam sesi tersebut, setiap kemajuan dari pengembangan aplikasi—termasuk modul interaksi, alur simulasi, dan tampilan user interface—dipresentasikan dan didemonstrasikan secara langsung apabila bagian tersebut telah fungsional. Dari sesi tersebut, tim akan memberikan umpan balik teknis dan non-teknis untuk menjadi bahan perbaikan pada sprint selanjutnya. Proses ini sekaligus berperan sebagai validasi fungsional awal terhadap komponen-komponen yang telah selesai dikembangkan, meskipun belum dilakukan pengujian secara formal menggunakan protokol UX atau pengukuran performa.



Gambar 3.41. Weekly meeting team internal.

Sebagai bagian dari dokumentasi teknis dan persiapan untuk pengujian formal di masa mendatang, telah disusun daftar *test case* untuk tiap fitur utama dalam simulasi. *Test case* ini berfungsi sebagai referensi saat proses quality assurance dilakukan secara lebih menyeluruh. Struktur penomoran test case menggunakan format TC-XX untuk memudahkan pelacakan dan dokumentasi.

Tabel 3.5. Dokumentasi test case

Test Case	Deskripsi
TC-01	Verify that the application launches successfully without crashing.
TC-02	Check if the VR environment is initialized properly, including controllers and headset tracking.
TC-03	Validate the loading screen displays progress or relevant information during initialization.
TC-04	Verify that the object loads correctly in the VR environment.
TC-05	Ensure users can select and highlight individual components of the object.
TC-06	Test the ability to grab, move, and rotate components of the object using VR controllers.
TC-07	Validate the precision of interaction (e.g., snapping components into place).
TC-08	Check if incorrect actions trigger appropriate feedback (e.g., vibration, audio cues, or error messages).
TC-09	Verify that components can be assembled in the correct order.
TC-10	Ensure that a step-by-step guide or hints are available for assembly (if applicable).
TC-11	Check if misaligned assembly attempts are blocked or corrected.
TC-12	Test the functionality of tools (virtual screwdrivers, wrenches, etc.) if included.
TC-13	Validate the completion notification or indication after the assembly process.
TC-14	Verify that components can be disassembled in the correct order.
TC-15	Ensure that tools (if required) are functional and accessible for disassembly.
TC-16	Check if users can freely detach components without breaking the process flow.
TC-17	Validate that warnings or instructions appear if the user performs incorrect disassembly actions.
TC-18	Verify that the VR controls (buttons, joysticks, gestures) function as expected.
	Lanjut pada halaman berikutnya

Tabel 3.5. Dokumentasi test case (versi) (Lanjutan)

Test Case	Deskripsi
TC-19	Ensure menus, toolbars, or HUD elements are accessible and usable in VR.
TC-20	Validate that tooltips or instructional text is legible and positioned correctly in the VR space.
TC-21	Check if the VR environment is free from graphical glitches.
TC-22	Ensure that lighting, shadows, and textures are rendered properly.
TC-23	Validate audio feedback for actions like picking up, assembling, or disassembling components.
TC-24	Test if environmental sounds or background music are immersive and non-intrusive.
TC-25	Verify that the application maintains a stable frame rate during use.
TC-26	Check if memory and GPU usage remain within acceptable limits.
TC-27	Test for latency in interactions (e.g., grabbing or snapping components).
TC-28	Ensure that the application gracefully handles scenarios like dropped connections or headset disconnection.
TC-29	Verify that error messages are clear and guide the user to resolve issues.
TC-30	Check if invalid or unsupported actions do not crash the application.
TC-31	Test the clarity of instructions provided for assembly and disassembly.
TC-32	Verify the ease of accessing the restart or reset function during a task.
TC-33	Ensure users can exit the application smoothly and save progress if applicable.
TC-34	Validate that text and UI elements are readable for users of varying heights.
TC-35	Ensure compatibility with left-handed and right-handed VR controller configurations.
TC-36	Test if colorblind-friendly options (e.g., high contrast modes) are available.

Test case di atas akan diperbarui dan dilengkapi sesuai dengan perkembangan fitur dan kebutuhan pengujian pada tahap selanjutnya. Struktur ini juga dapat dikembangkan menjadi bagian dari dokumentasi pengujian resmi ketika aplikasi memasuki fase deployment dan validasi akhir.

Secara keseluruhan, meskipun pengujian yang dilakukan belum bersifat menyeluruh, pendekatan progressive testing melalui evaluasi mingguan telah memberikan dasar yang kuat untuk memastikan arah pengembangan tetap sesuai dengan kebutuhan pengguna dan tujuan simulasi.

3.5.8 Tahap 5: Deployment

Hingga saat penyusunan laporan ini, proyek masih berada dalam tahap pengembangan dan belum mencapai fase deployment atau implementasi akhir ke perangkat target di lingkungan produksi. Oleh karena itu, proses deployment secara resmi belum dilakukan.

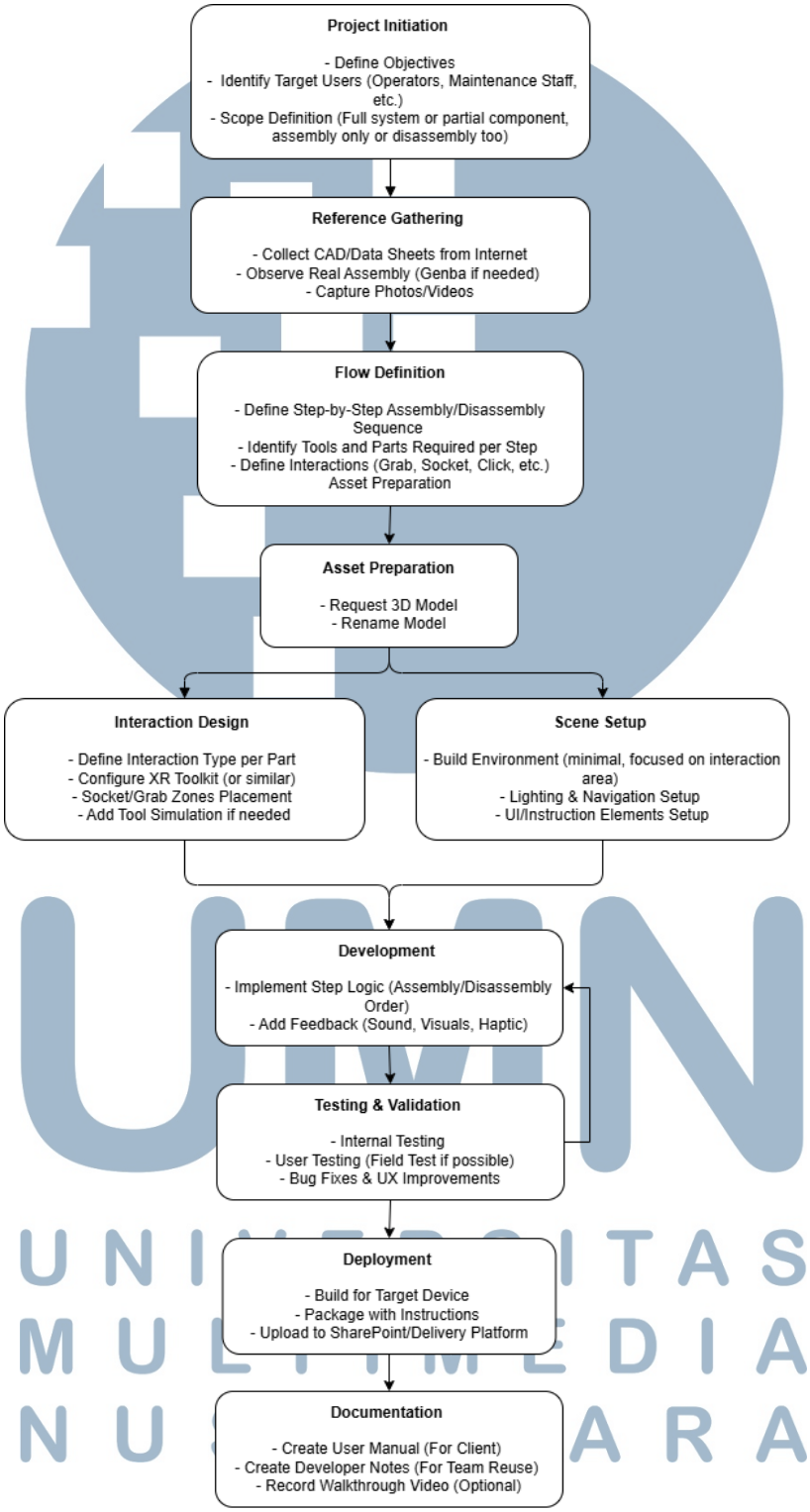
Namun demikian, rencana deployment telah disusun sejak awal dan dirancang agar dapat dilakukan dengan efisien begitu aplikasi dinyatakan siap. Rencana tersebut meliputi proses build aplikasi untuk perangkat Meta Quest 3, pengemasan file hasil build lengkap dengan dokumentasi penggunaan, serta distribusi melalui platform internal seperti SharePoint atau sistem penyimpanan cloud perusahaan. Selain itu, dokumentasi teknis bagi pengembang dan walkthrough visual juga direncanakan untuk disiapkan guna mendukung kelanjutan proyek atau iterasi pengembangan di masa mendatang.

Dalam implementasinya nanti, proses deployment akan mempertimbangkan kesiapan dari sisi konten, kestabilan sistem, serta integrasi dengan kebutuhan pelatihan operator di lingkungan pengguna akhir. Diharapkan ketika aplikasi telah mencapai versi final, deployment dapat dilakukan dengan lancar dan memenuhi standar kualitas yang diharapkan oleh klien.

3.6 Framework Pengembangan Aplikasi VR

Sebagai bagian dari upaya mendukung pengembangan proyek serupa di masa mendatang, dirumuskan sebuah framework teknis pengembangan aplikasi VR berdasarkan pengalaman selama magang dan pelaksanaan proyek ini. Framework ini bertujuan untuk memberikan panduan sistematis, mulai dari tahap inisiasi hingga dokumentasi akhir, yang dapat diikuti oleh tim pengembang dalam proyek-proyek

sejenis.



Gambar 3.42. Flowchart framework pengembangan aplikasi VR (disusun berdasarkan pengalaman proyek).

Framework ini terdiri atas beberapa tahapan utama:

1. Project Initiation: Menentukan tujuan proyek dan ruang lingkup simulasi.
2. Reference Gathering: Mengumpulkan referensi teknis seperti datasheet dan dokumentasi mesin.
3. Flow Definition: Menyusun urutan kerja operator secara detail.
4. Asset Preparation, Interaction Design, Scene Setup: Merancang dan menyiapkan komponen interaktif serta lingkungan simulasi.
5. Development: Implementasi logika kerja dan pemberian umpan balik visual/auditori.
6. Testing & Validation: Uji internal maupun lapangan serta perbaikan bug dan UX.
7. Deployment & Documentation: Kompilasi aplikasi untuk perangkat target, pelengkapan dokumentasi, serta panduan penggunaan.

Framework ini disusun secara modular agar dapat digunakan secara fleksibel, baik dalam pendekatan pengembangan iteratif seperti Agile, maupun pendekatan linier jika diperlukan. Dengan dokumentasi ini, diharapkan proyek-proyek VR ke depan dapat dilaksanakan lebih efisien, terstruktur, dan terdokumentasi dengan baik.

3.7 Kendala dan Solusi yang Ditemukan

3.7.1 Kendala

Selama pelaksanaan magang, terdapat beberapa kendala yang dihadapi, baik dari sisi teknis maupun manajerial. Kendala-kendala tersebut menjadi bagian alami dari proses pengembangan proyek simulasi berbasis Virtual Reality (VR), terutama karena proyek masih dalam tahap pengembangan awal dan melibatkan banyak pihak lintas divisi. Berikut adalah rangkuman kendala beserta solusi yang diterapkan:

1. Keterbatasan ketersediaan aset 3D pada awal pengembangan.
2. Ketidakpastian dalam alur validasi prosedur kerja di lapangan.

3. Tantangan dalam pengaturan interaksi fisika di Unity.
4. Koordinasi lintas divisi yang memerlukan validasi kolektif.
5. Belum dilakukan pengujian dan deployment akhir proyek.

3.7.2 Solusi

Sebagai tindak lanjut atas berbagai kendala yang ditemui selama proses pengembangan, dilakukan perumusan solusi yang bersifat aplikatif dan tepat sasaran guna mengatasi permasalahan yang muncul. Pendekatan penyelesaian tidak hanya berfokus pada aspek teknis, tetapi juga mempertimbangkan keberlanjutan sistem serta kemudahan integrasi dengan ekosistem layanan yang sudah berjalan. Dengan metode penanganan yang sistematis, setiap permasalahan diupayakan untuk diselesaikan secara optimal. Adapun solusi yang diterapkan dalam menjawab masing-masing kendala dijelaskan sebagai berikut.

1. Tim pengembang memanfaatkan waktu dengan membangun terlebih dahulu *Proof of Concept* (PoC) yang berfokus pada interaksi dasar seperti *grab*, *release*, validasi *socket*, serta pemukulan paku. Hal ini memungkinkan progres tetap berjalan sambil menunggu aset final tersedia.
2. Dilakukan observasi langsung ke lapangan (*genba*) dan sesi diskusi dengan tim Project Management (PM) serta teknisi pengguna akhir. Hasilnya dirumuskan dalam bentuk *flowchart* prosedur kerja yang akurat dan digunakan sebagai dasar pengembangan simulasi.
3. Masalah seperti penggabungan objek dengan *FixedJoint*, konfigurasi *collider* untuk *socket interaction*, serta pengaturan umpan balik visual diatasi dengan pendekatan *trial-and-error*, studi dokumentasi XR Toolkit, dan referensi *best practice* dari komunitas pengembang Unity.
4. Diterapkan komunikasi rutin melalui *weekly meeting* untuk menyinkronkan progres, melakukan validasi fitur, serta mendokumentasikan keputusan bersama dengan semua pihak terkait.
5. Dengan menerapkan proses pengembangan yang iteratif dan terbuka, tim dapat mengidentifikasi serta menyelesaikan kendala lebih awal, memperkuat fondasi teknis dan kesiapan untuk fase selanjutnya.