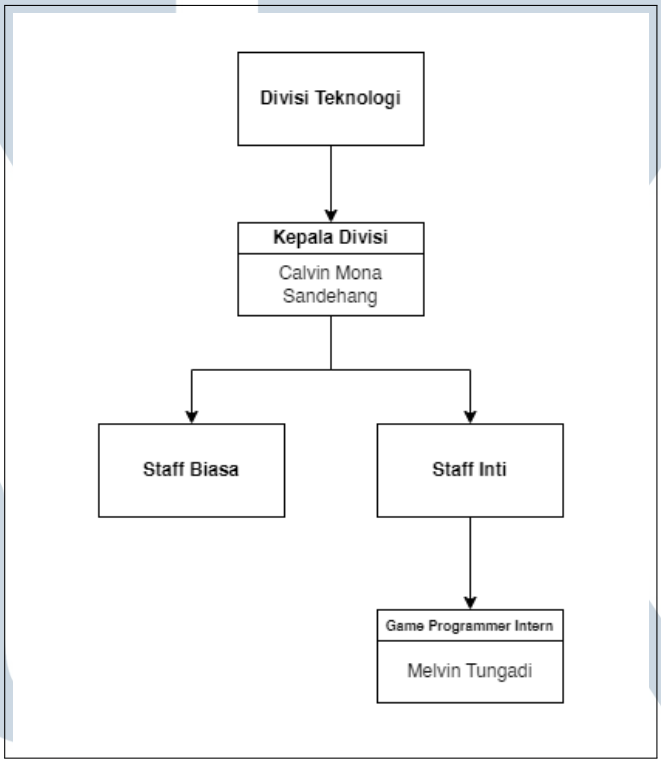
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Kedudukan posisi dalam pelaksanaan kegiatan kerja magang di KUMAGEMA berada dibawah Divisi Teknologi yang dikepalai oleh Calvin Mona Sandehang selaku CEO atau Direktur Utama KUMAGEMA sebagai *staff* inti dengan posisi sebagai *Game Programmer Intern* yang dapat dilihat pada Gambar 3.1.



Gambar 3.1. Kedudukan Pelaksanaan Magang di KUMAGEMA

Setiap divisi memiliki dua jenis *staff* yaitu *staff* inti dan *staff* biasa. *Staff* inti merupakan anggota tim yang memiliki peran utama dalam pengembangan proyek *game* yang sedang aktif dan memerlukan partisipasi dalam pembuatan keputusan kreatif dan teknis. Sementara *staff* biasa merupakan anggota tim yang mendukung kegiatan sehari-hari dan operasional dasar di dalam divisi.

3.2 Tugas yang Dilakukan

Dalam melaksanakan kegiatan kerja magang di KUMAGEMA, tugas yang diberikan merupakan pengembangan proyek utama perusahaan yaitu *role-playing game* simulasi bisnis penjualan minuman dengan nama *Drink-It-Up*. Fokus utama dari kerja magang diarahkan untuk mengembangkan implementasi *user interface* (UI), *user experience* (UX), dan elemen visual ke dalam *game*. Selain fokus utama pekerjaan, dilakukan juga pengujian *game* secara berkala untuk menemukan dan memperbaiki *bug* serta diskusi bersama *artist* dan *game designer* untuk membahas tahap implementasi dari desain dan rancangan yang sudah dibuat.

Program *game* dikembangkan menggunakan *game engine* Unity dengan bahasa pemrograman C# untuk platform Windows, Mac, dan Linux. Sebelum menerapkan implementasi fitur dan sistem ke dalam kode atau *game engine* dengan pemrograman berorientasi objek, dirancang terlebih dahulu diagram untuk memetakan sistem apa saja yang harus dibuat dan mengetahui ketergantungan satu sistem dengan sistem lainnya. Berikut merupakan sistem dari proyek *game* yang dikembangkan selama pelaksanaan kerja magang:

1. Grafik radar rasa bahan baku minuman.
2. Informasi statistik laporan penjualan.
3. Halaman menu utama penjualan.
4. Halaman pembuatan resep minuman.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kegiatan kerja magang di KUMAGEMA dapat diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan selama pelaksanaan kegiatan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Perkenalan dan pemaparan mengenai alur kerja dalam perusahaan dan melakukan pembelajaran secara mandiri mengenai konsep pemrograman berorientasi objek dalam Unity.
2	Pengujian dan pencarian *bug* untuk versi *game* yang pertama.
3	Pengujian dan pencarian *bug* untuk versi *game* yang pertama.
4	Perancangan dan pengembangan mekanik game baru untuk versi *game* yang kedua berdasarkan umpan balik dan evaluasi dari pengujian *game* versi pertama.
5	Pengembangan implementasi UI, UX, dan elemen visual untuk grafik radar rasa bahan baku minuman.
6	Pengembangan implementasi UI, UX, dan elemen visual untuk informasi statistik laporan penjualan.
7	Pengembangan implementasi UI, UX, dan elemen visual untuk halaman menu utama penjualan.
8	Pengembangan implementasi UI, UX, dan elemen visual untuk halaman menu utama penjualan.
9	Pengembangan implementasi UI, UX, dan elemen visual untuk halaman pembuatan resep minuman.
10	Pengembangan implementasi UI, UX, dan elemen visual untuk halaman pembuatan resep minuman.

Kegiatan kerja magang pada minggu pertama diawali dengan perkenalan *supervisor* dan anggota tim lainnya serta penjelasan mengenai alur kerja yang digunakan dalam perusahaan seperti *software* apa saja yang digunakan, standar dan pedoman pengerjaan, dan prosedur pengumpulan hasil pekerjaan. Dalam tahap awal ini, *supervisor* menyarankan untuk melakukan pembelajaran secara mandiri mengenai konsep pemrograman berorientasi objek dalam Unity untuk menyesuaikan kode yang dibuat dengan standar perusahaan. Pembelajaran dilakukan melalui dokumentasi resmi Unity dan video tutorial dari Youtube

mengenai topik seperti penerapan *clean code*, penggunaan *class* sebagai komponen, dan pemahaman mengenai berbagai macam *design pattern*.

Pada minggu kedua dan ketiga, perusahaan baru saja menyelesaikan tahap pengembangan *game* untuk versi yang pertama dan dilanjutkan dengan pengujian secara internal dan eksternal untuk mendapatkan umpan balik sebagai referensi untuk pengembangan versi berikutnya. Tugas yang diberikan oleh *supervisor* adalah membantu dalam pengujian dan mencari *bug* yang masih ada baik itu dalam bentuk visual ataupun pada alur *game*. Pengujian dilakukan dengan memainkan *game* berulang kali dan mencatat *bug-bug* apa saja yang masih muncul. Tahap pengujian ini juga berguna untuk lebih memahami cara kerja mekanik dan *flow* di dalam *game* yang akan dikembangkan kedepannya.

Pada minggu keempat, tugas yang diberikan selanjutnya adalah mengembangkan konsep mekanik baru berdasarkan umpan balik yang didapatkan dari pengujian *game* versi pertama yang dilakukan bersama dengan *programmer* lainnya. Mekanik yang akan dibuat berpusat pada tipe-tipe alat untuk membuat bahan baku mentah menjadi bahan olahan untuk membuat minuman. Pengembangan mekanik baru masih dilakukan pada *repository* sendiri agar tidak mempengaruhi pekerjaan pada *repository* utama *game*. Pengembangan mekanik baru juga dilakukan sebagai asesmen untuk menguji kemampuan masing-masing *programmer* dalam bekerja sebagai tim yang nantinya berguna pada saat pembagian tugas yang akan dikerjakan pada *repository* utama.

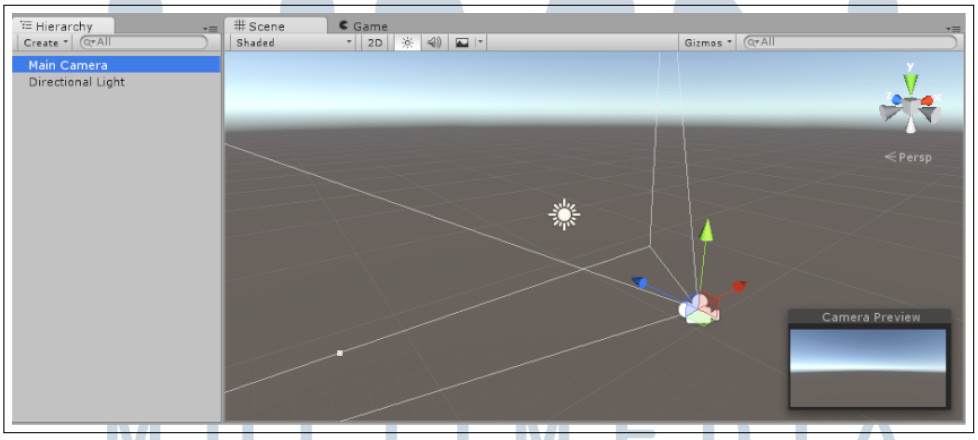
Pada minggu kelima sampai minggu kesepuluh, pengembangan proyek *game* pada *repository* utama dapat dilaksanakan setelah menandatangani surat perjanjian kerja sebagai upaya untuk tidak sembarang menggunakan atau mengubah aset di dalam *repository* utama tanpa sepengetahuan atau izin dari *supervisor*. Pengerjaan proyek diarahkan untuk mengembangkan bagian UI, UX, dan elemen visual berdasarkan hasil asesmen yang diberikan oleh *supervisor*. Pengembangan proyek *game* dimulai dari membuat komponen visual yang digunakan untuk mempresentasikan data yaitu grafik radar rasa bahan baku minuman dan informasi statistik laporan penjualan. Setelah selesai membuat komponen visual, tugas selanjutnya adalah membuat halaman menu utama penjualan dan halaman pembuatan resep minuman sebagai alur utama *game* yang akan berinteraksi dengan pemain.

3.4 Pengembangan Proyek Game

Penjelasan pengembangan proyek *game* terbagi menjadi beberapa bagian yang mencakup penggunaan dan alur kerja pada Unity, konsep *design pattern* yang digunakan dalam Unity untuk mengembangkan sistem *game*, dan proses pengembangan masing-masing sistem *game* yang dikerjakan. Dalam proses pengembangan sistem *game*, alur pengerjaan dijabarkan lebih lanjut mulai dari rancangan *game design* yang telah dibuat, *flowchart* sistem yang akan dibuat berdasarkan *game design*, *class* diagram sistem untuk memetakan komponen yang akan diterapkan ke dalam Unity, pengaturan penempatan objek dan properti sistem di Unity, dan hasil implementasi sistem secara keseluruhan.

3.4.1 Penggunaan Unity

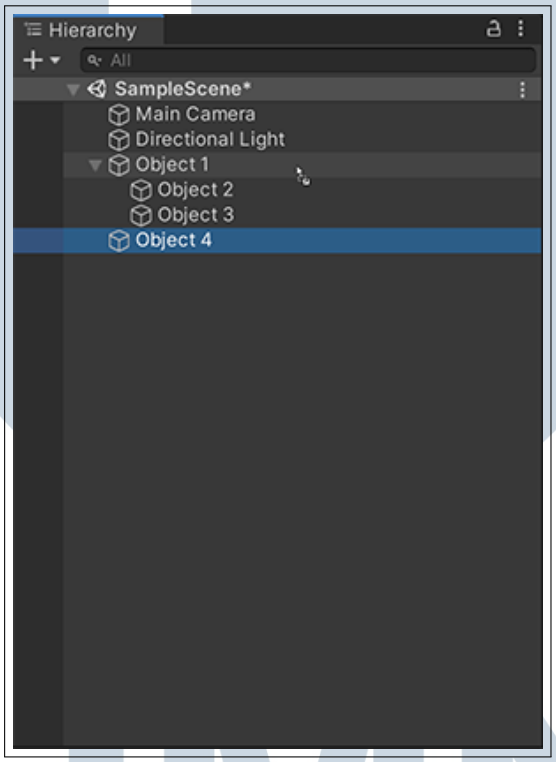
Unity merupakan *game engine* yang digunakan dalam proses pengembangan *game* dan memiliki berbagai macam fitur untuk mengatur dan memanipulasi objek di dalam *game*. Penambahan dan pengaturan penempatan objek dilakukan di dalam suatu *Scene*. *Scene* merupakan area kerja yang dapat menampung berbagai macam komponen *game* mulai dari lingkungan, karakter, UI, dan lain-lainnya [12]. Setiap *Scene* memiliki sebuah kamera yang berfungsi untuk menampilkan seluruh isi di dalam *Scene* tersebut.



Gambar 3.2. *Scene* dalam Unity

Sumber: [12]

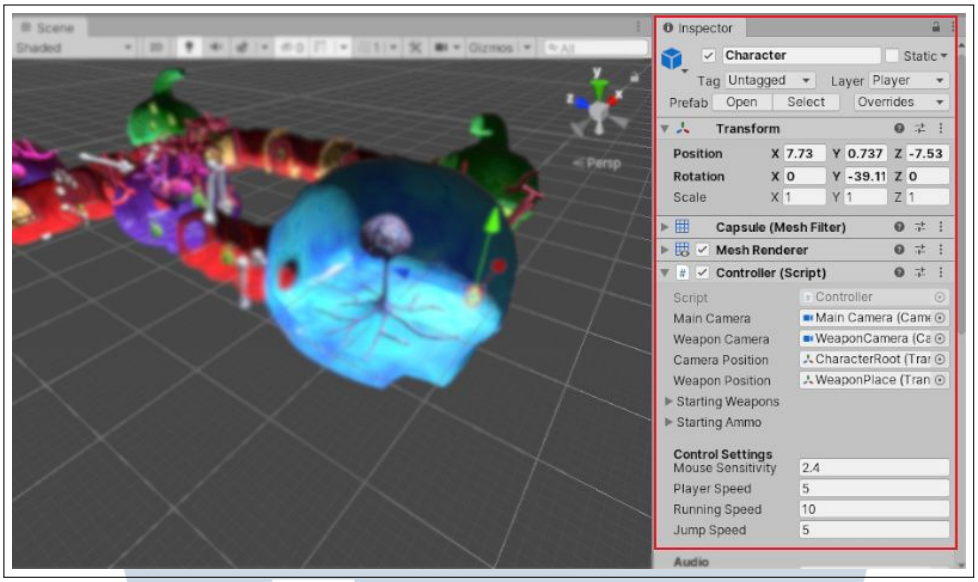
Setiap objek yang ditambahkan ke dalam *Scene* dapat dilihat kedudukan dan urutannya melalui panel *Hierarchy*. *Hierarchy* menampilkan seluruh objek di dalam *Scene* yang dapat diubah urutannya menjadi di depan atau di belakang objek lain dan dapat digrup menjadi suatu anak (*child*) dari induk (*parent*) objek [13]. *Hierarchy* berfungsi sebagai pengaturan *layer* dari objek dan mengakses properti dari objek tersebut.



Gambar 3.3. *Hierarchy* dalam Unity

Sumber: [13]

Properti objek yang terdapat pada *Hierarchy* dapat dilihat pada panel *Inspector*. *Inspector* berfungsi untuk melihat dan menyunting properti dari segala hal yang terdapat di dalam editor Unity mulai dari objek *game*, pengaturan komponen, pengaturan aset, dan lain-lain [14]. *Inspector* dapat melihat properti objek yang memiliki komponen bawaan dari Unity seperti informasi posisi, rotasi, dan skala objek di dalam *Scene* dan juga berbagai macam komponen lain yang dapat ditambahkan. Salah satunya adalah komponen *script* C# yang berguna untuk menjalankan kode di dalam *script* dan mendapatkan referensi properti objek pada *Scene* yang dapat diubah nilai atau datanya melalui kode di dalam *script*.



Gambar 3.4. *Inspector* dalam Unity

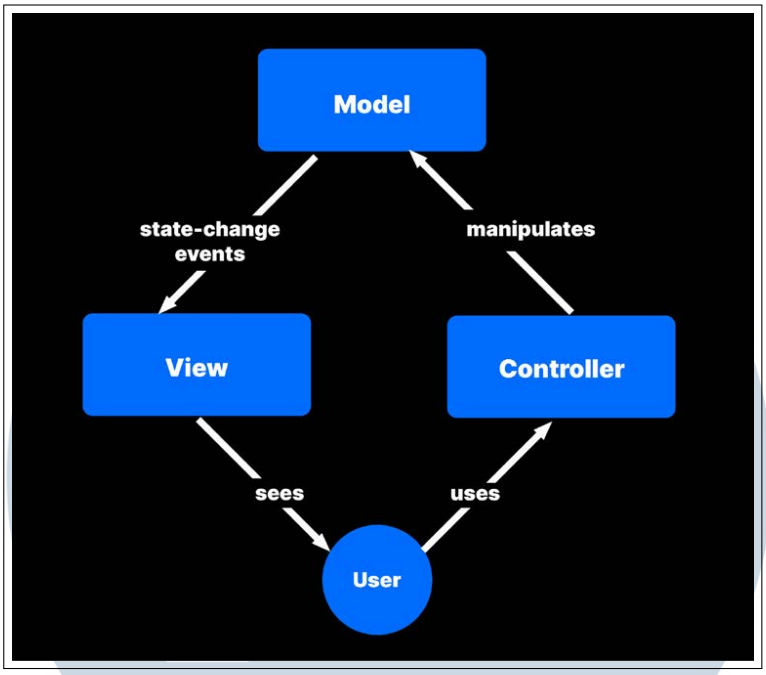
Sumber: [14]

Setiap *script* C# merupakan suatu *class* tersendiri, namun *class* biasa tidak dapat ditambahkan sebagai komponen untuk suatu objek dan dijalankan oleh Unity sebelum menurunkan *class MonoBehaviour*. *MonoBehaviour* merupakan *class* dasar yang diturunkan oleh sebagian besar *script* di Unity dan menampung beberapa metode *life cycle* [15]. Ketika *game* mulai dijalankan, Unity akan memanggil seluruh metode yang diturunkan dari *class MonoBehaviour* untuk menginisialisasi kode yang akan dijalankan pada masing-masing *class*.

3.4.2 Penggunaan Design Pattern

A. Model, View, Controller (MVC)

Dalam mengembangkan sistem untuk proyek *game* yang akan dibuat, diterapkan penggunaan *design pattern* untuk memastikan bahwa sistem dapat dengan mudah dibaca dan diatur seiring waktu serta konsisten dengan hasil pengerjaan *programmer* lain. *Design pattern* yang diterapkan adalah *Model, View, Controller* atau MVC sebagai dasar dari seluruh pengembangan sistem di dalam proyek *game*.



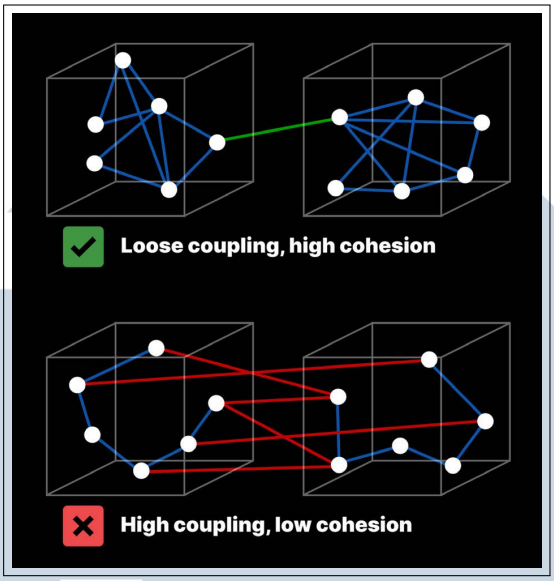
Gambar 3.5. Desain *Model, View, Controller*

Sumber: [16]

Prinsip utama dari desain MVC adalah memisahkan bagian logika, data, dan visual dari suatu sistem untuk mengurangi ketergantungan yang tidak diperlukan serta menghindari penerapan kode yang berantakan [16]. *Model* memiliki tugas utama dalam menyimpan seluruh nilai dan data yang ada di dalam sistem, *View* memiliki tugas utama untuk mempresentasikan data yang terdapat di dalam *Model*, dan *Controller* memiliki tugas utama sebagai pusat operasi logika dari sistem yang dapat mengubah data yang disimpan di dalam *Model* dan mengirimkan data untuk ditampilkan oleh *View*.

B. Dependency Inversion

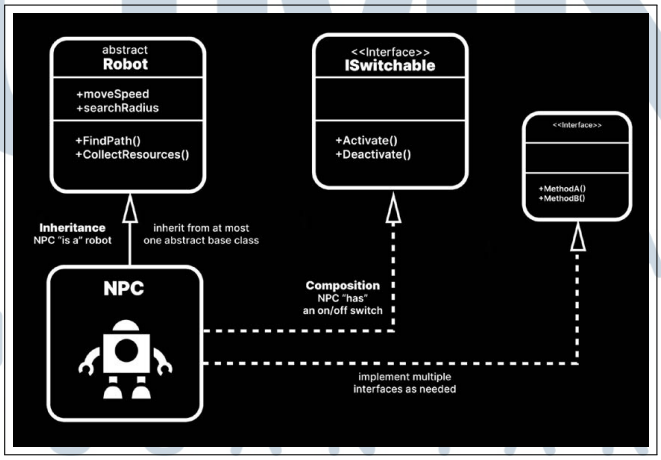
Selain penggunaan desain MVC, diterapkan juga salah satu prinsip inti dari desain perangkat lunak yaitu *dependency inversion*. *Dependency inversion* merupakan upaya untuk mengurangi ketergantungan antar *class* konkrit dengan menggunakan abstraksi dalam bentuk *class* abstrak atau penggunaan *interface* [16]. Ketika fungsi di dalam *class* saling berkegantungan dengan fungsi pada *class* lain, maka akan meningkatkan resiko terjadinya *error* bila salah satu *class* berubah. Hal tersebut dinamakan sebagai *high coupling* dan *low cohesion*.



Gambar 3.6. Ilustrasi *Coupling* dan *Cohesion* antar *Class*

Sumber: [16]

Dengan menerapkan *dependency inversion*, suatu *class* dapat mereferensikan *class* lain tanpa mengetahui isi atau cara kerja fungsi pada *class* yang direferensikan. Ketergantungan suatu *class* diarahkan ke bentuk abstraksi yang dapat diturunkan atau diimplementasikan oleh banyak *class*. Tujuan utama dari *dependency inversion* adalah membuat kerangka sistem yang dapat diimplementasikan dan dapat digunakan kembali tanpa memiliki ketergantungan pada masing-masing hasil implementasi.



Gambar 3.7. Penerapan *Dependency Inversion*

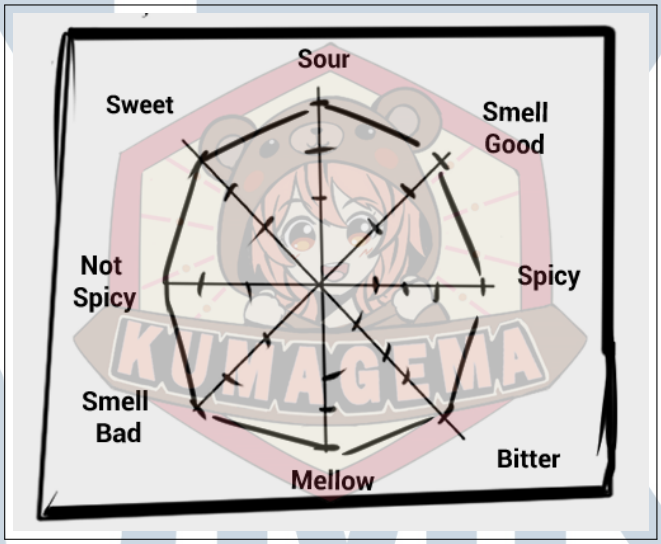
Sumber: [16]

3.4.3 Pengembangan Grafik Radar Rasa Bahan Baku Minuman

Grafik radar rasa bahan baku minuman merupakan sistem *game* untuk menampilkan informasi properti rasa dari bahan baku untuk membuat minuman. Informasi tersebut dapat digunakan pemain untuk mengetahui kualitas suatu bahan secara individu atau yang sudah dicampurkan menjadi minuman yang nantinya akan mempengaruhi daya beli pada saat proses penjualan.

A. Rancangan Game Design

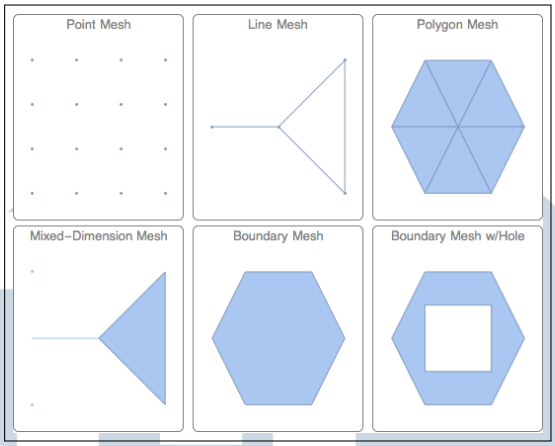
Proses pengembangan dimulai dari *game design* dalam bentuk rancangan *wireframe* yang telah dibuat oleh *game designer* dan *UX designer* yang dapat dilihat pada Gambar 3.8.



Gambar 3.8. *Wireframe* Grafik Radar Rasa Bahan Baku Minuman

Sumber: UX Designer KUMAGEMA

Dari rancangan tersebut, dibuat sebuah *requirements* untuk membuat sistem dengan menggunakan bentuk *Mesh* sebagai visualisasi grafik yang dapat diubah propertinya sesuai dengan nilai masing-masing tipe rasa. *Mesh* merupakan kerangka dari suatu objek di dalam game yang terdiri dari beberapa *vertices* dan *triangles* untuk membuat bentuk objek [17]. *Vertices* bertindak sebagai poin utama yang akan digunakan untuk membuat *triangles* yang akan menampilkan bentuk dan warna atau tampilan dari objek.

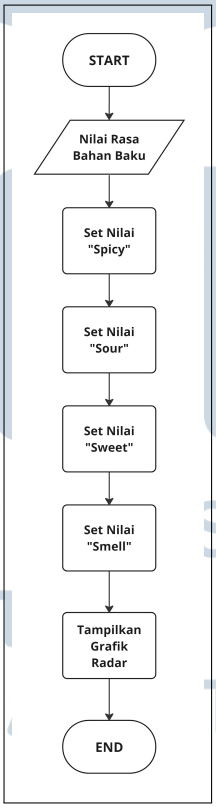


Gambar 3.9. Visualisasi *Mesh*

Sumber: [18]

B. Flowchart

Rancangan *game design* yang telah dibuat dapat digambarkan ke dalam bentuk *flowchart* yang dapat dilihat pada Gambar 3.10.

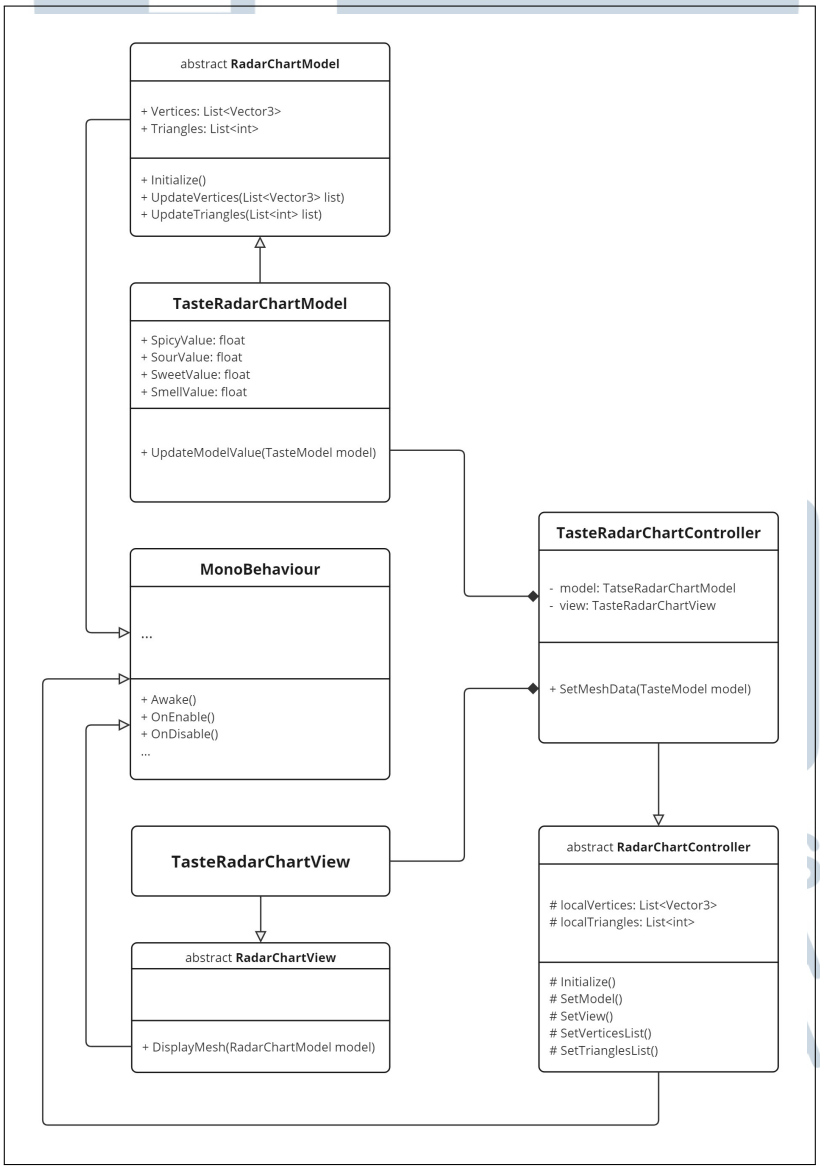


Gambar 3.10. *Flowchart* Grafik Radar Rasa Bahan Baku Minuman

Sistem akan mendapatkan informasi nilai untuk masing-masing tipe rasa yang ada di dalam bahan baku yang akan di-*set* sebagai data untuk masing-masing tipe rasa yang akan ditampilkan ke dalam bentuk grafik radar.

C. Class Diagram

Dari *flowchart* rancangan sistem yang telah dibuat, langkah selanjutnya adalah membuat *class* diagram untuk menggambarkan hubungan komponen di dalam sistem yang akan diterapkan ke dalam Unity yang dapat dilihat pada Gambar 3.11.

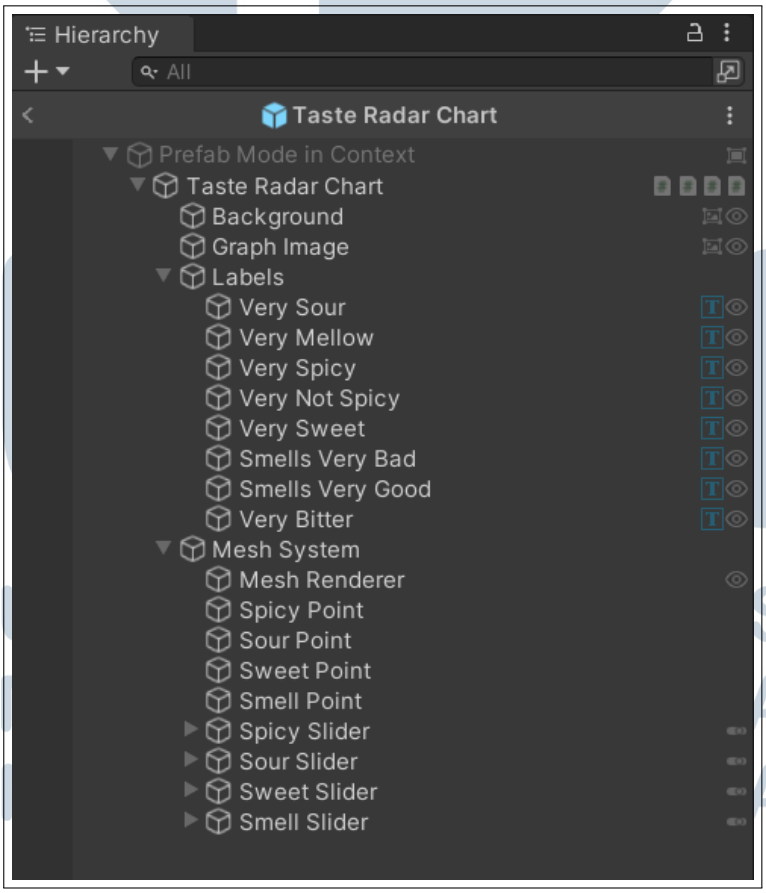


Gambar 3.11. *Class* Diagram Grafik Radar Rasa Bahan Baku Minuman

Komponen utama di dalam sistem adalah *TasteRadarChartModel*, *TasteRadarChartView*, dan *TasteRadarChartController*. *TasteRadarChartController* akan mendapatkan informasi nilai rasa dari *class* eksternal yang telah dibuat oleh *programmer* lain yaitu *TasteModel* untuk setiap tipe rasa (*Spicy, Sour, Sweet, Smell*) yang akan diproses untuk mendapatkan data *vertices* dan *triangles* yang disimpan ke dalam *TasteRadarChartModel* untuk ditampilkan ke dalam bentuk *Mesh* oleh *TasteRadarChartView*. Sistem akan diinisialisasi saat memulai *game* atau digunakan oleh suatu bagian *game* lainnya menggunakan metode *Awake*, *OnEnable*, dan *OnDisable* yang dimiliki oleh *MonoBehaviour*.

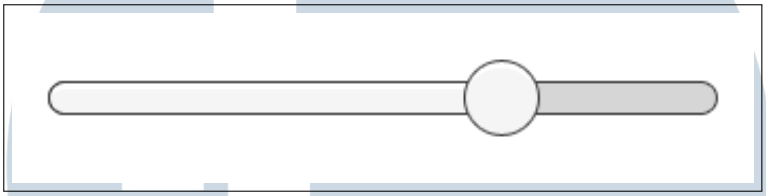
D. Pengaturan di Unity

Berikut adalah *Hierarchy* objek untuk sistem grafik radar rasa bahan baku minuman yang dapat dilihat pada Gambar 3.12.



Gambar 3.12. *Hierarchy* Sistem Grafik Radar Rasa Bahan Baku Minuman

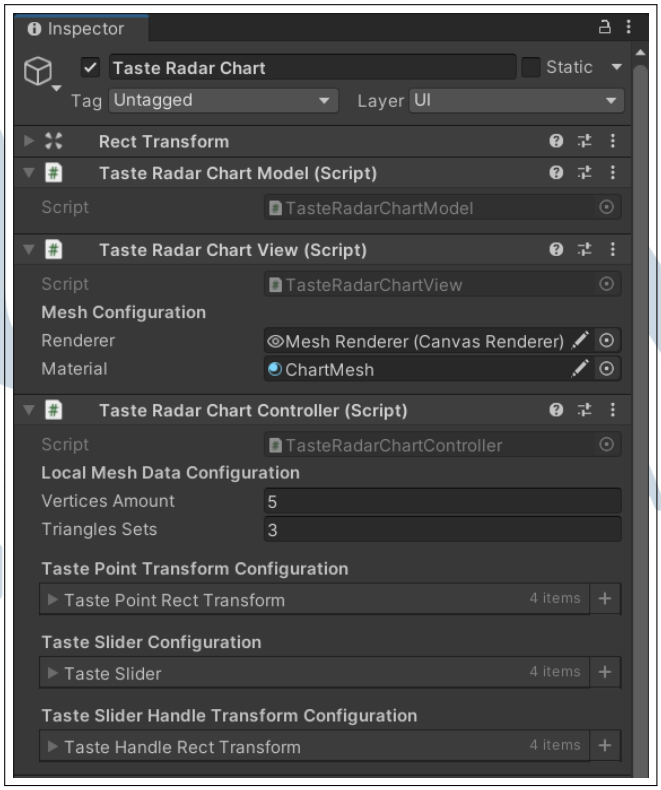
Untuk mengatur posisi *vertices* yang digunakan oleh *triangles* agar dapat menampilkan *Mesh* di dalam *Scene*, digunakan komponen UI bawaan Unity yaitu *Slider* yang dapat diubah persentase nilainya sesuai pengaturan nilai maksimum dan minimum. Objek *point* untuk masing-masing tipe rasa bertindak sebagai *vertices* untuk mengambil referensi posisi *Slider* masing-masing tipe rasa yang digunakan untuk membuat *triangles* dalam *Mesh*.



Gambar 3.13. *Slider* dalam Unity

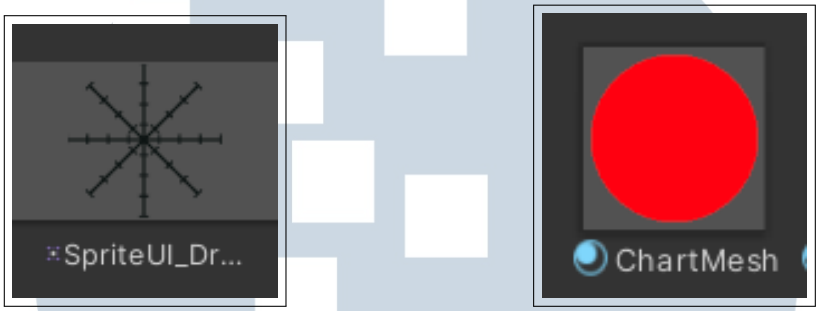
Sumber: [19]

Komponen *script* untuk sistem grafik radar rasa bahan baku minuman dimasukkan ke dalam objek untuk diinisialisasi oleh Unity. *Script* juga dapat mengambil referensi objek lainnya yang akan diatur propertinya di dalam kode dan nilai-nilai yang dapat diatur melalui *Inspector* yang dapat dilihat pada Gambar 3.14.



Gambar 3.14. Pengaturan Komponen Grafik Radar

Aset untuk sistem grafik radar rasa bahan baku minuman yang akan diimplemenntasikan ke dalam *Scene* telah dibuat oleh 2D *Artist* yang dapat dilihat pada Gambar 3.15a dan 3.15b. *Sprite* digunakan untuk menampilkan gambar sumbu grafik radar dan *Material* digunakan untuk menampilkan warna grafik radar.

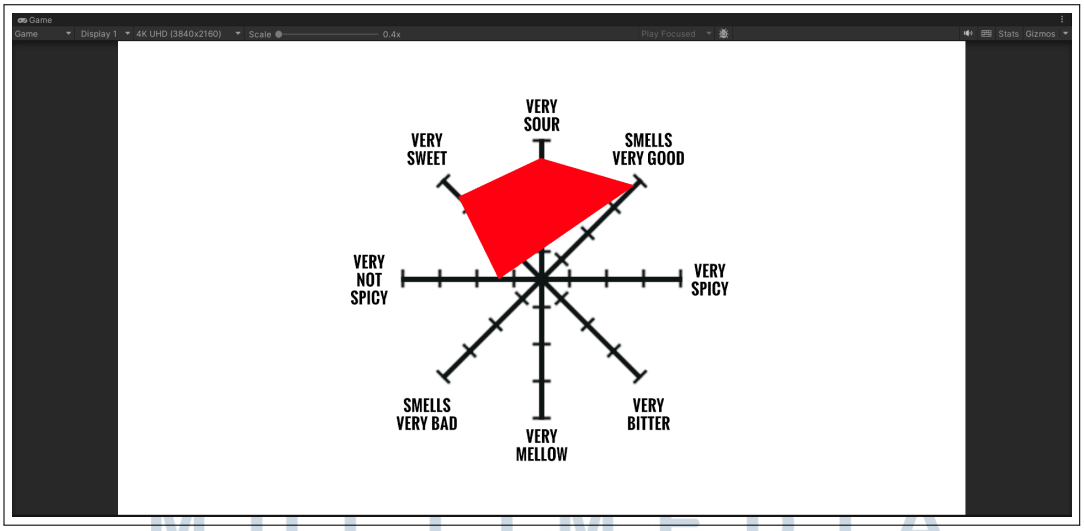


(a) *Sprite* Sumbu Grafik Radar (b) *Material* Grafik Radar

Gambar 3.15. Aset Grafik Radar Rasa Bahan Baku Minuman

E. Hasil Implementasi

Berikut adalah hasil implementasi sistem grafik radar rasa bahan baku minuman yang dapat dilihat pada Gambar 3.16.



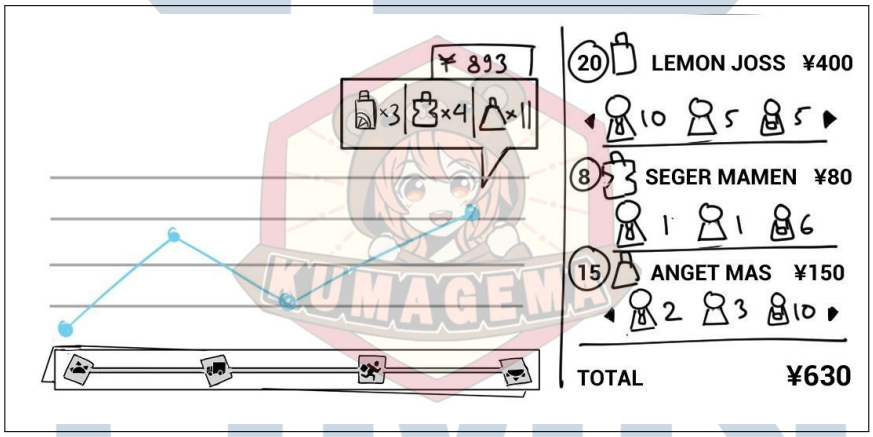
Gambar 3.16. Hasil Implementasi Grafik Radar Rasa Bahan Baku Minuman

3.4.4 Pengembangan Informasi Statistik Laporan Penjualan

Informasi statistik laporan penjualan merupakan sistem *game* untuk menampilkan informasi penjualan minuman yang mencakup jumlah minuman yang telah terjual, jenis minuman yang paling banyak dibeli, dan jumlah pendapatan dari hasil penjualan dalam satu hari. Informasi tersebut dapat digunakan pemain untuk mengetahui minuman apa saja yang paling laku pada jam tertentu dan tipe konsumen untuk masing-masing minuman untuk membuat strategi agar bisa mendapatkan lebih banyak pendapatan pada hari-hari selanjutnya.

A. Rancangan Game Design

Proses pengembangan dimulai dari *game design* dalam bentuk rancangan *wireframe* yang telah dibuat oleh *game designer* dan *UX designer* yang dapat dilihat pada Gambar 3.17.



Gambar 3.17. *Wireframe* Informasi Statistik Laporan Penjualan

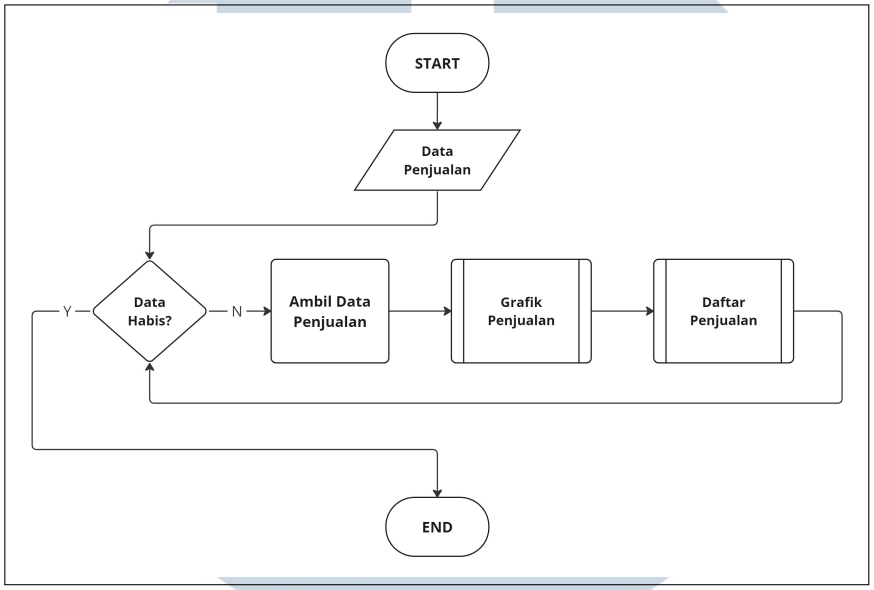
Sumber: UX Designer KUMAGEMA

Dari rancangan tersebut, dibuat sebuah *requirements* untuk membuat sistem yang dapat dijabarkan sebagai berikut:

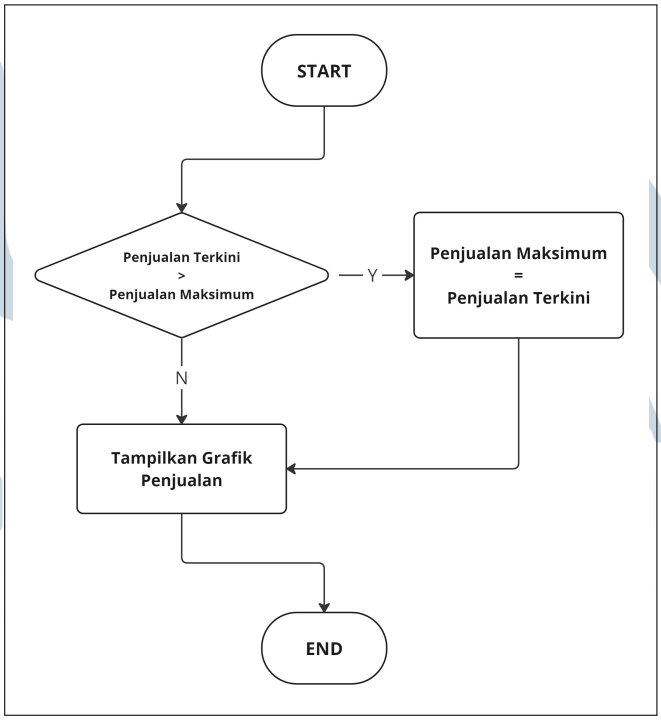
- Komponen grafik total penjualan untuk setiap jam.
- Komponen informasi penjualan minuman terbanyak dan total pendapatan untuk setiap jam.
- Komponen daftar jumlah penjualan minuman dan pendapatan dalam satu hari.

B. Flowchart

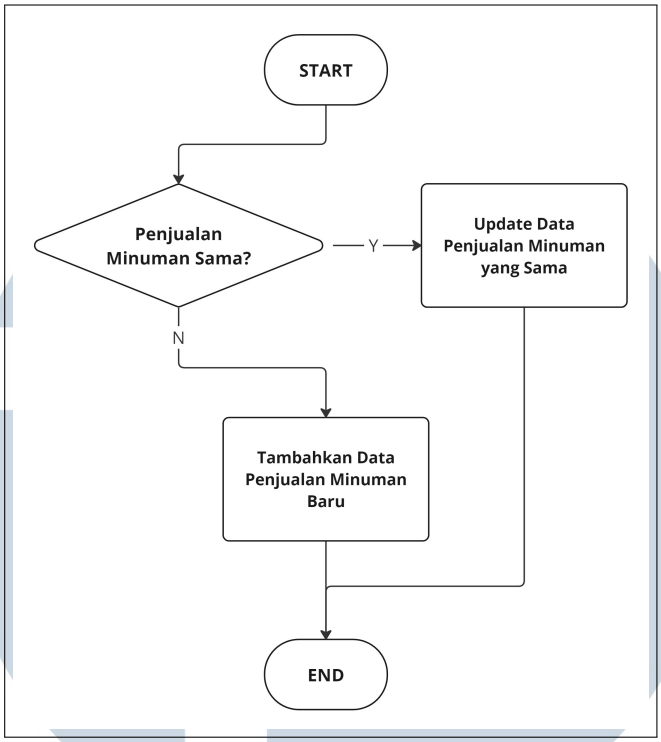
Rancangan *game design* yang telah dibuat dapat digambarkan ke dalam bentuk *flowchart* yang dapat dilihat pada Gambar 3.18, 3.19, dan 3.20.



Gambar 3.18. *Flowchart* Informasi Statistik Laporan Penjualan - Sistem Utama



Gambar 3.19. *Flowchart* Informasi Statistik Laporan Penjualan - Sistem Grafik

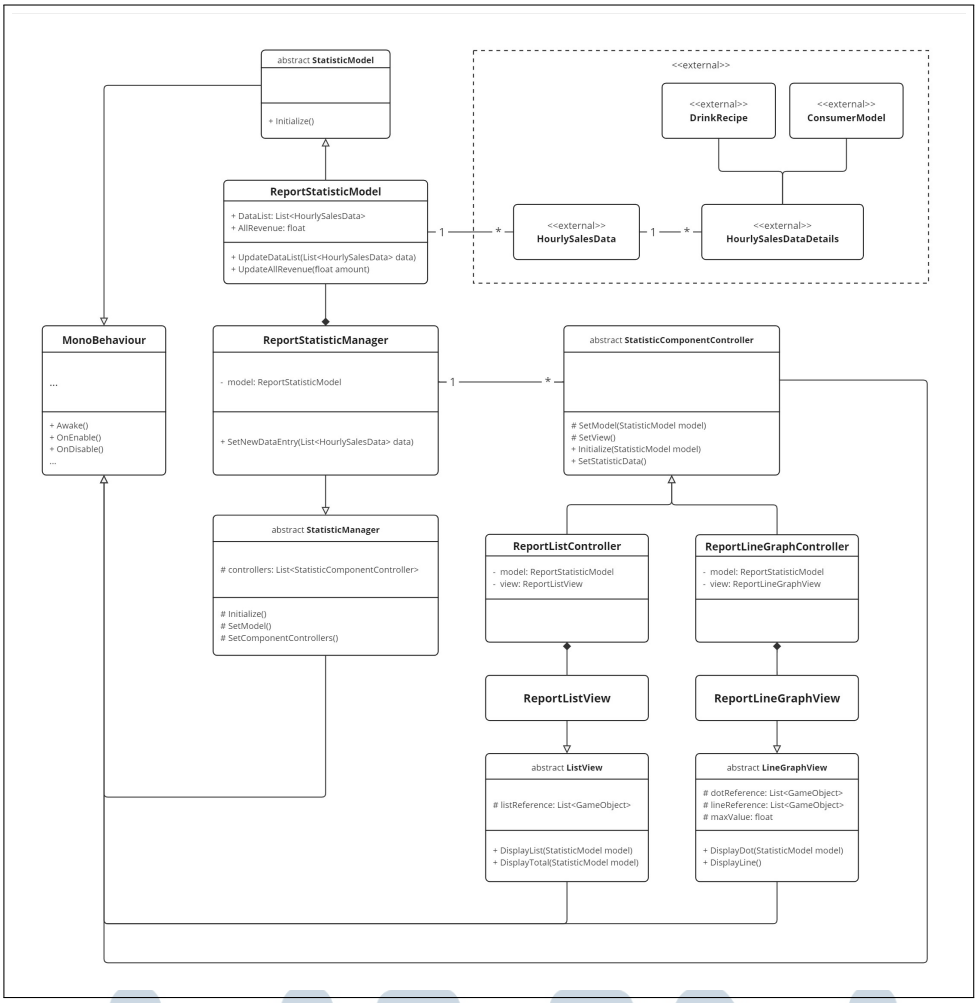


Gambar 3.20. *Flowchart* Informasi Statistik Laporan Penjualan - Sistem Daftar

Sistem akan mendapatkan informasi data penjualan dalam satu hari yang dijabarkan ke dalam bentuk grafik dan daftar penjualan. Grafik penjualan menampilkan jumlah penjualan untuk setiap jamnya dengan menyimpan data penjualan maksimum untuk menormalisasikan ukuran grafik dan menampilkan data penjualan minuman terbanyak. Sementara daftar penjualan menampilkan informasi jenis minuman yang dibeli oleh konsumen beserta jumlah pendapatan untuk setiap minuman yang terjual.

C. Class Diagram

Dari *flowchart* rancangan sistem yang telah dibuat, langkah selanjutnya adalah membuat *class* diagram untuk menggambarkan hubungan komponen di dalam sistem yang akan diterapkan ke dalam Unity yang dapat dilihat pada Gambar 3.21.

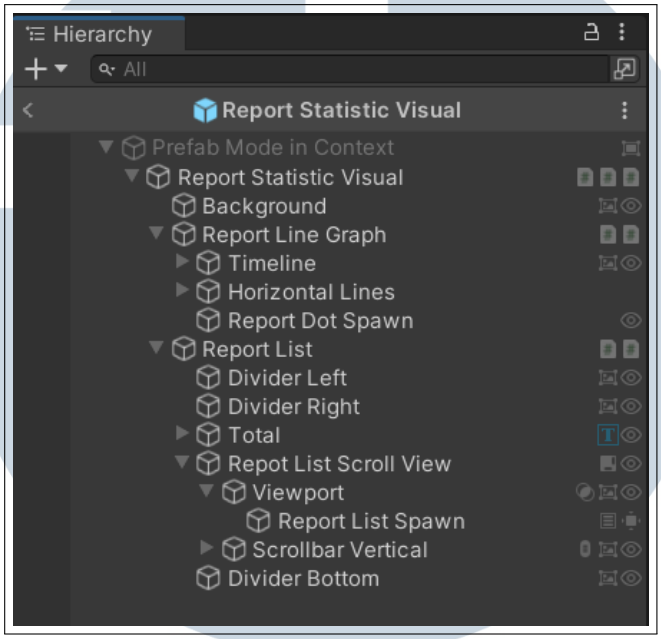


Gambar 3.21. *Class* Diagram Informasi Statistik Laporan Penjualan

Komponen utama di dalam sistem adalah *ReportStatisticModel*, *ReportStatisticManager*, *ReportListController*, *ReportListView*, *ReportLineGraphController*, dan *ReportLineGraphView*. *ReportStatisticManager* akan mendapatkan informasi data penjualan dari *class* eksternal yang telah dibuat oleh *programmer* lain yaitu *HourlySalesData* yang memuat detil penjualan pada *HourlySalesDataDetails* untuk informasi jenis minuman (*DrinkRecipe*), jenis konsumen (*ConsumerModel*), jumlah penjualan, dan jumlah pendapatan. Data penjualan tersebut disimpan ke dalam *ReportStatisticModel* untuk ditampilkan oleh *ReportListView* dan *ReportLineGraphView* melalui *controller*-nya masing-masing yang diatur oleh *ReportStatisticManager*. Sistem akan diinisialisasi saat memulai *game* atau digunakan oleh suatu bagian *game* lainnya menggunakan metode *Awake*, *OnEnable*, dan *OnDisable* yang dimiliki oleh *MonoBehaviour*.

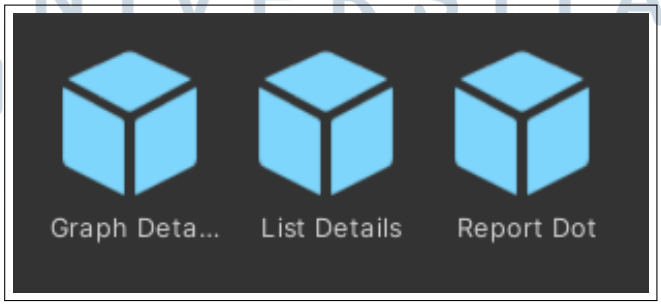
D. Pengaturan di Unity

Berikut adalah *Hierarchy* objek untuk sistem informasi statistik laporan penjualan yang dapat dilihat pada Gambar 3.22.



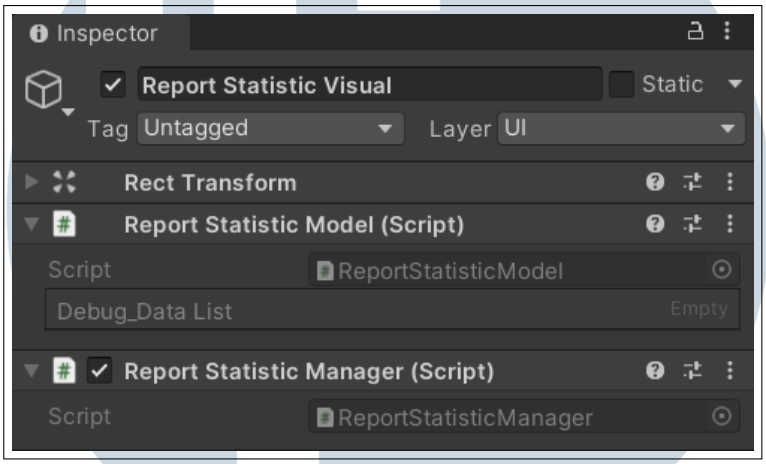
Gambar 3.22. *Hierarchy* Sistem Informasi Statistik Laporan Penjualan

Penggunaan *prefab* digunakan untuk membuat komponen visual yang terdapat di dalam sistem yang dapat dilihat pada Gambar 3.23. *Prefab* merupakan komponen di dalam Unity yang dapat dikonfigurasikan untuk menyimpan objek-objek yang dapat digunakan kembali [20]. *Graph Details* merupakan komponen untuk menampilkan data minuman yang paling banyak dibeli setiap jam, *List Details* merupakan komponen untuk menampilkan informasi detil penjualan di dalam daftar penjualan, dan *Report Dot* merupakan komponen untuk menampilkan titik di dalam grafik yang memuat total pendapatan setiap jam.

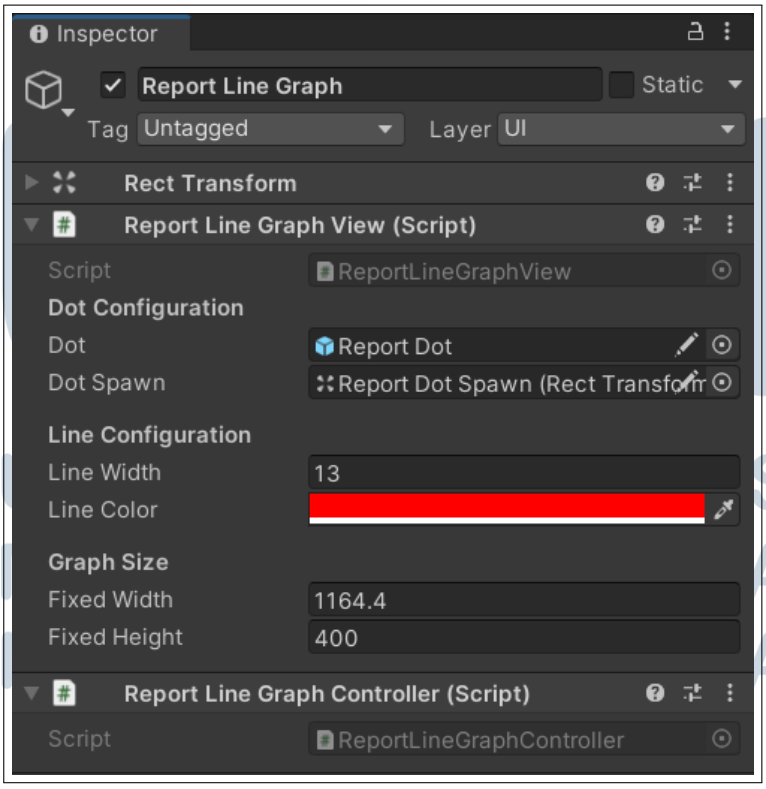


Gambar 3.23. *Prefab* Komponen Visual Informasi Statistik Laporan Penjualan

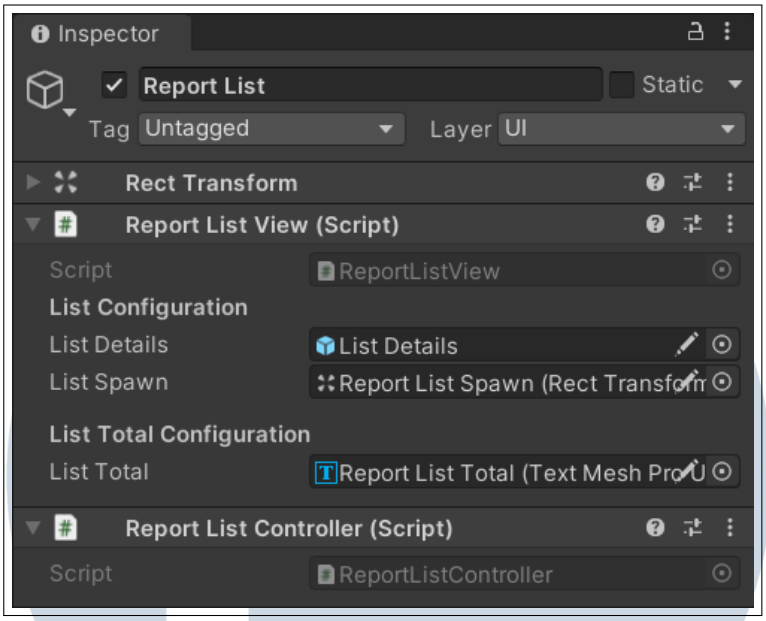
Komponen *script* untuk sistem informasi statistik laporan penjualan dimasukkan ke dalam objek untuk diinisialisasi oleh Unity. *Script* juga dapat mengambil referensi objek lainnya yang akan diatur propertinya di dalam kode dan nilai-nilai yang dapat diatur melalui *Inspector* yang dapat dilihat pada Gambar 3.24, 3.25, dan 3.26.



Gambar 3.24. Pengaturan Komponen Laporan Penjualan - Sistem Utama

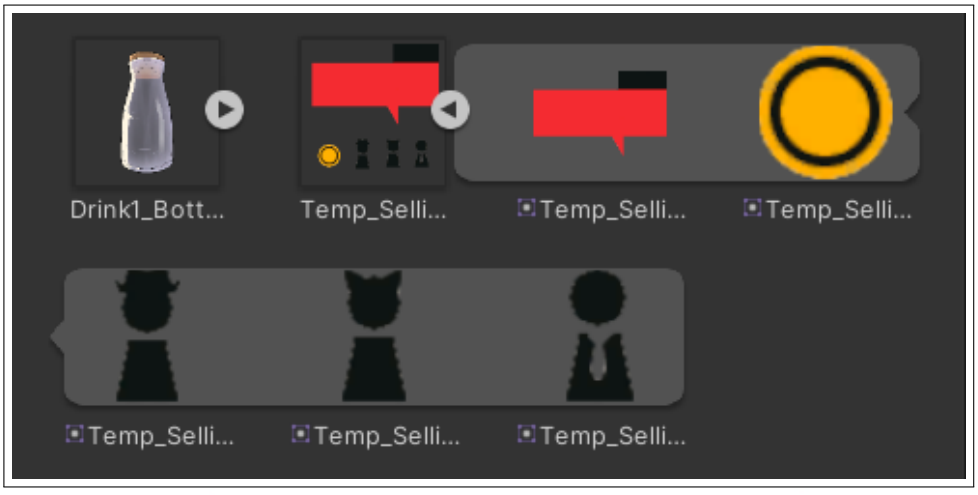


Gambar 3.25. Pengaturan Komponen Laporan Penjualan - Sistem Grafik



Gambar 3.26. Pengaturan Komponen Laporan Penjualan - Sistem Daftar

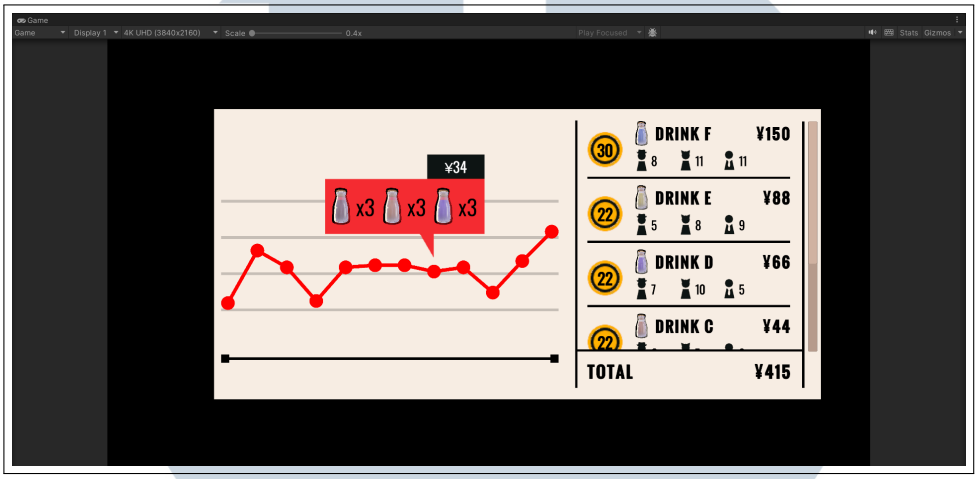
Aset untuk sistem informasi statistik laporan penjualan yang akan diimplemenntasikan ke dalam *Scene* telah dibuat oleh 2D *Artist* yang dapat dilihat pada Gambar 3.27



Gambar 3.27. Aset Informasi Statistik Laporan Penjualan

E. Hasil Implementasi

Berikut adalah hasil implementasi sistem informasi statistik laporan penjualan yang dapat dilihat pada Gambar 3.28.



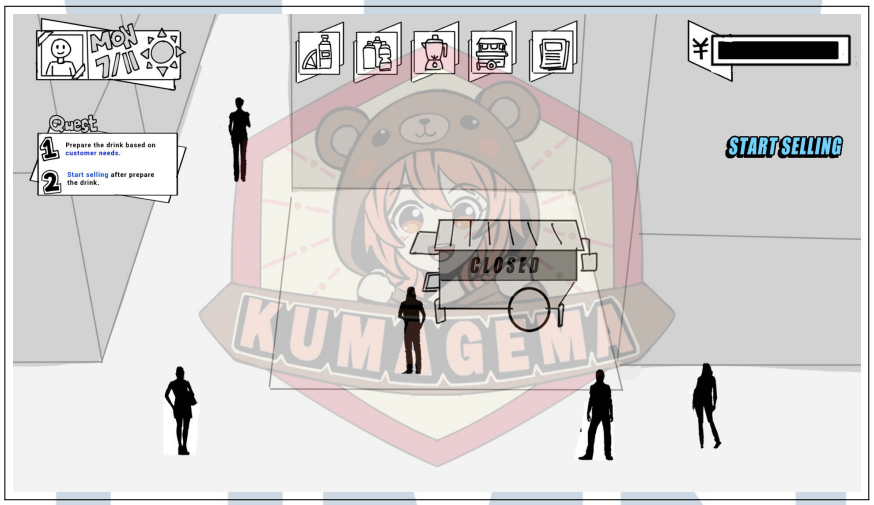
Gambar 3.28. Hasil Implementasi Informasi Statistik Laporan Penjualan

3.4.5 Pengembangan Halaman Menu Utama Penjualan

Halaman menu utama penjualan merupakan sistem *game* yang mengatur alur *gameplay* sebelum dan saat proses penjualan dimulai dan berfungsi sebagai penghubung untuk mengakses atau menggunakan bagian *game* lainnya.

A. Rancangan Game Design

Proses pengembangan dimulai dari *game design* dalam bentuk rancangan *wireframe* yang telah dibuat oleh *game designer* dan *UX designer* yang dapat dilihat pada Gambar 3.29.



Gambar 3.29. *Wireframe* Halaman Menu Utama Penjualan

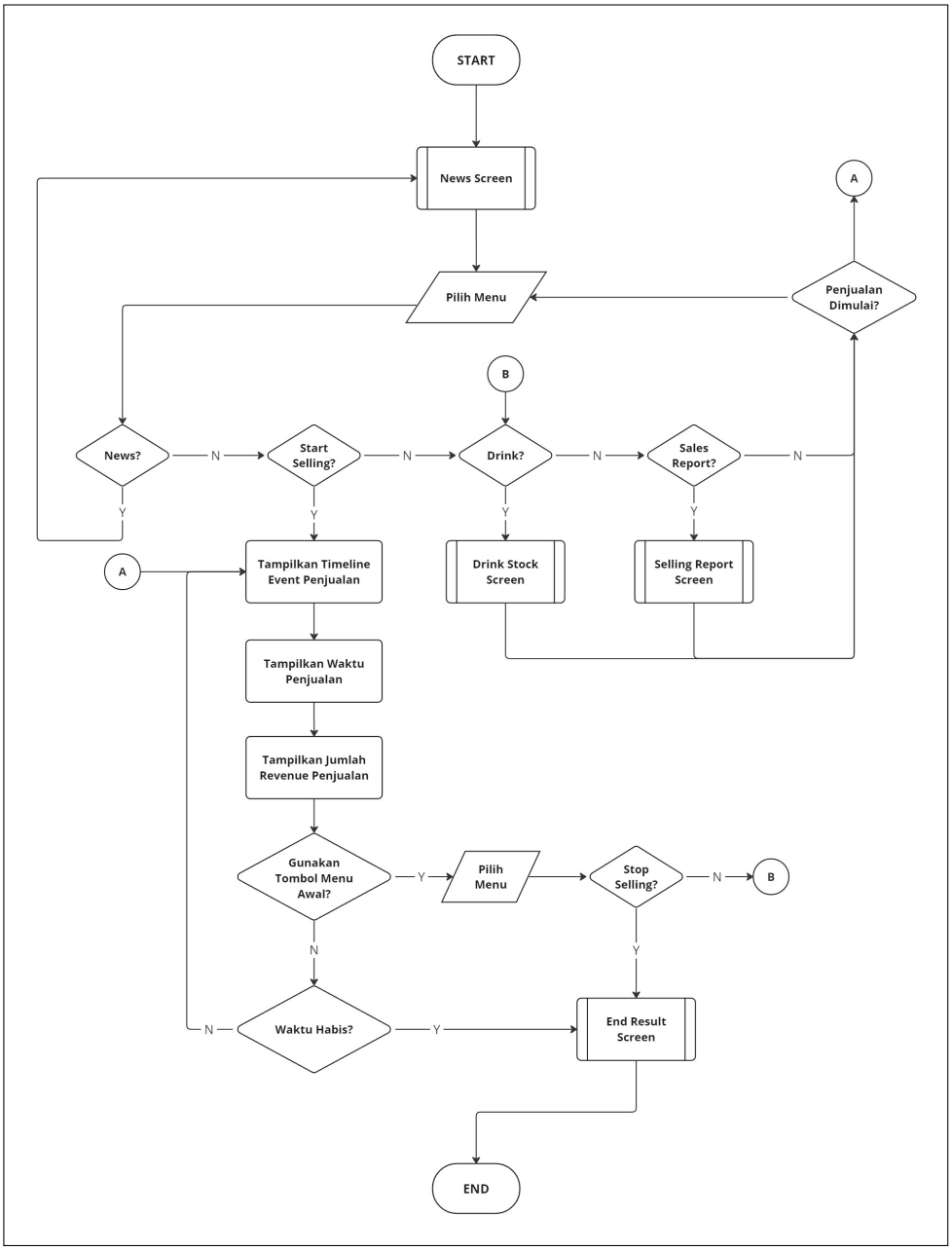
Sumber: UX Designer KUMAGEMA

Dari rancangan tersebut, dibuat sebuah *requirements* untuk membuat sistem yang dapat dijabarkan sebagai berikut:

- Interaksi dengan tombol menu.
- Pengaturan tampilan visual objek dan halaman pada menu utama penjualan.

B. Flowchart

Rancangan *game design* yang telah dibuat dapat digambarkan ke dalam bentuk *flowchart* yang dapat dilihat pada Gambar 3.30.

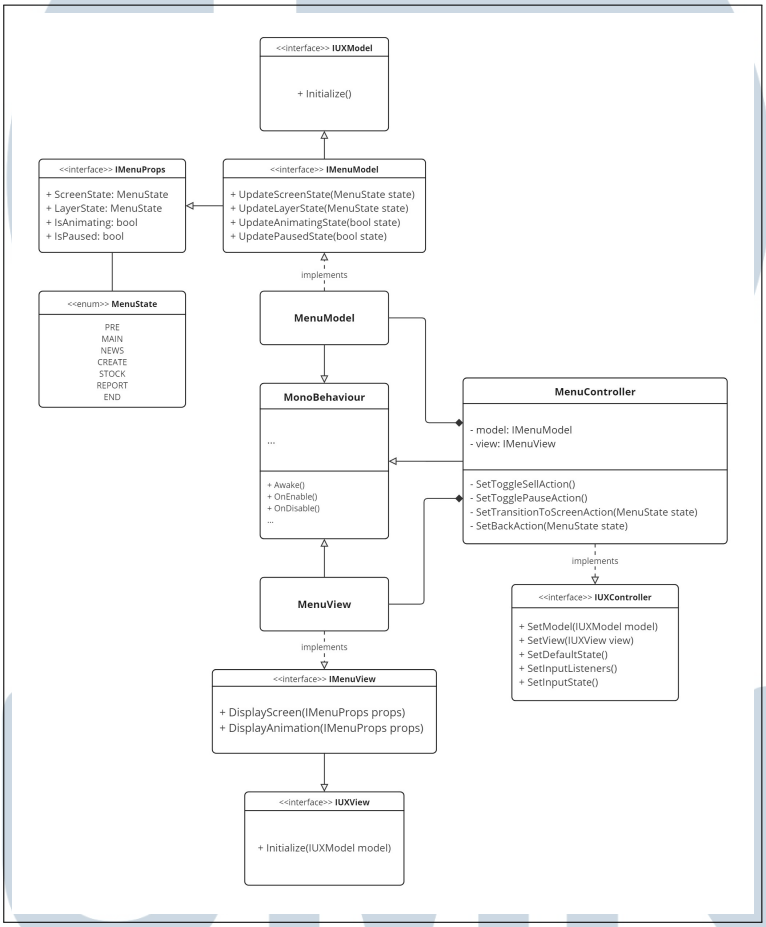


Gambar 3.30. *Flowchart* Halaman Menu Utama Penjualan

Pemain akan memilih tombol menu yang tersedia pada halaman menu utama penjualan dan sistem akan menampilkan halaman atau objek lainnya sesuai dengan *input* yang dilakukan oleh pemain. Jika pemain memilih menu yang berada pada bagian atas (*News, Drink, Sales Report*), maka sistem akan menampilkan halaman sesuai dengan menu yang dipilih. Jika pemain memilih menu *Start Selling*, maka sistem akan memulai *gameplay* utama dan akan menampilkan informasi penjualan yang berada pada halaman menu utama penjualan.

C. Class Diagram

Dari *flowchart* rancangan sistem yang telah dibuat, langkah selanjutnya adalah membuat *class* diagram untuk menggambarkan hubungan komponen di dalam sistem yang akan diterapkan ke dalam Unity yang dapat dilihat pada Gambar 3.31.

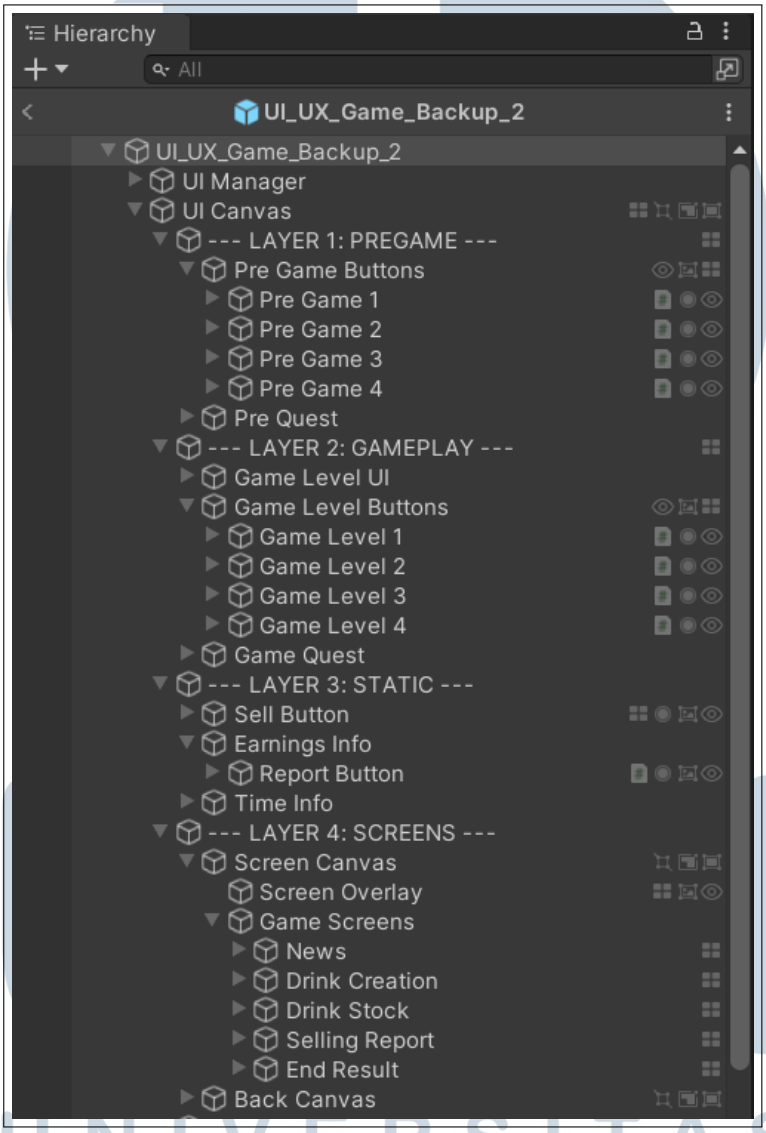


Gambar 3.31. *Class* Diagram Halaman Menu Utama Penjualan

Komponen utama di dalam sistem adalah *MenuModel*, *MenuView*, dan *MenuController*. *MenuController* akan mendapatkan informasi *input* yang dilakukan oleh pemain yang nantinya akan mengubah *MenuState* dan *state* lainnya pada *MenuModel* untuk halaman dan objek yang akan ditampilkan oleh *MenuView*. Sistem akan diinisialisasi saat memulai *game* atau digunakan oleh suatu bagian *game* lainnya menggunakan metode *Awake*, *OnEnable*, dan *OnDisable* yang dimiliki oleh *MonoBehaviour*.

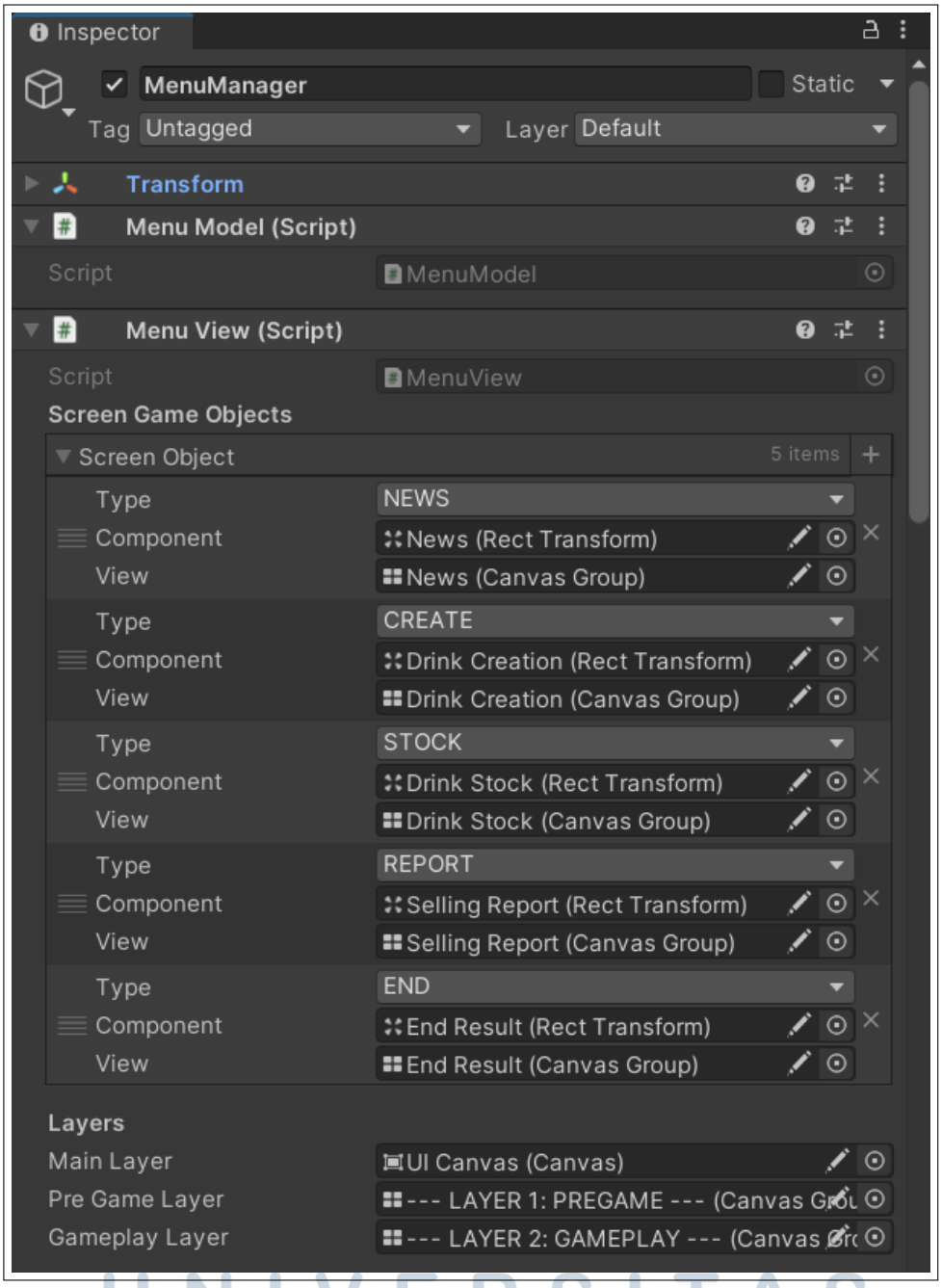
D. Pengaturan di Unity

Berikut adalah *Hierarchy* objek untuk sistem halaman menu utama penjualan yang dapat dilihat pada Gambar 3.32.

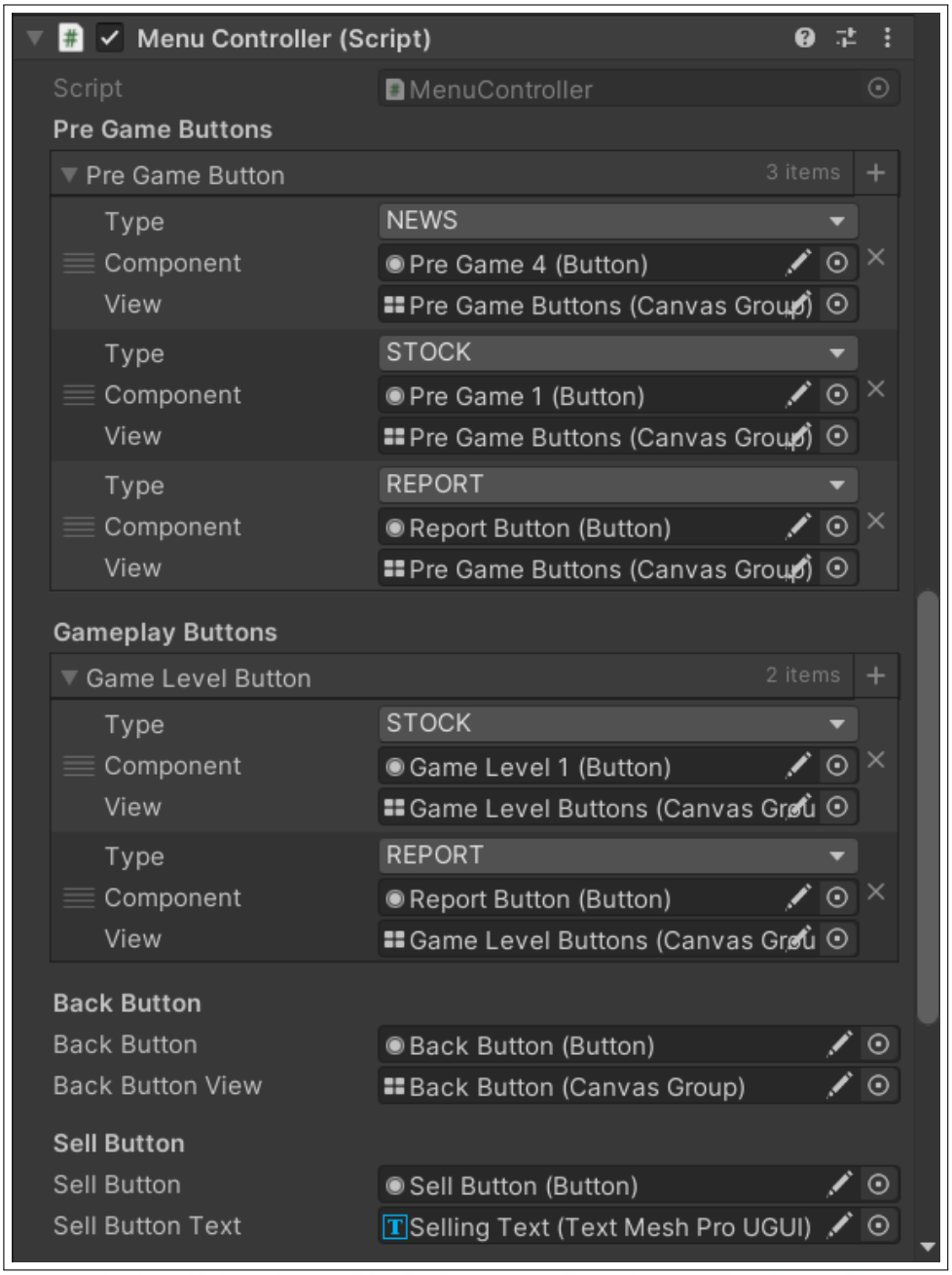


Gambar 3.32. *Hierarchy* Sistem Halaman Menu Utama Penjualan

Komponen *script* untuk sistem halaman menu utama penjualan dimasukkan ke dalam objek untuk diinisialisasi oleh Unity. *Script* juga dapat mengambil referensi objek lainnya yang akan diatur propertinya di dalam kode dan nilai-nilai yang dapat diatur melalui *Inspector* yang dapat dilihat pada Gambar 3.33 dan 3.34.



Gambar 3.33. Pengaturan Komponen Menu Utama Penjualan - *Model & View*



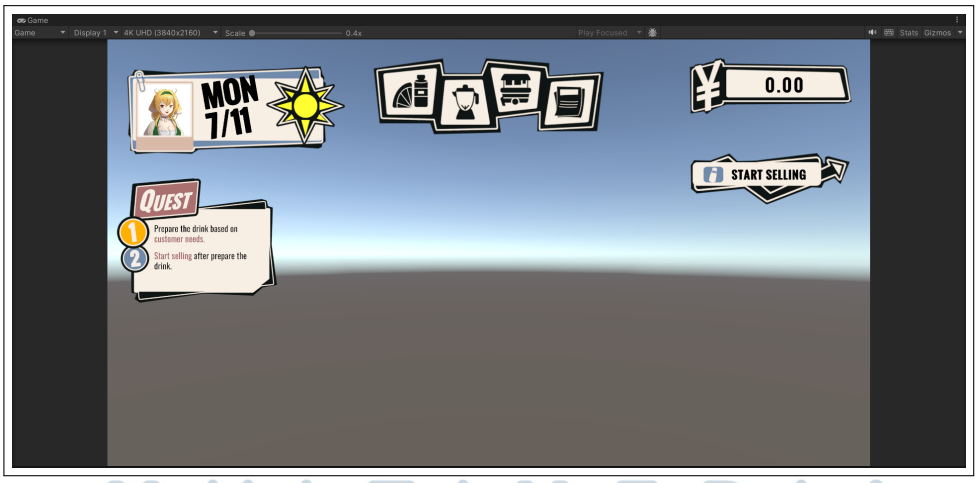
Gambar 3.34. Pengaturan Komponen Menu Utama Penjualan - *Controller*

Aset untuk sistem halaman menu utama penjualan yang akan diimplemenntasikan ke dalam *Scene* telah dibuat oleh 2D *Artist* yang dapat dilihat pada Gambar 3.35

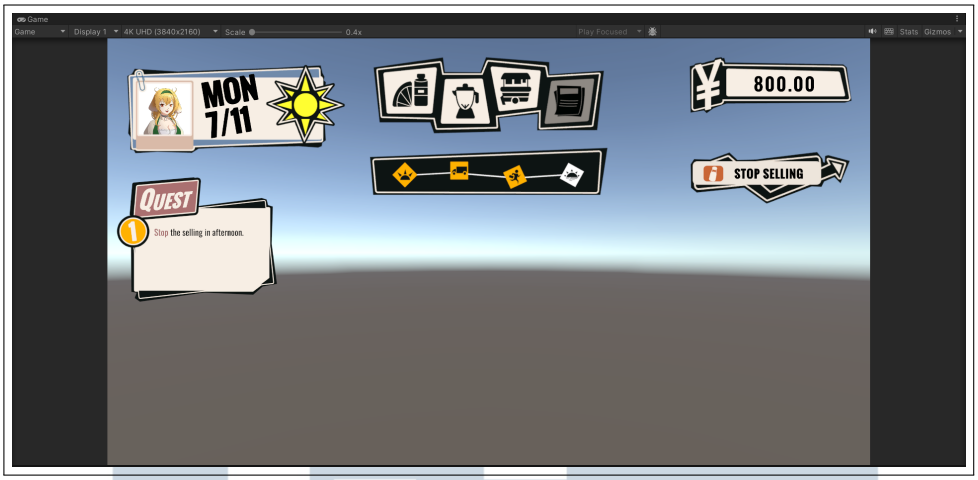
Gambar 3.35. Aset Halaman Menu Utama Penjualan

E. Hasil Implementasi

Berikut adalah hasil implementasi sistem halaman menu utama penjualan yang dapat dilihat pada Gambar 3.36, 3.37, dan 3.38.



Gambar 3.36. Hasil Implementasi Halaman Menu Utama Penjualan - Sebelum Memulai Penjualan



Gambar 3.37. Hasil Implementasi Halaman Menu Utama Penjualan - Saat Memulai Penjualan

Gambar 3.38. Hasil Implementasi Halaman Menu Utama Penjualan - Menampilkan Halaman Lain

3.4.6 Pengembangan Halaman Pembuatan Resep Minuman

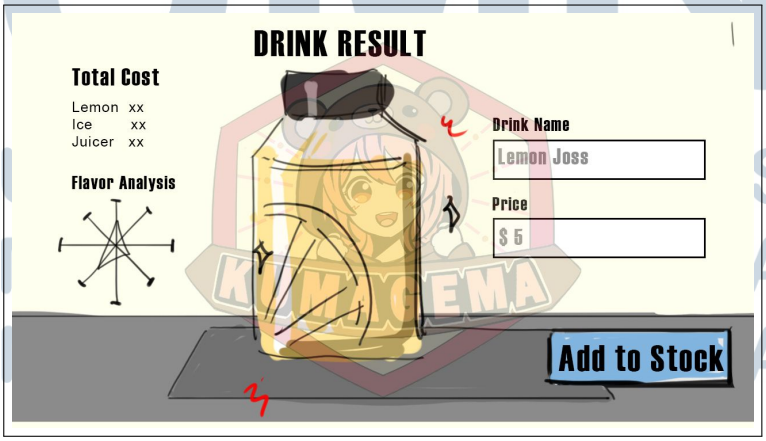
Halaman pembuatan resep minuman merupakan sistem *game* yang mengatur alur *gameplay* untuk proses pembuatan minuman. Pemain dapat memilih bahan-bahan yang tersedia untuk dicampurkan menggunakan suatu alat yang akan menghasilkan jenis minuman yang bervariasi.

A. Rancangan Game Design

Proses pengembangan dimulai dari *game design* dalam bentuk rancangan *wireframe* yang telah dibuat oleh *game designer* dan *UX designer* yang dapat dilihat pada Gambar 3.39 dan 3.40.

Gambar 3.39. *Wireframe* Halaman Pembuatan Resep Minuman - Halaman Utama

Sumber: UX Designer KUMAGEMA



Gambar 3.40. *Wireframe* Halaman Pembuatan Resep Minuman - Hasil Minuman

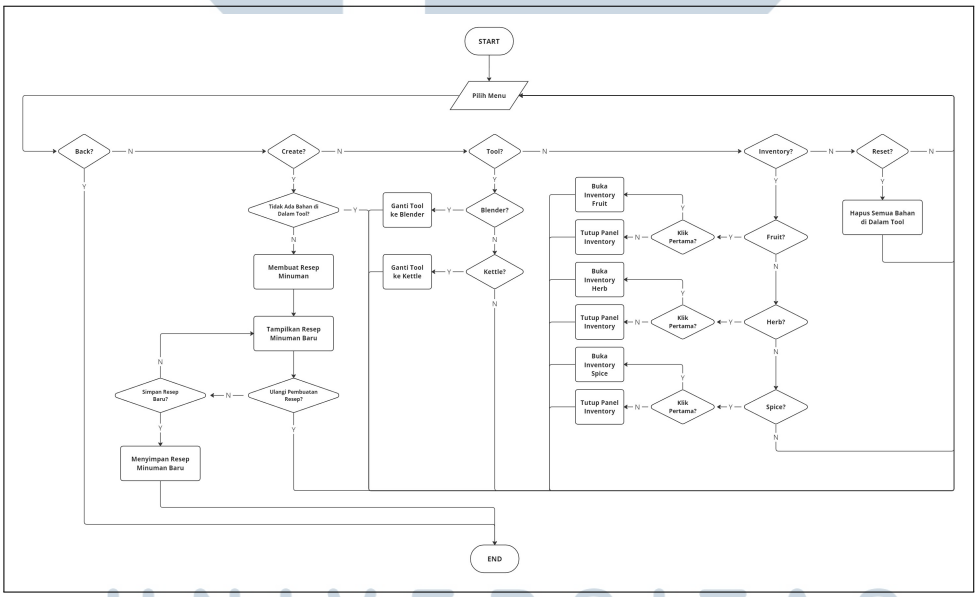
Sumber: UX Designer KUMAGEMA

Dari rancangan tersebut, dibuat sebuah *requirements* yang dibutuhkan untuk membuat sistem yang dapat dijabarkan sebagai berikut:

- Interaksi dengan tombol menu.
- Pengaturan tampilan visual halaman utama pembuatan minuman.
- Pengaturan tampilan visual *inventory* bahan.
- Pengaturan tampilan visual alat untuk membuat minuman.
- Pengaturan tampilan visual halaman hasil minuman.

B. Flowchart

Rancangan *game design* yang telah dibuat dapat digambarkan ke dalam bentuk *flowchart* yang dapat dilihat pada Gambar 3.41.



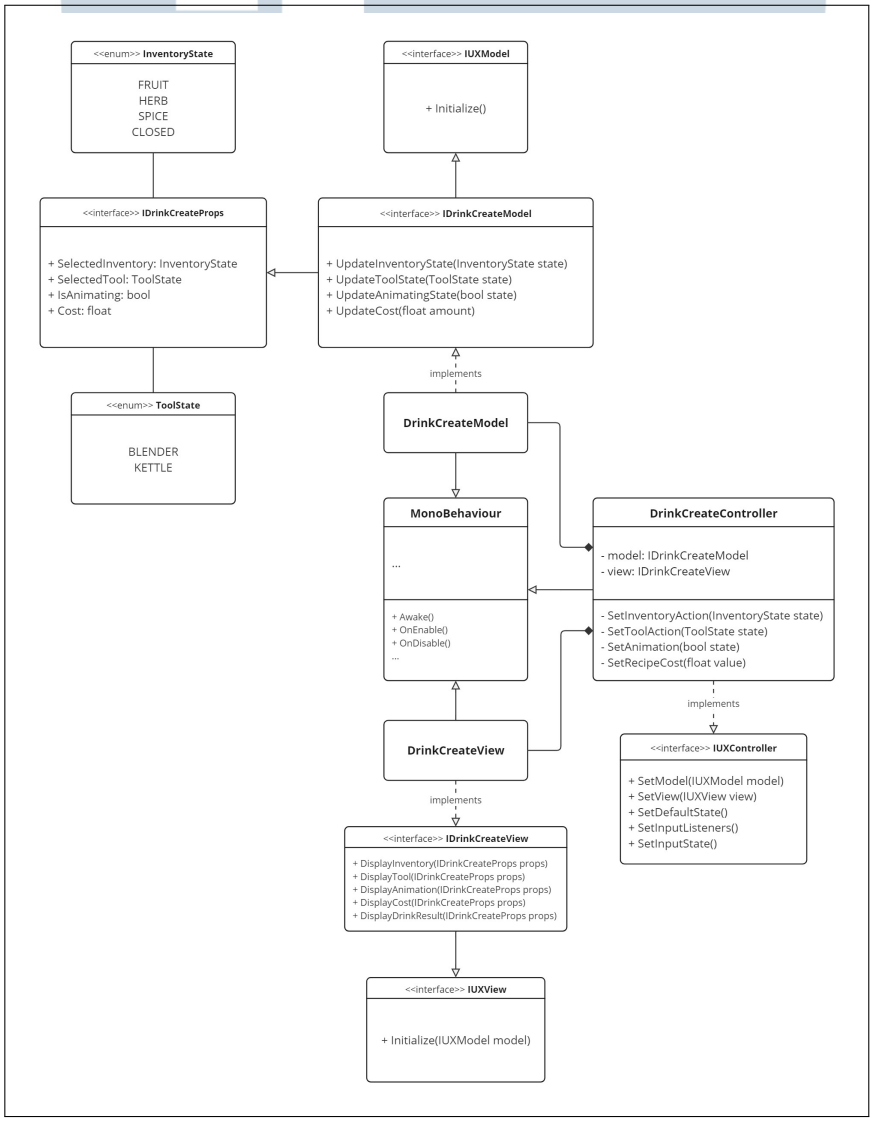
Gambar 3.41. *Flowchart* Halaman Pembuatan Resep Minuman

Pemain akan memilih tombol menu yang tersedia pada halaman pembuatan resep minuman dan sistem akan menampilkan halaman atau objek lainnya yang sesuai dengan *input* yang dilakukan oleh pemain. Jika pemain memilih menu *Inventory*, maka akan membuka jendela *inventory* sesuai dengan kategori yang dipilih (*Fruit, Herb, Spice*). Jika pemain memilih menu *Tool*, maka akan mengubah alat yang digunakan untuk membuat minuman (*Blender & Kettle*). Jika pemain

memilih menu *Reset*, maka bahan yang berada di dalam alat akan dihilangkan. Jika pemain memilih menu *Create*, maka akan membuat dan menampilkan hasil minuman.

C. Class Diagram

Dari *flowchart* rancangan sistem yang telah dibuat, langkah selanjutnya adalah membuat *class* diagram untuk menggambarkan hubungan komponen di dalam sistem yang akan diterapkan ke dalam Unity yang dapat dilihat pada Gambar 3.42.

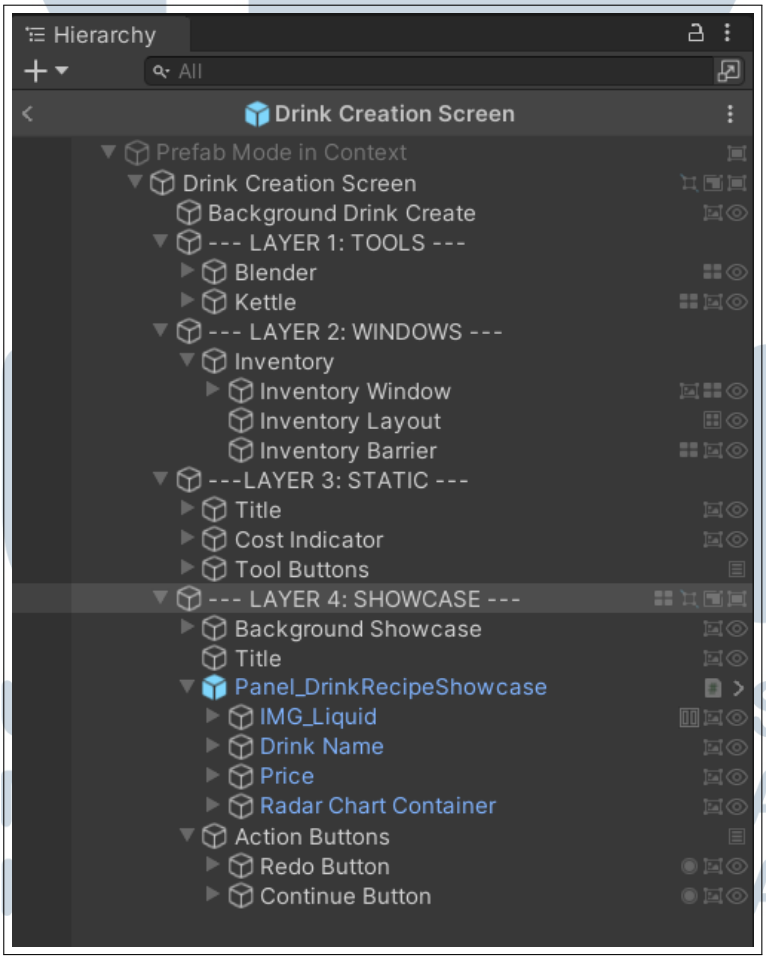


Gambar 3.42. *Class* Diagram Halaman Pembuatan Resep Minuman

Komponen utama di dalam sistem adalah *DrinkCreateModel*, *DrinkCreateView*, dan *DrinkCreateController*. *DrinkCreateController* akan mendapatkan informasi *input* yang dilakukan oleh pemain yang nantinya akan mengubah *InventoryState*, *ToolState*, dan *state* lainnya pada *DrinkCreateModel* untuk halaman dan objek yang akan ditampilkan oleh *DrinkCreateView*. Sistem akan diinisialisasi saat memulai *game* atau digunakan oleh suatu bagian *game* lainnya menggunakan metode *Awake*, *OnEnable*, dan *OnDisable* yang dimiliki oleh *MonoBehaviour*.

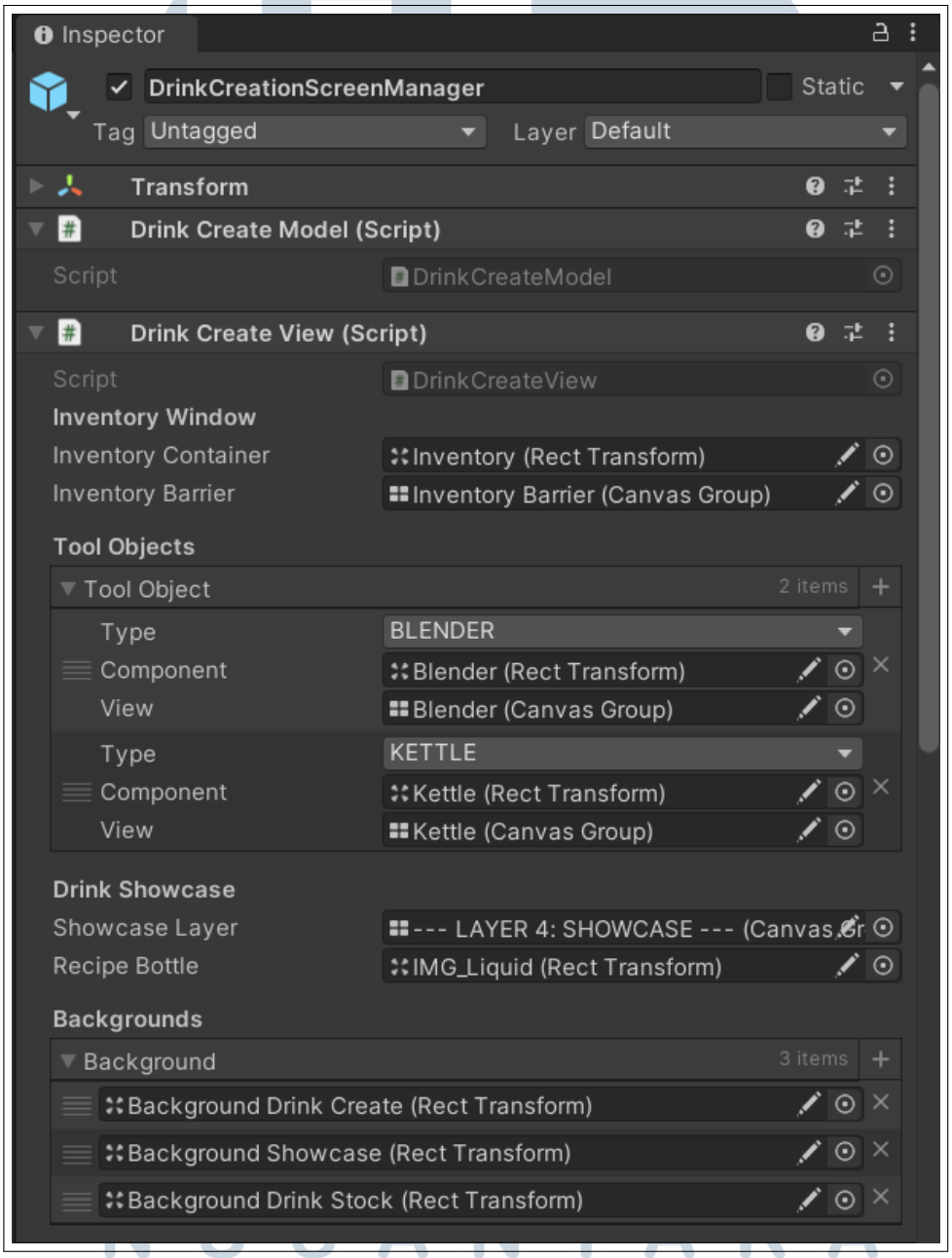
D. Pengaturan di Unity

Berikut adalah *Hierarchy* objek untuk sistem halaman pembuatan resep minuman yang dapat dilihat pada Gambar 3.43.

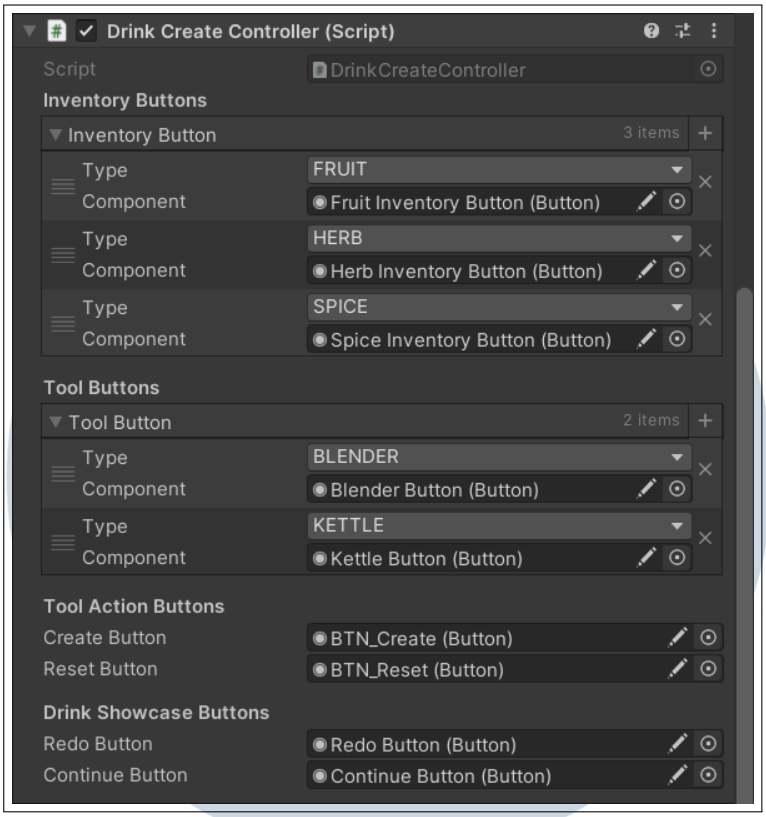


Gambar 3.43. *Hierarchy* Sistem Halaman Pembuatan Resep Minuman

Komponen *script* untuk sistem halaman pembuatan resep minuman dimasukkan ke dalam objek untuk diinisialisasi oleh Unity. *Script* juga dapat mengambil referensi objek lainnya yang akan diatur propertinya di dalam kode dan nilai-nilai yang dapat diatur melalui *Inspector* yang dapat dilihat pada Gambar 3.44 dan 3.45.



Gambar 3.44. Pengaturan Komponen Pembuatan Resep Minuman - *Model & View*



Gambar 3.45. Pengaturan Komponen Pembuatan Resep Minuman - *Controller*

Aset untuk sistem halaman pembuatan resep minuman yang akan diimplementasikan ke dalam *Scene* telah dibuat oleh 2D *Artist* yang dapat dilihat pada Gambar 3.46

Gambar 3.46. Aset Halaman Pembuatan Resep Minuman

E. Hasil Implementasi

Berikut adalah hasil implementasi sistem halaman menu utama penjualan yang dapat dilihat pada Gambar 3.47, 3.48, 3.49, 3.50, 3.51, dan 3.52.

Gambar 3.47. Hasil Implementasi Halaman Pembuatan Resep Minuman - Halaman Awal

Gambar 3.48. Hasil Implementasi Halaman Pembuatan Resep Minuman - Membuka *Inventory Fruit*

Gambar 3.49. Hasil Implementasi Halaman Pembuatan Resep Minuman - Membuka *Inventory Herb*

Gambar 3.50. Hasil Implementasi Halaman Pembuatan Resep Minuman - Membuka *Inventory Spice*

Gambar 3.51. Hasil Implementasi Halaman Pembuatan Resep Minuman - Menambahkan Bahan ke Dalam Alat

Gambar 3.52. Hasil Implementasi Halaman Pembuatan Resep Minuman - Hasil Minuman

3.5 Kendala dan Solusi yang Ditemukan

3.5.1 Kendala

Beberapa kendala yang dialami selama melaksanakan kegiatan kerja magang di KUMAGEMA adalah sebagai berikut:

1. Proses *loading* yang lama saat ingin menjalankan *Scene* untuk mengetes alur *game* sehingga membutuhkan waktu lebih untuk menunggu dan melihat hasil implementasi yang berdampak pada kurangnya produktivitas kerja.
2. Terdapat beberapa referensi objek yang hilang saat ingin mengintegrasikan satu sistem dengan sistem lainnya sehingga membutuhkan waktu lebih untuk mengatur referensi objek satu per satu. Proses pengaturan referensi objek juga membuat bingung karena tidak dapat mengetahui nama atau jenis objek yang sesuai untuk direferensikan pada *Inspector*.

3.5.2 Solusi

Berdasarkan kendala yang telah dijabarkan, berikut adalah solusi yang diterapkan untuk mengatasi beberapa kendala diatas:

1. Mengubah pengaturan proyek di dalam Unity untuk tidak memuat ulang *Scene* dan *Domain* proyek saat setiap kali menjalankan *Scene* agar langsung menampilkan isi *Scene* yang sedang aktif.
2. Memberikan nama objek yang mudah untuk diidentifikasi dan disesuaikan dengan sistem yang membutuhkan referensi objek tersebut serta menggunakan *tooltip* pada *Inspector* untuk mengetahui nama dan jenis objek yang sesuai untuk direferensikan ke dalam sistem.