BAB 2
LANDASAN TEORI

2.1 Obat Herbal

Obat herbal adalah obat yang dibuat dari bahan alam, baik tumbuhan, hewan atau mineral. Definisi obat herbal seringkali dicampuradukkan dengan obat tradisional. Obat tradisional adalah bahan atau ramuan bahan yang dapat berupa bahan tumbuhan, hewan, mineral atau campuran ketiganya yang sudah digunakan secara turun temurun untuk pengobatan. Obat tradisional sudah pasti obat herbal, tapi obat herbal belum tentu obat tradisional. Saat ini banyak ahli mengembangkan berbagai obat herbal baru, yang belum digunakan secara turun temurun, sehingga tidak dapat dikategorikan sebagai obat tradisional.

Perbedaan mendasar antara obat herbal dan obat kimia adalah bahwa obat herbal mengandung campuran berbagai zat kimia. Campuran tersebut dapat saling bersinergi ataupun memiliki efek antagonis antar komponen, yang pada akhirnya akan menimbulkan efek pada tubuh manusia. Efek sinergi untuk zat kimia yang bermanfaat tentunya menguntungkan karena dapat memperkuat efek terapi. Namun efek antagonis juga dapat menguntungkan karena kemungkinan dapat mengurangi efek merugikan dari zat kimia tertentu dalam satu obat herbal. Sementara itu obat kimia adalah obat yang mengandung satu zat kimia tunggal yang dapat bekerja sendiri dan menimbulkan efek [5].

2.2 Game

Game merupakan Permainan yang terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun dengan untuk memaksimalkan kemampuan sendiri atau pun meminimalkan kemenangan lawan (Neumann, 1953). Definisi game menurut Agustinus Nilwan Game merupakan permainan komputer yang dibuat dengan teknik dan metode animasi (Nilwan, 1995). Pengertian game yang lain

menurut Foreman game merupakan potential learning environment. Bermain game merupakan sebuah litelatur baru dalam pendidikan (Foreman, 2007) [6].

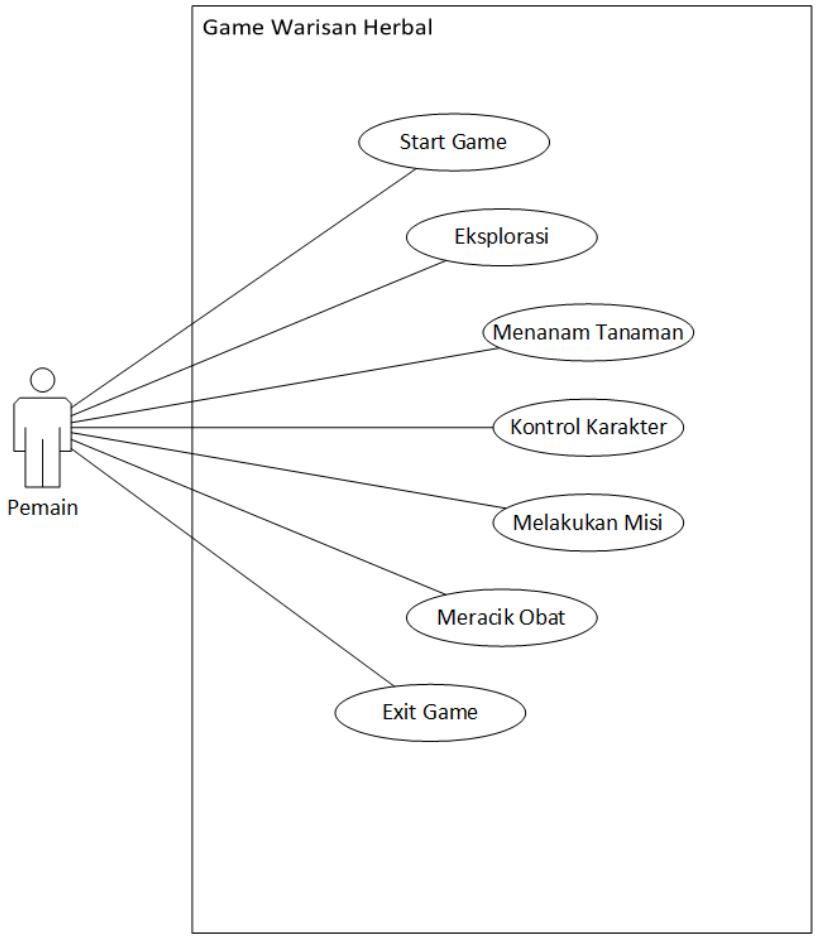
Game edukasi adalah jenis game yang tidak hanya dapat menghibur tetapi juga dapat mengajar karena pengguna dapat bermain sambil belajar. Game, menurut Greg Costikyan, dapat didefinisikan sebagai karya seni di mana pemain, yang disebut pemain, membuat keputusan untuk menggunakan benda-benda di dalam game untuk mengendalikan sumber daya yang dimilikinya untuk mencapai tujuan tertentu. Permainan didefinisikan oleh Joan Freeman dan Utami Munandar sebagai suatu kegiatan yang membantu anak berkembang secara fisik, intelektual, sosial, moral, dan emosional. Marc Prensky menyatakan bahwa game edukasi adalah game yang dirancang untuk belajar, tetapi juga digunakan untuk bermain dan bersenang-senang. Menurutnya, game edukasi adalah gabungan dari konten edukasi, prinsip pembelajaran, dan game komputer. Game jenis ini biasanya digunakan untuk mengajak penggunanya belajar sambil bermain, dan melalui proses belajar ini, penggunanya dapat belajar sambil bermain. Game edukasi juga dapat memberikan ilmu pengetahuan kepada anak melalui proses pembelajaran, merangsang perkembangan daya pikir dan daya cipta, menciptakan lingkungan bermain yang menarik, memberikan rasa aman dan nyaman, dan meningkatkan kualitas pembelajaran anak.

Menurut José P. Zagal dan Sebastian Deterding, “role-playing games” (RPGs) adalah istilah yang digunakan oleh berbagai kelompok sosial untuk merujuk pada berbagai bentuk dan gaya aktivitas bermain serta objek yang melibatkan penciptaan dan peran karakter dalam dunia fiksi yang diatur oleh aturan. Meskipun ada banyak variasi dalam bentuk dan gaya RPG, beberapa karakteristik umum yang sering muncul adalah aktivitas bermain dan objek yang berpusat pada penciptaan karakter dan peran dalam dunia fiksi yang diatur oleh aturan [7]. Sejarah game *RPG* dimulai pada awal abad ke-20 dengan pengembangan permainan peran dan simulasi militer. Namun, penciptaan yang paling berpengaruh dalam sejarah *RPG* adalah permainan *Dungeons & Dragons (D&D)* yang dirilis pada tahun 1974 oleh Gary Gygax dan Dave Arneson, menjadi tonggak dalam pengembangan genre

RPG modern. Pada tahun 1970-an dan 1980-an, *RPG* meja seperti *D&D* menjadi populer di kalangan penggemar permainan, sementara pada saat yang sama, teknologi komputer semakin maju, membawa munculnya *RPG* komputer, seperti *Ultima, Wizardry*, dan *Final Fantasy*. Ini membawa munculnya *MMO* (*Massively Multiplayer Online*) *RPG*, seperti *EverQuest* dan *World of Warcraft*, yang memungkinkan ribuan pemain untuk berinteraksi dalam dunia virtual bersama-sama secara online. Seiring waktu, genre RPG terus berkembang dengan adopsi berbagai inovasi dalam permainan dan teknologi, tetapi tetap menjadi salah satu genre yang paling dicintai dan berpengaruh dalam industri permainan [8].

2.3 Usecase

Usecase adalah sebuah konsep dalam rekayasa perangkat lunak yang digunakan untuk mendeskripsikan interaksi antara sistem dan entitas atau aktor yang terlibat dalam suatu skenario tertentu. Dalam landasan teorinya, usecase memberikan gambaran tentang bagaimana sistem berperilaku dalam situasi nyata atau skenario penggunaan tertentu. Usecase membantu mengidentifikasi fungsionalitas utama sistem dan menyajikan informasi secara terstruktur mengenai langkah-langkah yang harus diambil oleh sistem untuk memenuhi kebutuhan pengguna atau aktor tertentu. Dengan menganalisis usecase, pengembang perangkat lunak dapat memahami secara rinci bagaimana interaksi antara pengguna dan sistem harus terjadi, sehingga memudahkan dalam perancangan dan pengembangan sistem yang efisien dan sesuai dengan kebutuhan pengguna [9]. Berikut adalah contoh usecase pada gambar 2.1.



Gambar 2.1 Contoh Usecase Diagram

Berikut ini adalah beberapa komponen utama dalam use case diagram:

1. Sistem digambarkan ke dalam bentuk persegi panjang. Fungsinya untuk membatasi use case dengan interaksi dari luar sistem.
2. Aktor adalah orang atau sistem lain yang berinteraksi dengan sistem yang akan dibuat.
3. Usecase digambarkan dengan elips digunakan untuk interaksi antar aktor dan sistem.
4. *Association* adalah penghubung antara aktor dengan usecase.
5. *Generalization* digunakan untuk menunjukan spesialisasi aktor .
6. *Include* digunakan untuk menunjukan bahwa salah satu usecase merupakan fungsionalitas usecase lain.

7. *Extend* digunakan untuk menunjukan bahwa salah satu usecase merupakan tambahan fungsionalitas usecase lain jika syarat terpenuhi.

2.4 Usecase Scenario

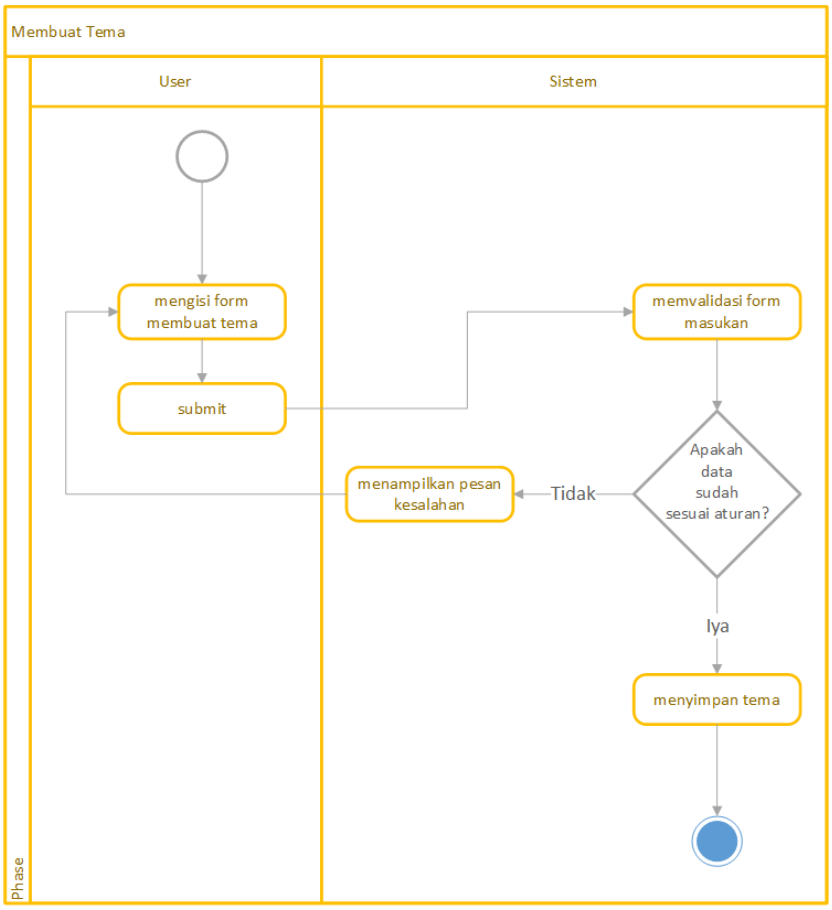
Usecase scenario, dalam konteks pengembangan perangkat lunak, merujuk pada deskripsi lebih lanjut dari suatu use case. Scenario ini memberikan gambaran yang lebih mendetail tentang bagaimana suatu sistem berinteraksi dengan aktor atau entitas tertentu dalam suatu situasi atau kejadian khusus. Secara sederhana, *usecase scenario* memberikan narasi tentang langkah-langkah atau aksi yang terjadi dalam rangka mencapai tujuan tertentu yang diinginkan oleh pengguna atau aktor sistem. Contoh *usecase scenario* bisa melibatkan urutan langkah-langkah, interaksi antara pengguna dan sistem, serta respons sistem terhadap masukan atau kejadian tertentu. *Scenario* ini membantu dalam pemahaman lebih lanjut tentang bagaimana sistem berperilaku dalam konteks penggunaan nyata. Berikut adalah contoh *usecase scenario* pada gambar 2.2.

Use case name:	MenghapusTema
Primary Actor:	User
Secondary Actor:	Sistem
Goal in contex:	Tema berhasil dihapus
Precondition:	Pengguna sudah login
Successfull End Condition:	Pengguna berhasil menghapus tema
Scenario:	1. Pengguna menekan tombol hapus
	2. Sistem menghapus
Exception:	2.1. Sistem gagal menghapus tema
	2.2. Sistem menampilkan notifikasi gagal

Gambar 2.2 Contoh Usecase Scenario

2.5 Activity Diagram

Activity Diagram adalah salah satu jenis diagram *UML (Unified Modeling Language)* yang digunakan untuk memodelkan aktivitas, proses, dan alur kerja dalam suatu sistem. Diagram ini membantu dalam menggambarkan serangkaian aktivitas atau tindakan yang terjadi dalam suatu proses atau use case tertentu. *Activity diagram* memvisualisasikan alur kerja dari suatu kegiatan, menunjukkan urutan aktivitas, pengambilan keputusan, dan percabangan dalam proses. Berikut adalah contoh *activity diagram* pada gambar 2.2



Gambar 2.3 Contoh Activity Diagram

Beberapa elemen penting dalam activity diagram meliputi:

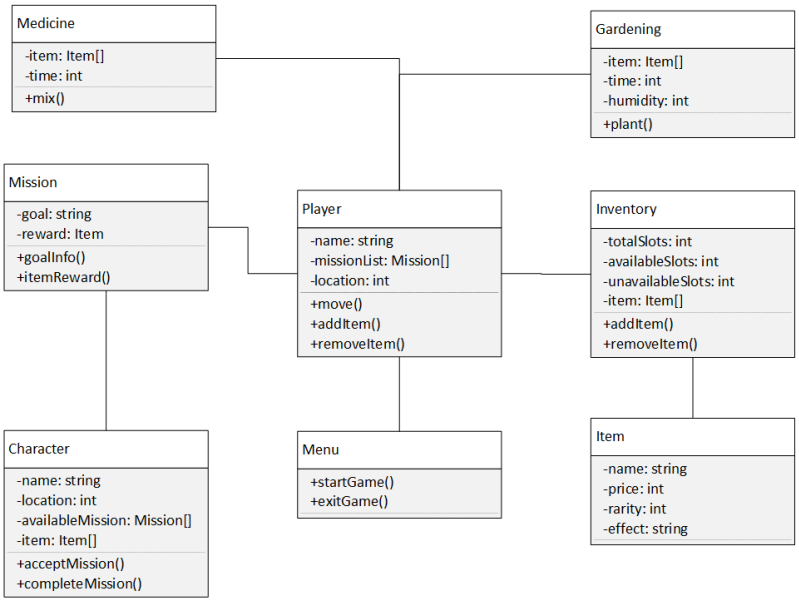
1. *Activity:* Representasi dari tindakan atau langkah-langkah dalam proses. Biasanya, aktivitas direpresentasikan dengan kotak elips.

2. *Flow Edge:* Panah yang menghubungkan aktivitas-aktivitas dan menunjukkan urutan atau alur kerja. Garis alur mengindikasikan bagaimana suatu aktivitas mengarah ke aktivitas berikutnya.
3. *Decision:* Menunjukkan percabangan dalam alur kerja. Pada titik ini, sistem membuat keputusan dan memilih antara beberapa jalur yang berbeda.
4. *Merge:* Menunjukkan titik di mana beberapa jalur yang berbeda kembali bergabung setelah percabangan.
5. *Fork and Join: Fork* menunjukkan titik di mana alur kerja bercabang menjadi beberapa jalur. *Join* (bergabung) menunjukkan titik di mana jalur-jalur yang berbeda kembali bersatu.
6. *Swimlane:* Terkadang, diagram aktivitas dapat dibagi menjadi swimlane, yaitu bagian yang menunjukkan pemilik atau pemegang tanggung jawab dari setiap aktivitas.

Activity diagram berguna untuk memahami dan menggambarkan urutan langkah-langkah dalam suatu proses, sehingga membantu dalam analisis dan perancangan sistem dengan lebih jelas [10].

2.6 Class Diagram

Diagram kelas (*class diagram*) adalah jenis diagram *UML (Unified Modeling Language)* yang digunakan dalam rekayasa perangkat lunak untuk memodelkan struktur statis dari suatu sistem, dengan fokus pada kelas dan hubungan antar kelas. Diagram kelas memberikan gambaran tentang entitas-entitas yang ada dalam sistem, atribut-atribut yang dimiliki oleh entitas tersebut, dan hubungan antar entitas. Berikut adalah contoh *class diagram* pada gambar 2.3.



Gambar 2.4 Contoh Class Diagram

Elemen-elemen utama dalam diagram kelas meliputi:

1. *Class:* Mewakili suatu jenis objek atau entitas dalam sistem. Kelas menggambarkan struktur data dan perilaku yang dimiliki oleh objek tersebut. Kelas direpresentasikan dalam diagram oleh kotak dengan tiga bagian: nama kelas, atribut-atribut, dan metode-metode.

2. *Attribute:* Menunjukkan data atau informasi yang dimiliki oleh suatu kelas. Atribut direpresentasikan oleh nama atribut diikuti oleh tipe data.

3. *Method:* Merupakan tindakan atau fungsi yang dapat dilakukan oleh suatu kelas. Metode direpresentasikan oleh nama metode diikuti oleh parameter dan tipe kembalian (jika ada).

4. *Association:* Menunjukkan hubungan antara dua atau lebih kelas. Asosiasi menggambarkan bagaimana objek dari satu kelas berhubungan dengan objek dari kelas lain.

5. Agregasi dan Komposisi: Merupakan bentuk asosiasi yang menunjukkan hubungan bagian-keseluruhan antara kelas-kelas. Agregasi menunjukkan

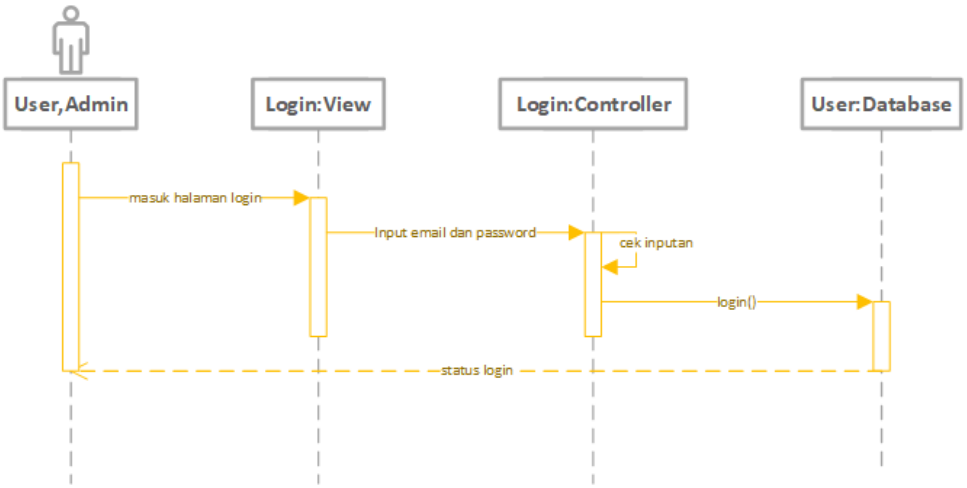
hubungan yang lebih longgar, sedangkan komposisi menunjukkan hubungan yang lebih kuat.

6. *Inheritance:* Menunjukkan hubungan hierarki antara kelas-kelas. Kelas yang mewarisi disebut kelas anak atau subclass, sementara kelas yang memberikan warisan disebut kelas induk atau superclass.

Diagram kelas membantu dalam merancang dan memahami struktur statis dari suatu sistem perangkat lunak. Diagram ini juga digunakan sebagai landasan untuk implementasi kode program dalam berbagai bahasa pemrograman [11].

2.7 Sequence Diagram

Diagram Sequence (*sequence diagram*) adalah jenis diagram *UML (Unified Modeling Language)* yang digunakan untuk menggambarkan interaksi antara objek- objek dalam suatu skenario atau proses tertentu. Diagram ini menunjukkan urutan pesan atau panggilan metode yang terjadi antara objek-objek selama eksekusi suatu use case atau scenario. Berikut adalah contoh *sequence diagram* pada gambar 2.4



Gambar 2.5 Contoh Sequence Diagram

Elemen-elemen utama dalam diagram sequence meliputi:

1. *Object:* Mewakili instance dari suatu kelas. Objek direpresentasikan dengan sebuah kotak yang berisi nama objek dan garis hidup (lifeline) yang menunjukkan durasi hidup objek selama proses.
2. *Message:* Menunjukkan panggilan metode atau pertukaran pesan antara objek-objek. Pesan direpresentasikan dengan garis panah yang menghubungkan objek-objek, dan dapat diberi label untuk menunjukkan nama metode yang dipanggil.
3. *Lifeline:* Menunjukkan durasi hidup objek selama proses. Garis hidup direpresentasikan oleh garis vertikal yang menghubungkan objek dengan garis waktu (time axis).
4. *Activation:* Menunjukkan periode waktu ketika suatu objek sedang melakukan operasi atau merespon pesan. Aktivasi direpresentasikan oleh kotak yang menggantikan garis hidup dan menunjukkan waktu di mana objek sedang aktif.
5. *Interaction Fragment:* Bagian dari diagram yang menunjukkan kondisi, percabangan, atau iterasi dalam urutan interaksi. Contoh fragment interaksi adalah fragment opsional (optional) dan loop.

Diagram sequence membantu dalam memodelkan dan memahami bagaimana objek-objek saling berinteraksi dan bertukar pesan selama eksekusi suatu proses atau use case. Diagram ini sering digunakan dalam fase analisis dan desain untuk menggambarkan dinamika eksekusi sistem secara visual [12].

2.8 RPG Maker MV

Dalam penelitian ini pembangunan game menggunakan *RPG Maker MV*. *RPG Maker*, dikenal sebagai *RPG Tsukūru* di Jepang, adalah serangkaian perangkat lunak pengembangan game role-playing (RPG) yang awalnya dikembangkan oleh kelompok ASCII Jepang dan kemudian diambil alih oleh Enterbrain. Nama Jepang "*Tsukūru*" menggabungkan kata "*tsukuru,*" yang berarti "membuat" atau "menciptakan," dengan "*tsūru*," transkripsi bahasa Jepang dari kata bahasa Inggris "*tool*.". *RPG Maker* untuk *PC* memungkinkan pengguna untuk membuat permainan

video *RPG* mereka sendiri dengan editor grafis dan skenario peristiwa yang mudah digunakan. Setiap versi termasuk tilesets, karakter, dan skenario yang dapat digunakan dalam membuat game baru. Meskipun ditujukan untuk menciptakan *RPG, RPG Maker* juga dapat digunakan untuk membuat game dari genre lain, seperti petualangan atau novel grafis. *RPG Maker MV* merupakan versi terbaru dalam seri *RPG Maker*, dirilis pada tahun 2015. *RPG Maker MV* menggunakan Javascript sebagai bahasa pemrograman dan mendukung resolusi tinggi serta dukungan multi-OS. Ini juga memperkenalkan kembali fitur layered tilesets yang telah dihapus di versi sebelumnya.

RPG Maker telah mengalami beberapa versi, termasuk *RPG Tsukūru Dante 98, RPG Maker 95, RPG Maker 2000, RPG Maker 2003, RPG Maker XP, RPG Maker VX, RPG Maker VX Ace, dan RPG Maker MV*. Setiap versi memiliki fitur dan perbaikan yang berbeda, mulai dari peningkatan resolusi hingga kemampuan scripting yang lebih kuat. Selain versi untuk *PC, RPG Maker* juga memiliki versi untuk konsol, dengan *RPG Tsukūru Super Dante* yang dirilis untuk *Super Famicom* sebagai port dari *RPG Tsukūru Dante 98.* Beberapa versi *RPG Maker* memiliki rilis resmi dalam bahasa Inggris, sementara versi lainnya telah diterjemahkan dan didistribusikan secara tidak resmi oleh komunitas. Degica adalah penerbit resmi *RPG Maker* untuk versi bahasa Inggris, yang merilis *RPG Maker XP, RPG Maker VX, dan RPG Maker VX Ace*. Berikut adalah beberapa keunggulan dari *RPG Maker MV* dibanding versi lain:

1. Dukungan Kontrol Layar Sentuh dan Mouse: RPG Maker MV mendukung kontrol layar sentuh dan mouse, memudahkan pengguna dalam berinteraksi dengan game.
2. Peningkatan Batas Maksimal Database: RPG Maker MV memiliki peningkatan batas maksimal database, memungkinkan pengguna untuk menyimpan lebih banyak data dalam game.
3. Pembagian Peta Menjadi Tiga Lapis: *RPG Maker MV* membagi peta menjadi tiga lapis, memberikan lebih banyak fleksibilitas dalam mendesain peta.

4. Dukungan Resolusi Tinggi: *RPG Maker MV* mendukung resolusi tinggi, memungkinkan game untuk ditampilkan dengan lebih jelas dan detail.
5. *Fitur Character Generator Parts*: RPG Maker MV dilengkapi dengan fitur character generator parts and more yang memungkinkan pengguna untuk dengan mudah dan cepat membuat karakter2.
6. *JavaScript dan HTML5 Combination Export*: *RPG Maker MV* menggunakan JavaScript sebagai bahasa pemrograman dan mendukung ekspor kombinasi HTML5, memungkinkan game untuk dijalankan di berbagai platform.
7. *Simple and Intuitive Event System*: *RPG Maker MV* memiliki sistem event yang sederhana dan intuitif, memudahkan pengguna dalam membuat puzzle dan quest.

Dengan keunggulan-keunggulan ini, *RPG Maker MV* menjadi alat yang sangat berguna bagi pengembang game, terutama bagi mereka yang ingin membuat game *RPG* tanpa memerlukan keterampilan pemrograman yang tinggi. Selain itu, *RPG Maker MV* juga dilengkapi dengan berbagai fitur kreatif yang memungkinkan pengguna untuk menciptakan pengalaman permainan yang unik dan menarik. *RPG Maker* telah menjadi alat yang populer di kalangan pengembang game indie untuk membuat game *RPG* tanpa memerlukan keterampilan pemrograman yang tinggi, dengan menyediakan antarmuka yang intuitif dan berbagai fitur kreatif untuk menciptakan pengalaman permainan yang unik [13] [14].

2.9 Black Box Testing

Black Box Testing, sebuah bentuk pengujian yang dilakukan untuk mengevaluasi fungsionalitas, keamanan, kinerja, dan aspek lainnya dari game. Dynamic code analysis adalah contoh dari automated black box security testing. Evaluator black box mendefinisikan test cases dan berinteraksi dengan perangkat lunak seperti pengguna untuk memvalidasi bahwa perangkat lunak berfungsi sebagaimana mestinya. Pengujian ini dapat dirancang untuk mencapai beberapa tujuan berbeda, termasuk:

1. *Functional Testing*: Bertujuan untuk memvalidasi bahwa aplikasi melakukan apa yang seharusnya dilakukan.
2. *Non-Functional Testing*: Mengevaluasi seberapa baik aplikasi melaksanakan fungsi intinya.
3. *Regression Testing*: Dirancang untuk memastikan bahwa perubahan pada aplikasi tidak merusak fungsionalitas.

Beberapa teknik umum untuk melakukan evaluasi black box meliputi:

A. *Equivalence Class Testing*: Mengidentifikasi kelas-kelas input yang menghasilkan hasil yang sama dan hanya menguji satu nilai dalam kelas tersebut.
B. *Boundary Value Evaluation*: Menguji input di mana aplikasi berubah dari satu alur kontrol ke alur lainnya untuk memastikan bahwa sistem menangani kasus tepi ini dengan benar.
C. *Decision Table Testing*: Melibatkan enumerasi setiap kombinasi input dan hasil yang diharapkan serta mengembangkan test case untuk memvalidasi setiap kombinasi.
D. *State Transition Evaluation*: Mengidentifikasi situasi di mana aplikasi mengubah status dan mengembangkan test case untuk memvalidasinya.
E. *Error Checking*: Menguji kesalahan umum yang mungkin dibuat pengembang saat membuat aplikasi, sering kali berkisar pada sanitasi input dan memastikan bahwa asumsi tentang input ditegakkan.

Kesimpulannya adalah Black box testing adalah metode pengujian yang dilakukan dengan tujuan untuk mengevaluasi berbagai aspek aplikasi seperti fungsionalitas, keamanan, dan kinerja. Pengujian ini mencakup beberapa jenis, yaitu functional testing untuk memastikan aplikasi berfungsi sesuai yang diharapkan, non-functional testing untuk mengevaluasi kinerja aplikasi, dan regression testing untuk memastikan perubahan tidak merusak fungsionalitas yang ada [15].

2.10 SPSS

SPSS, yang merupakan singkatan dari Statistical Package for the Social Sciences, adalah perangkat lunak untuk pengolahan data statistik yang sering digunakan dalam analisis statistik interaktif maupun batch. Awalnya, SPSS banyak digunakan dalam ilmu sosial, namun kini penggunaannya telah meluas ke berbagai disiplin ilmu. Perangkat lunak ini banyak dipakai oleh peneliti pasar, perusahaan survei, peneliti kesehatan, pemerintah, peneliti pendidikan, dan berbagai organisasi pemasaran. SPSS menawarkan kemampuan analisis statistik yang tinggi dan sistem manajemen data yang terintegrasi dalam lingkungan grafis, menggunakan menu-menu deskriptif dan kotak dialog yang sederhana, sehingga mudah digunakan.

SPSS pertama kali dirilis pada tahun 1968 oleh Norman Nie, seorang lulusan Fakultas Ilmu Politik dari Stanford University yang kini menjadi Profesor Peneliti di Fakultas Ilmu Politik Stanford dan Profesor Emeritus di University of Chicago. Pada awalnya, SPSS hanya digunakan untuk ilmu sosial, tetapi kemudian berkembang untuk berbagai bidang ilmu lainnya, sehingga singkatannya berubah menjadi "Statistical Product and Service Solutions." SPSS sangat membantu dalam menyajikan data dalam berbagai bentuk, seperti gambar, grafik, diagram, plot, serta statistik deskriptif dan analisis yang kompleks. Selain itu, SPSS mampu mengelola data dalam jumlah besar.

Saat SPSS dijalankan, terdapat dua jendela utama, yaitu data editor dan output viewer. Data editor adalah jendela untuk memasukkan data yang akan diolah, sementara output viewer menampilkan hasil pengolahan data tersebut. Data yang akan dianalisis dimasukkan melalui menu data editor, yang muncul secara otomatis saat SPSS dijalankan. Setelah diproses, hasilnya ditampilkan di jendela output viewer yang terpisah [16].

2.11 Skala Likert

Skala Likert merupakan metode pengukuran yang sering digunakan dalam penelitian untuk menilai sikap, pengetahuan, opini, dan persepsi responden terhadap suatu objek melalui angket atau kuesioner. Skala ini dikembangkan oleh Rensis Likert pada tahun 1932 dan menjadi alat yang umum dalam mengukur

berbagai tanggapan responden. Skala Likert biasanya hadir dalam beberapa pilihan, seperti:

1. Skala 5 Poin: Sangat tidak setuju, Tidak setuju, Netral, Setuju, Sangat setuju.
2. Skala 7 Poin: Sangat tidak setuju, Tidak setuju, Sedikit tidak setuju, Netral, Sedikit setuju, Setuju, Sangat setuju.

Oleh karena itu, Skala Likert adalah alat yang efektif dalam mengumpulkan data kuantitatif terkait sikap, opini, atau persepsi individu atau kelompok. Namun, penting untuk memastikan bahwa hasilnya ditafsirkan dengan cermat agar tidak terjadi kesalahan dalam interpretasi [17].

2.12 Shapiro Wilk

Uji Shapiro-Wilk adalah metode statistik yang digunakan untuk mengevaluasi apakah sebuah sampel berasal dari distribusi normal. Metode ini dikembangkan oleh Samuel Shapiro dan Martin Wilk pada tahun 1965 dan merupakan salah satu alat yang paling efektif untuk menguji normalitas distribusi, terutama ketika ukuran sampel kecil (50 sampel atau kurang). Uji ini sering dipilih karena kemampuannya yang tinggi dalam mendeteksi deviasi dari normalitas pada sampel kecil, dibandingkan dengan uji normalitas lainnya seperti uji Kolmogorov-Smirnov.

Dalam uji Shapiro-Wilk, hipotesis nol yang diuji adalah bahwa data berasal dari distribusi normal. Jika hasil uji menunjukkan nilai p yang lebih kecil dari tingkat signifikansi yang ditetapkan (misalnya, 0,05), maka hipotesis nol ditolak, yang berarti data tidak mengikuti distribusi normal. Sebaliknya, jika nilai p lebih besar dari tingkat signifikansi, tidak ada cukup bukti untuk menolak hipotesis nol, dan dapat diasumsikan bahwa data mengikuti distribusi normal. Uji ini sangat berguna dalam berbagai aplikasi statistik, seperti analisis regresi dan uji parametrik lainnya yang mengharuskan data mengikuti distribusi normal [18].

2.13 Uji T Berpasangan

Uji-t adalah metode statistik yang digunakan untuk menilai apakah dua sampel berasal dari populasi yang sama dengan rata-rata yang serupa. Uji-t dapat diterapkan pada satu atau dua populasi, di mana uji-t untuk satu sampel membandingkan dua rata-rata (mean) untuk menentukan apakah perbedaan yang diamati signifikan secara statistik atau hanya terjadi secara kebetulan. Uji ini sangat berguna dalam berbagai bidang penelitian, seperti psikologi, ekonomi, dan ilmu sosial, di mana membandingkan kelompok data kecil sering kali diperlukan.

Uji-t berpasangan adalah jenis uji parametrik yang digunakan untuk dua set data yang berhubungan, seperti sebelum dan sesudah intervensi pada kelompok yang sama. Uji-t sampel independen, di sisi lain, digunakan untuk menilai apakah terdapat perbedaan rata-rata antara dua sampel independen atau tidak terkait. Dalam kasus sampel berpasangan, jumlah data biasanya sama atau berasal dari sumber yang sama, yang memungkinkan perbandingan yang lebih valid.

Langkah penting berikutnya dalam analisis ini adalah menentukan apakah data yang digunakan bersifat parametrik atau non-parametrik. Data parametrik memenuhi asumsi tertentu, seperti distribusi normal, yang memungkinkan penerapan uji-t. Jika data tidak memenuhi asumsi ini, maka analisis non-parametrik, seperti uji Mann-Whitney atau Wilcoxon, mungkin lebih tepat untuk digunakan [19].