# BAB III
# LANDASAN TEORI

## 3.1. Konsep Dasar Sistem Informasi

Sistem adalah sekumpulan unsur/elemen yang saling berkaitan dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan (Azis, 2022). Sistem adalah gabungan dari kumpulan elemen, komponen atau variabel yang saling berhubungan satu sama lainnya guna untuk mencapai suatu tujuan tertentu (Surya & Kurniawan, 2024). Sistem merupakan suatu kesatuan yang terdiri dari berbagai elemen, bagian, atau sub-sistem yang saling terintegrasi dan berinteraksi satu sama lain demi mencapai target atau tujuan tertentu (Soufitri, 2023).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, sistem adalah gabungan atau sekumpulan unsur, elemen, komponen, atau sub-sistem yang saling berkaitan, berhubungan, dan saling mempengaruhi satu sama lain secara sinergis untuk melakukan kegiatan bersama guna mencapai suatu tujuan tertentu yang telah ditetapkan.

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya (Wijoyo, 2021). Informasi merupakan sesuatu yang mengandung makna yang sangat penting dalam kegiatan proses pengambilan keputusan. Karena informasi harus benar-benar bebas dari kesalahan-kesalahan yang menyesatkan dan informasi itu sendiri mengandung nilai penuh yakni keakuratan, tepat waktu, dan relevan (Surya & Kurniawan, 2024). Informasi merupakan data yang telah diolah, dibentuk, ataupun dimanipulasi sesuai dengan keperluan tertentu bagi penggunanya (Soufitri, 2023). Informasi juga adalah data yang diolah menjadi bentuk lebih berguna dan lebih berarti bagi yang menerimanya (Aisah et al., 2021).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, informasi adalah data yang telah diolah, dibentuk, atau dimanipulasi sehingga menjadi bentuk yang lebih berarti, memiliki makna, dan berguna bagi penerimanya. Informasi yang

berkualitas harus bebas dari kesalahan yang menyesatkan serta memenuhi karakteristik akurasi, ketepatan waktu, dan relevansi, sehingga dapat berfungsi sebagai dasar yang sangat penting dalam proses pengambilan keputusan sesuai keperluan penggunanya.

Sistem Informasi adalah data yang telah diklasifikasikan atau diolah atau interpretasikan untuk digunakan dalam proses pengambilan keputusan (Aisah et al., 2021). Sistem informasi merupakan kombinasi yang ada pada suatu organisasi yang mempertemukan kebutuhan pengelolaan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang dibutuhkan (Anna, 2021). Sistem Informasi adalah sekumpulan *hardware, software, brainware,* prosedur dan atau aturan yang diorganisasikan secara integral untuk mengolah data menjadi informasi yang bermanfaat guna memecahkan masalah dan pengambilan keputusan (Noviana et al., 2021). Sistem informasi merupakan sebuah kumpulan dari beberapa komponen yang mengelola data supaya data yang diolah dapat dijadikan sebagai informasi yang bermakna dan dapat membantu mencapai tujuan organisasi (Lim & Ridho, 2021).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, sistem informasi adalah kombinasi terintegrasi dari komponen *hardware, software, brainware,* serta prosedur yang berfungsi untuk mengolah, mengklasifikasikan, dan menginterpretasikan data menjadi informasi yang bermakna. Sistem ini bertujuan untuk mendukung kebutuhan operasional, transaksi harian, dan kegiatan strategis manajerial dalam suatu organisasi, serta berfungsi sebagai alat bantu yang krusial dalam proses pemecahan masalah dan pengambilan keputusan guna mencapai tujuan organisasi yang telah ditetapkan.

Konsep sistem informasi ini menjadi fondasi utama dalam mengintegrasikan elemen *hardware, software,* dan *brainware* pada proyek SINEO. Implementasinya berfokus pada pengolahan data mentah orbital menjadi informasi yang berkualitas akurat, tepat waktu, dan relevan sehingga dapat mendukung literasi sains dan proses pengambilan keputusan bagi masyarakat terkait potensi bahaya benda langit.

### 3.2. *Monitoring*

*Monitoring* merupakan proses sistematis yang melibatkan pengamatan, pengukuran, dan evaluasi secara berkelanjutan terhadap suatu aktivitas, sistem, atau kondisi tertentu untuk memastikan kesesuaiannya dengan standar atau tujuan yang telah ditetapkan. *Monitoring* berperan penting dalam berbagai bidang, seperti teknologi, lingkungan, kesehatan, dan manajemen, guna mendeteksi penyimpangan serta memungkinkan tindakan korektif yang diperlukan agar sistem atau program tetap berjalan secara optimal (Susilania, 2025). *Monitoring* atau pengawasan adalah suatu kegiatan yang dapat digambarkan sebagai kesadaran akan apa yang ingin diketahui, pengawasan tingkat tinggi yang dilakukan sehingga pengukuran dari waktu ke waktu dapat dilakukan yang menunjukkan pergerakan menuju atau menjauh dari target (Trisnawati et al., 2022). *Monitoring* adalah proses rutin pengumpulan data atau pemantauan untuk menemukan dan memperbaiki kesalahan dalam suatu program atau objek. Ini berfokus pada proses dan hasilnya. Pemantauan akan memberikan laporan tentang status dan kemajuan perubahan yang dilakukan secara teratur (Hutagaol et al., 2022).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *monitoring* adalah sebuah proses pengawasan sistematis dan rutin yang dilakukan secara berkelanjutan untuk mengamati, mengukur, serta mengevaluasi perkembangan suatu objek atau program. Kegiatan ini berfungsi sebagai instrumen untuk menjaga kesadaran terhadap arah pergerakan target, sehingga setiap penyimpangan dapat dideteksi sedini mungkin. Dengan penyajian laporan status yang teratur, *monitoring* memungkinkan adanya tindakan korektif dan perbaikan kesalahan guna memastikan sistem tetap berjalan optimal dan sesuai dengan standar atau tujuan yang telah ditetapkan.

Teori *monitoring* ini diterapkan pada fungsi utama sistem untuk mengamati pergerakan objek NEO secara berkelanjutan. Dengan menyajikan laporan status yang teratur pada antarmuka web, sistem SINEO memungkinkan adanya

pengawasan tingkat tinggi terhadap ancaman asteroid, sehingga setiap pergerakan yang menjauh atau mendekat dari target (bumi) dapat dideteksi sedini mungkin.

### 3.3. Prediksi

Prediksi adalah memperkirakan keadaan di masa yang akan datang melalui pengujian keadaan di masa lalu (Sumari et al., 2021). Prediksi adalah salah satu proses memperkirakan secara sistematis tentang sesuatu yang paling mungkin terjadi di masa depan berdasarkan informasi masa lalu dan sekarang yang dimiliki, agar kesalahannya (selisih antara sesuatu yang terjadi dengan hasil perkiraan) dapat diperkecil (Adiguno et al., 2022). Prediksi merupakan suatu tindakan untuk memperkirakan keadaan pada masa mendatang berdasarkan data masa lampau (Sipayung et al., 2024).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, prediksi adalah suatu tindakan atau proses memperkirakan keadaan yang akan terjadi di masa depan secara sistematis dengan menggunakan landasan data historis (masa lalu) serta informasi masa kini. Tujuan utama dari proses ini adalah untuk menghasilkan perkiraan yang paling mendekati kenyataan, sehingga selisih atau tingkat kesalahan antara hasil prediksi dengan kejadian aktual dapat diminimalisir.

Landasan teori prediksi digunakan sebagai dasar pengembangan fitur estimasi jarak terdekat (*close approach*) pada sistem. Penulis memanfaatkan data historis sekuensial untuk memprediksi kondisi masa depan dengan tujuan memperkecil selisih kesalahan (*error*) antara hasil perkiraan algoritma dengan kejadian aktual, sehingga informasi yang dihasilkan lebih mendekati kenyataan.

### 3.4. *Near-Earth Objects* (NEO)

Populasi objek dekat Bumi (NEO) terdiri dari asteroid dan komet yang berada pada orbit yang membawa mereka dekat dengan Bumi. NEO didefinisikan sebagai objek dengan orbit yang membawa mereka lebih dekat dari 1,3 AU (jarak *perihelion* $q \leq$ 1,3 AU) dari Matahari (Grav et al., 2023). Sejak waktu pembentukan (lebih dari 4,5 miliar tahun lalu), planet kita telah banyak kali terkena benda-benda alami yang

memiliki orbit ke dalam tata surya bagian dalam. Benda-benda semacam itu umumnya disebut Objek Dekat Bumi (NEO) dimana sebagian besar dari mereka adalah Asteroid Dekat Bumi atau *Near-Earth Asteroid* (NEA), kemungkinan objek tersebut merupakan fragmen dari Asteroid Sabuk Utama atau *Main-Belt Asteroid* (MBA) yang setelah peristiwa tumbukan, mengalami evolusi (resonansi dan gangguan non-gravitasi yang memainkan peran penting) hingga mencapai orbit mendekati Bumi (Tommei, 2021).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *Near-Earth Objects* (NEO) adalah populasi benda langit berupa asteroid dan komet yang memiliki lintasan orbit mendekati Bumi dengan jarak perihelion (titik terdekat ke Matahari) kurang dari atau sama dengan 1,3 au.

Definisi dan klasifikasi NEO sangat penting dalam penelitian ini untuk membatasi ruang lingkup objek yang dipantau, yaitu asteroid. Pengetahuan mengenai evolusi orbit objek ini membantu penulis dalam menentukan variabel elemen orbital mana saja yang harus ditarik dari API NASA untuk diproses lebih lanjut oleh sistem

#### 3.4.1. *Center for* neo *studies* (cneos)

CNEOS merupakan pusat dari sistem pemantauan dampak Sentry milik *Jet Propulsion Laboratory* (JPL), yang melakukan analisis jangka panjang terhadap kemungkinan orbit asteroid berbahaya di masa depan, serta mencari potensi tabrakan selama satu abad ke depan. Secara serupa, sistem *Scout* dari CNEOS memantau halaman web MPC untuk penemuan asteroid baru yang potensial dan menghitung berbagai kemungkinan pergerakan di masa depan, bahkan sebelum objek-objek tersebut dikonfirmasi sebagai penemuan resmi.

### 3.5. *Recurrent Neural Network* (RNN)

*Recurrent Neural Network* (RNN) adalah algoritma dalam *deep learning* yang dapat mengolah data berurutan atau *time series* (Farisi et al., 2024). *Recurrent Neural Network* (RNN) adalah model *deep learning* yang menggunakan konsep *supervised learning* (pembelajaran terawasi). *Deep learning* memiliki kemampuan

untuk menangani jaringan saraf yang lebih kompleks dan utamanya berfokus pada data berurutan (*sequential data*). Jaringan rekuren ini mampu memproses data satu per satu dengan tetap mempertahankan elemen informasi tertentu yang tercermin dalam jangka waktu yang lama (Kaur & Mohta, 2019). *Recurrent Neural Networks* (RNN) adalah arsitektur jaringan saraf yang memiliki *hidden state* (keadaan tersembunyi) dan menggunakan *feedback loops* (ikatan umpan balik) untuk memproses urutan data yang pada akhirnya memengaruhi *output* akhir. Oleh karena itu, model RNN mampu mengenali karakteristik sekuensial (berurutan) dalam data dan membantu memprediksi titik data berikutnya yang paling mungkin terjadi dalam urutan data tersebut (Das et al., 2023).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *Recurrent Neural Networks* (RNN) adalah arsitektur *deep learning* berbasis *supervised learning* yang dirancang khusus untuk mengolah data berurutan (*sequential*) atau deret waktu (*time-series*).

Dalam laporan ini, RNN diimplementasikan sebagai 'otak' komputasi untuk mengenali karakteristik sekuensial dari elemen orbital NEO. Konsep *hidden state* dan *feedback loops* pada RNN memungkinkan sistem untuk mempertahankan informasi jangka lama dari data historis, sehingga mampu memprediksi titik data berikutnya dalam lintasan objek secara presisi.

### 3.5.1. Arsitektur dan Mekanisme Kerja Model RNN

Menurut (Masih, 2024)arsitektur dasar RNN terdiri dari sekumpulan *state* tersembunyi yang diperbarui pada setiap langkah waktu berdasarkan input saat ini dan state tersembunyi sebelumnya. Hal ini dapat dijelaskan secara matematis sebagai berikut:

Misalkan $xt$ adalah input pada langkah waktu $t$, $ht$ adalah *state* tersembunyi dan $ht-1$ adalah state tersembunyi sebelumnya pada langkah waktu $t$, dan $yt$ adalah *output* pada langkah waktu $t$. Persamaan yang mengatur RNN adalah:

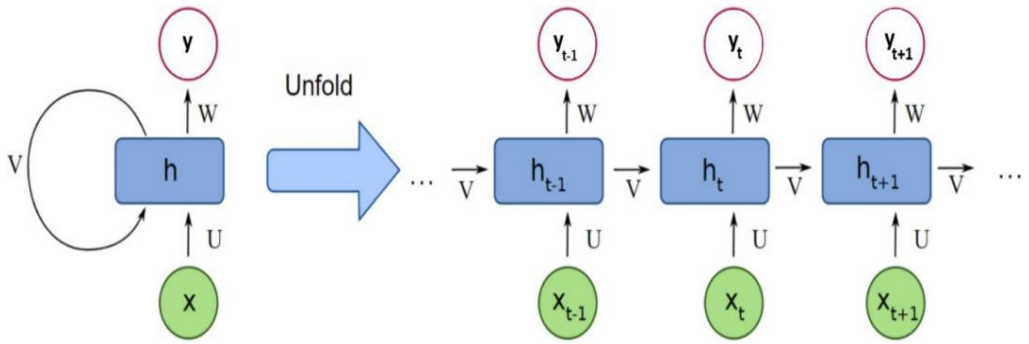
$$h_t = f(W_h x_t + U_h h_{t-1} + b_h) \quad (1)$$

$$y_t = f(W_y h_t + b_y) \quad (2)$$

Di sini $Wh$ dan $Wy$ adalah matriks bobot untuk input dan output masing-masing. $Uh$ adalah matriks bobot untuk keadaan tersembunyi, $bh$ adalah vektor bias untuk keadaan tersembunyi, $by$ adalah vektor bias untuk keluaran, $f$ adalah fungsi aktivasi, yang sering digunakan karena sifat nonliniernya.

Gambar 3. 1 di bawah ini mengilustrasikan arsitektur Jaringan Saraf Berulang (RNN). Di sebelah kiri, representasi ringkas menunjukkan satu unit berulang dengan input $x$, keadaan tersembunyi $h$, dan keluaran $y$. Koneksi menunjukkan bagaimana keadaan tersembunyi $h$ dipengaruhi oleh input $x$ saat ini, keadaan tersembunyi sebelumnya (*loop*), dan berkontribusi pada keluaran $y$.

Di sebelah kanan, representasi yang diperluas menggambarkan bagaimana RNN memproses urutan input selama langkah waktu $t-1$, $t$, dan $t+1$. Setiap langkah waktu $t$ memiliki inputnya sendiri $xt$, keadaan tersembunyi $ht$, dan *output* $yt$. Keadaan tersembunyi $ht$ diperbarui oleh keadaan tersembunyi sebelumnya $ht-1$ dan *input* saat ini $xt$. Matriks bobot $U$, $V$, dan $W$ dibagi di semua langkah waktu, menggambarkan kemampuan RNN untuk menangani data sekuensial dengan mempertahankan dan memperbarui memori input sebelumnya.




Gambar 3. 1 Arsitektur Dasar *Recurrent Neural Network* (RNN) Terlipat (*Folded*) dan Terurai (*Unfolded*) (Sumber:(Masih, 2024) )

Model *deep learning* yang diterapkan pada sistem SINEO dikembangkan secara khusus untuk mengenali karakteristik sekuensial dari elemen orbital asteroid. Mengingat sifat data koordinat orbital dan jarak meleset (*miss distance*) NASA

memiliki pola fluktuasi deret waktu yang dinamis, arsitektur model ini dirancang melalui beberapa tahapan komputasi sebagai berikut:

a. *Temporal Input Layer* (*Sliding Window*)

Model menerima *input* berupa matriks sekuensial dengan ukuran jendela (*window size*) sepanjang 30 data kejadian terakhir. Fitur *input* terdiri dari 4 elemen orbital dasar hasil ekstraksi API NASA CNEOS, yaitu eksentrisitas (*e*), inklinasi (*i*), jarak perihelion (*q*), dan kecepatan relatif (*V*). Langkah awal ini merepresentasikan memori temporal jangka pendek untuk melihat tren pergerakan asteroid sebelum memproyeksikannya ke masa depan.

*b. Dual-Stacked Hidden Layer*

Untuk meningkatkan kapasitas model dalam mempelajari representasi fitur yang hierarkis, sistem SINEO menerapkan dua lapisan RNN yang disusun secara vertikal (bertumpuk). Lapisan pertama memiliki 64 *neuron* dengan fungsi aktivasi *Hyperbolic Tangent* (*Tanh*) yang dikonfigurasi untuk mengeluarkan seluruh sekuens *output* (*return sequences = True*). Lapisan ini bertugas menangkap pola kasar fluktuasi orbital global. Lapisan kedua memiliki 32 *neuron* dengan aktivasi *Tanh* untuk menyaring informasi temporal dari lapisan pertama secara mendalam. Penggunaan fungsi aktivasi *Tanh* (rentang nilai -1 hingga 1) sangat krusial untuk menjaga aliran gradien tetap stabil saat algoritma *Backpropagation Through Time* (BPTT) bekerja memperbarui bobot jaringan, sekaligus meminimalisir risiko terjadinya *vanishing gradient* (kondisi di mana model melupakan pola data masa lalu pada awal jendela sekuensial).

Proses konversi nilai *Final Loss* berbasis *Mean Squared Error* (MSE) murni ke dalam persentase akurasi kedekatan prediksi dilakukan melalui pendekatan matematis yang ketat. Berdasarkan hasil pelatihan model *Dual-Layer Recurrent Neural Network* (RNN), didapatkan nilai MSE pada ruang normalisasi (*normalized space*). Untuk mentransformasikan nilai *error* kuadratik ini menjadi matriks evaluasi yang intuitif, nilai tersebut terlebih dahulu dikonversi ke dalam bentuk *Root Mean Squared Error* (RMSE) melalui Persamaan 3 berikut:

$$Akurasi = \left(1 - \left(\sqrt{MSE} \times Faktor\ Skala\right)\right) \times 100 \quad (3)$$

Nilai *error* kuadratik yang sangat kecil dalam *normalized space* berisiko memicu kesalahan penarikan kesimpulan performa jika dikonversi secara linear, sehingga diperlukan penyelarasan terhadap deviasi ruang data asli. Penggunaan faktor skala dalam sistem SINEO bertindak sebagai parameter Standar Deviasi (*sigma*) untuk merepresentasikan fluktuasi dan volatilitas jarak sebaran lintasan asteroid yang ekstrem. Dengan demikian, reduksi nilai akurasi memberikan gambaran kapabilitas model RNN yang lebih objektif, transparan, dan terbebas dari indikasi manipulasi performa semu (*overfitting bias*) saat disajikan pada antarmuka halaman utama analisis kecerdasan buatan sistem SINEO).

*c. Dense Output Layer & Recursive Forecasting*

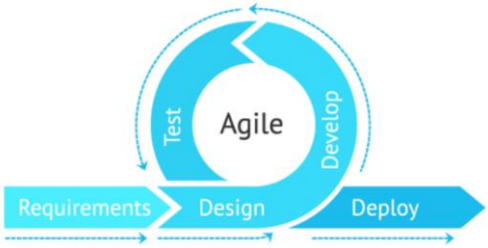
*Output* dari lapisan RNN diteruskan ke sebuah *Dense Layer* (lapisan terkoneksi penuh) dengan 32 *neuron* beraktivasi ReLU sebelum dialirkan menuju lapisan *output* akhir berdimensi 1 yang menghasilkan nilai regresi linear berupa prediksi estimasi jarak terdekat (*miss distance_km*). Untuk memproyeksikan lintasan masa depan, mesin inferensi *Python* pada SINEO menjalankan metode *Recursive Multi-step Forecasting*. Nilai prediksi pada langkah waktu $t + 1$ akan dimasukkan kembali sebagai fitur *input* untuk memprediksi langkah waktu $t + 2$ secara berulang (*rolling window*). Guna menghindari stagnasi matematika (*fixed point*) dan mensimulasikan perturbasi atau gangguan kecil akibat gravitasi planet lain di alam semesta, sistem mengimplementasikan teknik *Jitter Correction* berupa penambahan variasi gangguan acak (*stochastic noise*) pada setiap iterasi proyeksi masa depan.

### 3.6. *Website*

*Website* merupakan sebuah informasi yang berbasis digital, dimana *website* dipergunakan untuk *E-Commerce*, berita, Informasi sekolah dan lain-lain (Kusdiana, 2025). Web atau *website* merupakan kumpulan halaman yang diakses melalui *browser* dengan menggunakan *hyperlink* seperti HTTP untuk mengaksesnya (Handayani & Rachmat, 2022). *Website* adalah suatu kumpulan informasi yang bisa di akses lewat internet oleh semua orang dimanapun dan kapanpun bisa menggunakannya selama terhubung ke jaringan internet (Fauzi et al., 2024).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *website* adalah sekumpulan informasi digital berupa halaman-halaman yang dapat diakses oleh semua orang melalui internet menggunakan *browser* dan protokol *hyperlink* seperti HTTP. *Website* berfungsi sebagai media informasi yang fleksibel karena dapat digunakan kapan saja dan di mana saja selama terhubung ke jaringan internet.

### 3.7. *Agile*




Gambar 3. 2 *Agile*
*(Sumber: Afriyandi et al., 2025)*

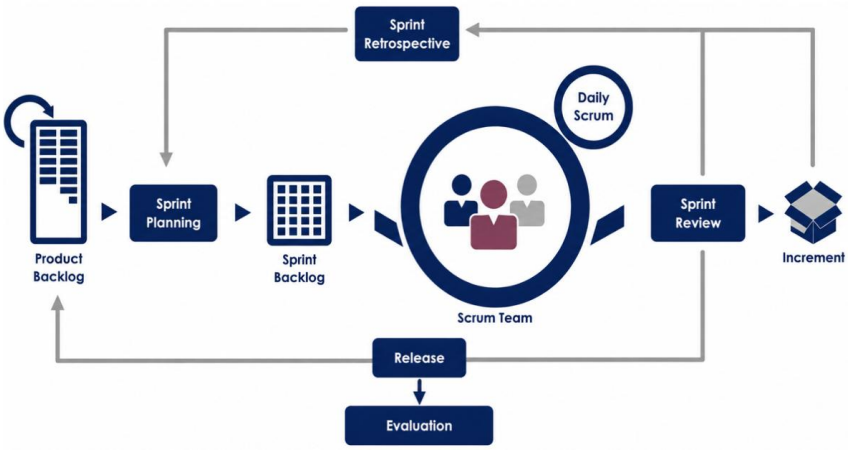
Metode *Agile* adalah pendekatan pengembangan perangkat lunak yang menekankan kolaborasi tim, responsibilitas bersama, fleksibilitas, dan adaptasi terhadap perubahan yang cepat. Pendekatan ini berfokus pada pengiriman nilai secara berkala kepada pengguna dan mengutamakan komunikasi langsung antara pengembang dan pemangku kepentingan (Afriyandi et al., 2025). *Agile* adalah pendekatan pengembangan perangkat lunak yang iteratif dan fleksibel, dapat disesuaikan seiring berjalannya waktu (Rifki et al., 2024). Metode *Agile* adalah metodologi pengembangan perangkat lunak yang mengandalkan proses iteratif dan berulang, dengan aturan dan solusi yang telah disepakati. Metode ini melibatkan kolaborasi tim secara terstruktur dan terorganisir untuk mencapai hasil yang lebih baik (Abdurahman et al., 2025). Metode *Agile* merupakan salah satu pendekatan yang efektif dalam pengembangan perangkat lunak. Istilah *Agile* sendiri menggambarkan sifat yang cepat, ringan, dan mampu beradaptasi dengan baik. Metode ini menekankan pada kebutuhan pengguna yang dinamis, yang seringkali mengalami perubahan permintaan dengan cepat *Agile* dikenal karena fleksibilitasnya dalam menghadapi perubahan yang terjadi (Rahma et al., 2025).

Ada beberapa macam metode *agile*, diantaranya adalah *Extreme Programming*, *Adaptive Software Development*, *Dynamic Systems Development Method* (DSDM) dan *Scrum*. Terdapat beberapa prinsip yang berlaku untuk mengimplementasikan pengembangan perangkat lunak, yakni kepuasan pelanggan adalah prioritas utama, menerima perubahan *requirement*, bahkan di akhir pengembangan, memberikan hasil/perangkat lunak dalam beberapa minggu hingga bulan, selama proses pengembangan, lingkungan tim yang dapat dipercaya dan memotivasi satu anggota yang lain, mengedepankan komunikasi antar tim dan mengedepankan hasil fungsi dari *software* tersebut (Putra & Tanaem, 2022).

### 3.7.1. Metode *scrum*

Rekayasa perangkat lunak yang menerapkan prinsip-prinsip cepat dan mengandalkan sumber daya tim untuk mencapai hasil akhir merupakan pengertian dari *Scrum*. *Scrum* adalah *framework* yang mampu menyelesaikan masalah kompleks yang terus berubah dan juga dianggap mampu secara kreatif dan produktif menghadirkan produk berkualitas tinggi sesuai keinginan pengguna (Putra & Tanaem, 2022). Di antara kerangka kerja *Agile*, *Scrum* dikenal luas karena siklus pengembangan iteratifnya dan penekanan pada kolaborasi antara tim pengembang dan pemangku kepentingan. Dalam jurnal tersebut menekankan, bahwa *Scrum* dapat secara signifikan meningkatkan produktivitas dan kualitas perangkat lunak, khususnya pada proyek yang dicirikan oleh persyaratan yang dinamis dan sering berubah. Sistem ini membutuhkan alur kerja yang berorientasi pada garis waktu dan keterlibatan klien yang berkelanjutan sepanjang fase pengembangan dan eksekusi (H. Maulana et al., 2026).

Adapun berikut merupakan adaptasi kerangka kerja proses *scrum*-nya, seperti sebagai berikut (H. Maulana et al., 2026):




Gambar 3. 3 *Scrum Methodology*
(Sumber: H. Maulana et al., 2026)

*a. Product backlog*

Dalam *Scrum*, *product backlog* mewakili daftar fitur, fungsionalitas, dan tugas pengembangan yang diprioritaskan yang diperlukan untuk mencapai tujuan proyek. Alih-alih berfungsi sebagai artefak desain perangkat lunak, *product backlog* berfungsi sebagai kumpulan item pekerjaan yang memandu pengembangan iteratif sepanjang siklus *sprint*. Pada tahapan ini juga *use case diagram* disertakan untuk mendukung identifikasi persyaratan dan definisi *backlog.* Model *use case* disini, tidak menggantikan *product backlog* melainkan bertindak sebagai artefak pendukung untuk mendefinisikan *item backlog* secara sistematis.

*b. Sprint planning*

Perencanaan *sprint* dilakukan dengan memilih *item backlog* berdasarkan jadwal dan estimasi upaya. Setiap *sprint* mewakili fase kegiatan, memastikan keselarasan antara aktivitas pengembangan dan alur kerja dunia nyata.

*c. Sprint execution (development) / client validation loop*

Selama fase eksekusi sprint, aktivitas pengembangan dilakukan secara iteratif oleh tim pengembangan. Pada tahap ini model sistem terperinci menggunakan *Unified Modeling Language* (UML) disertakan, termasuk diagram aktivitas, diagram urutan, dan diagram kelas. Model-model ini terus diperbarui berdasarkan umpan balik yang diperoleh dari siklus validasi klien. Pendekatan ini memastikan bahwa

desain sistem tetap selaras dengan persyaratan pengguna yang berkembang sambil menjaga konsistensi antara pemodelan sistem dan implementasi.

*d. Sprint review & retrospective*

Di akhir setiap *sprint*, fitur yang telah selesai ditinjau, dan proses pengembangan dievaluasi untuk meningkatkan kinerja di masa mendatang.

*e. Release*

Sistem siap untuk diimplementasikan setelah menyelesaikan beberapa siklus *sprint*.

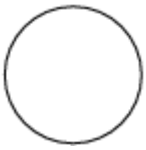
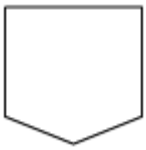
Metode *Scrum* digunakan sebagai kerangka kerja pengembangan perangkat lunak yang iteratif dan transparan dalam membangun SINEO. Penulis membagi pekerjaan ke dalam daftar *product backlog* yang dikerjakan melalui siklus *sprint*, sehingga perkembangan fitur monitoring dan integrasi AI dapat dievaluasi secara produktif di setiap akhir siklus.

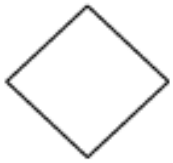
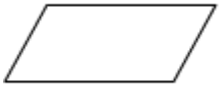
### 3.8. *Flowchart*

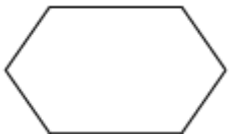
*Flowchart* adalah diagram yang menggambarkan alur proses atau logika suatu sistem menggunakan simbol standar untuk menunjukkan aktivitas, kondisi, dan alur logika. *Flowchart* dapat digunakan dalam berbagai bidang seperti pengembangan perangkat lunak, perencanaan bisnis, manajemen proyek, dan desain sistem (Abdurahman et al., 2025). Menurut Lambot Sitaurus, *flowchart* dapat diartikan sebagai langkah-langkah penyelesaian masalah yang dituliskan dalam suatu simbol-simbol tertentu. Diagram alir ini akan menunjukan alur didalam program secara logika (Khesya, 2021). *Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* sistem merupakan suatu urutan proses dalam sistem dengan menunjukkan alat dari media *input, output* serta jenis media yang digunakan untuk penyimpanan dalam proses pengolahan data sedangkan *flowchart* program merupakan suatu bagan dengan simbol-simbol tertentu yang menggambarkan suatu urutan dari proses secara detail dan berhubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program (Zalukhu et al., 2023).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *flowchart* (diagram alir) adalah representasi grafis yang menggunakan simbol-simbol standar untuk menggambarkan urutan logika, prosedur, dan alur kerja dari suatu sistem atau program. Simbol-simbol tersebut dapat dilihat pada Tabel 3. 1.

Tabel 3. 1 Simbol-simbol *Flowchart*

| **Simbol** | **Keterangan** |
|---|---|
|  | *Flow Symbol* yang digunakan untuk menggabungkan antara simbol yang satu dengan simbol yang lain. |
|  | *On-Page Connector Symbol* yang digunakan untuk keluar masuk atau penyambungan proses dalam halaman yang sama. |
|  | *Off-Page Connector Symbol* yang berfungsi untuk keluar masuk atau penyambungan proses pada lembar/ halaman yang berbeda. |
|  | *Terminator Symbol* yang menyatakan awal atau akhir suatu program. |
|  | *Process Symbol* yang menyatakan suatu proses yang dilakukan dalam komputer. |

| Simbol | Keterangan |
|---|---|
|  | *Decision Symbol* yang digunakan untuk memilih proses yang akan dilakukan berdasarkan kondisi tertentu. |
|  | *Input/output Symbol* yang menyatakan proses input atau output tanpa melihat jenis nya. |
|  | *Manual Operation Symbol* yang digunakan untuk menunjukan pengolahan yang tidak dilakukan oleh komputer. |
|  | *Document Symbol* yang menyatakan bahwa input berasal dari dokumen dalam bentuk fisik atau output yang perlu dicetak. |
|  | *Predefine Process Symbol* yang digunakan untuk mempersiapkan penyimpanan yang sedang/akan digunakan dengan memberikan harga awal |
|  | *Display Symbol* yang menyatakan peralatan output digunakan |

| Simbol | Keterangan |
|---|---|
|  | *Preparation* Simbol yang menyatakan penyediaan tempat penyimpanan pengolahan suatu untuk memberikan nilai awal |

### 3.9. UML

UML merupakan alat bantu pemodelan visual pada pengembangan sistem berorientasi objek. Bahasa ini membantu pengembang membuat rancangan standar yang mudah dipahami serta mendukung komunikasi desain antar tim (Hendriana et al., 2025). Model pengembang *Unified Modeling Language* (UML) adalah bahasa yang digunakan untuk pemodelan sistem tipe grafis yang akan menampilkan model model dalam bentuk notasi-notasi gambar yang dibantu beberapa diagram seperti *Class Diagram, Activity Diagram, Use Case Diagram dan Sequence Diagram* (Furqon & Rohmanto, 2024). UML merupakan bahasa visual untuk pemodelan dan komunikasi, mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung (Aisah et al., 2021). UML (*Unified Modeling Language*) merupakan salah satu bahasa yang banyak digunakan dalam membuat analisa & desain, serta menggambarkan arsitektur dalam pemrogram berorientasi objek (Noviana et al., 2021). UML (*Unified Modeling Language*) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem (Handayani & Rachmat, 2022).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *Unified Modeling Language* (UML) adalah bahasa pemodelan standar berbasis grafis yang digunakan untuk memvisualisasikan, menspesifikasikan, membangun, dan mendokumentasikan arsitektur sistem perangkat lunak berorientasi objek

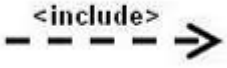
Penggunaan diagram UML (*Unified Modeling Language*) seperti *Use Case*, *Activity*, *Class* dan *Sequence Diagram* dalam skripsi ini bertujuan untuk mendokumentasikan rancangan sistem SINEO secara visual. Dengan alat bantu ini, penulis dapat memetakan interaksi antara pengguna dengan sistem serta alur pertukaran data secara mendalam sebelum masuk ke tahap pengodean (*coding*).

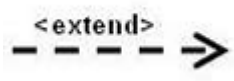
### *3.9.1. Use case diagram*

*Use Case Diagram* merupakan salah satu jenis diagram UML yang digunakan untuk menggambarkan bagaimana pengguna berinteraksi dengan sistem dalam konteks tertentu. *Use case diagram* digunakan untuk menggambarkan secara visual fungsionalitas sistem, sehingga mempermudah pemahaman dan komunikasi antara pengembang perangkat lunak dengan klien atau pengguna (Rahma et al., 2025). *Use Case Diagram* adalah gambaran kelakuan (*behavior*) sistem informasi yang akan dibuat. Diagram ini menampilkan hubungan interaksi antara satu atau lebih aktor dengan sistem, serta mempermudah dalam mengidentifikasi fungsi-fungsi yang terdapat di dalam sistem dan siapa saja yang berhak menggunakannya (Hendriana et al., 2025). *Use case* diagram merupakan salah satu permodelan untuk melakukan sistem informasi yang akan dibuat (Noviana et al., 2021). *Use Case* adalah diagram yang mengidentifikasi jenis pengguna (aktor) yang berinteraksi dengan sistem dan *use case* (fungsionalitas) yang disediakan sistem. Diagram ini digunakan untuk menangkap persyaratan fungsional dan menggambarkan interaksi antara aktor dan sistem (Furqon & Rohmanto, 2024).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *use case diagram* adalah model visual dalam UML yang berfungsi untuk memetakan persyaratan fungsional sistem melalui interaksi antara aktor (pengguna) dan sistem informasi. Simbol-simbol *use case* sendiri dapat dilihat pada Tabel 3. 2:

Tabel 3. 2 Simbol-simbol *Use Case Diagram*

| Gambar | Nama | Keterangan |
|---|---|---|
|  | *Actor* | Menspesifikasi himpunan kepada pemakai untuk memainkan pada saat interaksi dengan *Use Case*. |
|  | *Dependency* | Hubungan di saat perubahan yang terjadi kepada suatu elemen mandiri dan akan menjadi masalah di saat elemen yang bergantung padanya elemen yang tidak mandiri (*independent*). |
|  | *Generalization* | Hubungan yang dimana letak antara objek kecil atau disebut anak yang berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk. |
|  | *Use Case* | *Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, yang dinyatakan dengan menggunakan kata kerja. |
|  | *Include* | *Include* merupakan di dalam *Use Case* lain (*required*) atau pemanggilan *Use Case* oleh *Use Case* lain, contohnya adalah pemanggilan sebuah fungsi program. |

| Gambar | Nama | Keterangan |
|---|---|---|
|   | *Extend* | *Extend* merupakan perluasan dari *Use Case* lain jika kondisi atau syarat terpenuhi. |
|  | *Association* | *Asosiasi* antara aktor dan *Use Case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data. |

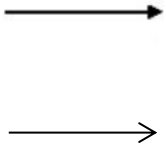
*3.9.2. Activity diagram*

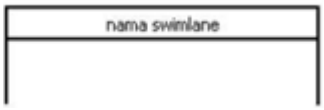
*Activity Diagram* merupakan jenis diagram yang memodelkan alur kerja atau proses dengan menggambarkan berbagai aktivitas dalam sistem secara terstruktur. Diagram ini memvisualisasikan proses dari satu aktivitas ke aktivitas lainnya serta pengaturan alur kerja secara keseluruhan (Furqon & Rohmanto, 2024). *Activity Diagram* merupakan gambaran aliran kerja (*work flow*) atau *activity* dari sebuah sistem atau menu yang ada pada sistem yang akan dirancang (Noviana et al., 2021). *Activity diagram* adalah diagram yang digunakan untuk menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam siatu operasi sehingga dapat juga digunakan untuk aktivitas lainnya seperti *use case* atau interaksi (Handayani & Rachmat, 2022). *Activity diagram* merupakan diagram yang digunakan untuk memodelkan urutan aktivitas yang ada pada sebuah sistem yang berfungsi untuk memberikan gambaran bagaimana masing-masing aktivitas berawal, pemilihan yang mungkin terjadi dan bagaimana suatu aktivitas berakhir (Setiawan et al., 2022).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *Activity Diagram* adalah instrumen pemodelan dalam UML yang berfungsi untuk

memvisualisasikan alur kerja (*workflow*) atau rangkaian aktivitas secara terstruktur di dalam sebuah sistem. Simbol-simbol tersebut dapat dilihat pada Tabel 3. 3:

Tabel 3. 3 Simbol-simbol *Activity Diagram*

| Gambar | Nama | Keterangan |
|---|---|---|
|  | *Start* | Menandakan titik awal atau dimulainya suatu alur aktivitas. Setiap diagram aktivitas harus memiliki satu titik awal. |
|  | *Activity* | Mewakili suatu langkah, tindakan, atau tugas yang harus diselesaikan dalam proses. Nama dari aktivitas biasanya ditulis di dalam simbol ini. |
|  | *Decision* | Digunakan untuk menunjukkan sebuah titik di mana ada pilihan atau keputusan yang harus dibuat. Biasanya diikuti oleh dua atau lebih aliran keluar, masing-masing dengan kondisi yang berbeda (disebut *guard*). |
|  | *End Point* | Menandakan berakhirnya alur aktivitas. Suatu diagram bisa memiliki lebih dari satu titik akhir. |
|  | *Control Flow* | Menghubungkan satu aktivitas ke aktivitas lain, menunjukkan urutan atau alur langkah-langkah dalam proses. |

| Gambar | Nama | Keterangan |
|---|---|---|
|  | *Swimlane* | Memisahkan organisai bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. |

### *3.9.3. Class diagram*

*Class Diagram* adalah representasi grafis dari struktur statis sistem yang mendefinisikan kelas-kelas serta hubungan mereka. Diagram ini menggambarkan detail tentang atribut-atribut dan metode-metode yang dimiliki oleh setiap kelas dan bagaimana kelas-kelas tersebut berinteraksi satu sama lain (Furqon & Rohmanto, 2024). Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (Aisah et al., 2021). *Class Diagram* menggambarkan struktur sistem dari segi pendefinisan kelas-kelas yang akan dibuat untuk membangun sistem sehingga kelas memiliki atribut dan metode atau operasi (Noviana et al., 2021). *Class diagram* merupakan diagram yang berfungsi untuk menggambarkan hubungan antar objek pada class dan merupakan *blueprint* atau cetak biru dari sistem yang akan dibangun. *Class* akan berasosiasi tepat dengan satu objek dan menunjukkan atribut serta *method* apa saja yang harus ada dalam sistem (Setiawan et al., 2022).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *Class Diagram* adalah representasi grafis dalam UML yang berfungsi sebagai *blueprint* atau cetak biru untuk menggambarkan struktur statis dan arsitektur suatu sistem perangkat lunak.

### *3.9.4. Sequence diagram*

*Sequence Diagram* adalah alat penting untuk memodelkan interaksi antar objek dalam sistem dengan fokus pada urutan waktu. Diagram ini membantu dalam merancang dan memahami bagaimana pesan dikirim di sepanjang alur interaksi sistem (Furqon & Rohmanto, 2024). *Sequence Diagram* menggambarkan kelakuan objek pada *use case* dengan menjelaskan waktu objek dan message yang dikirimkan

dan diterima antar objek (Noviana et al., 2021). *Sequence Diagram* merupakan diagram yang berfungsi untuk menjelaskan dan menampilkan interaksi antar objek yang ada dalam sistem secara sekuensial. Interaksi antar objek ditampilkan dalam dua dimensi yaitu dimensi vertikal yang merepresentasikan poros waktu, dan dimensi horizontal yang merepresentasikan objek-objek individual (Setiawan et al., 2022).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *Sequence Diagram* adalah diagram UML yang berfungsi untuk memodelkan interaksi antarobjek di dalam sistem secara kronologis dengan menekankan pada urutan waktu pengiriman pesan.

### 3.10. Draw.io

Draw.io adalah sebuah *website* yang didesain khusus untuk menggambarkan diagram UML secara *online*. Dimana punya tampilan yang sangat *responsive* dan terintergrasi dengan layanan penyimpanan file milik google yaitu Google Drive sehingga draw.io menjadi alternatif dalam pembuatan diagram UML dengan waktu yang lebih singkat (Marthiawati et al., 2024). Draw.io adalah *website* dan *software* yang digunakan untuk membuat *flowchart*, draw.io berguna untuk merancang *Use Case Diagram* maupun *activity diagram (A. Maulana, 2025).*

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, draw.io adalah *platform* daring dan perangkat lunak yang dirancang untuk membuat berbagai jenis diagram seperti UML, *flowchart*, *use case*, *activity diagram* dan lainnya dengan cepat, serta memiliki tampilan responsif dan terintegrasi dengan *Google Drive*.

### 3.11. *Database*

Basis data (*database*) adalah kumpulan data yang diatur dan memiliki hubungan secara logika menurut susunan tertentu dan disimpan dalam suatu media penyimpanan komputer. Basis data adalah sekumpulan informasi-informasi yang saling terkait satu dengan yang lainnya. Penggunaan basis data sudah sangat luas,

sudah banyak aplikasi yang memanfaatkan basis data untuk mengolah data yang ada, penggunaan basis data memudahkan setiap orang untuk melakukan aktivitasnya. Sistem basis data menyediakan dua jenis bahasa yaitu data *definition language* yang berfungsi untuk membuat skema basis data dan data *manipulation language* yang berfungsi untuk melakukan pencarian dan perbaruan data pada basis data. Konsep dasar basis data diantaranya (Handayani & Rachmat, 2022):

a. *Field*

merupakan implementasi dari suatu atribut data, *field* merupakan unit terkecil dari data yang berarti yang disimpan dalam suatu basis data.

b. *Record*

*field-field* yang di organisasikan yang telah disusun dalam format yang ditentukan.

c. *File* dan *table*

*record-record* yang diorganisasikan dalam grup yang disebut *file*, dan *table* merupakan relasional dari sebuah *file.*

Berdasarkan sumber tersebut didapatkan kesimpulan bahwa, basis data (*database*) adalah sekumpulan informasi yang saling terkait dan terorganisir secara logis dalam media penyimpanan komputer untuk memudahkan pengelolaan data dalam berbagai aplikasi.

### 3.12. Laragon

Laragon adalah perangkat lunak yang memiliki bahasa pemrograman PHP (*Hypertext Preprocessor*) dan MySQL sebagai tempat penyimpanan *database* dan *apache* sebagai *web server* yang akan digunakan untuk membangun *local development environment* pada sistem operasi *windows* (Hendra et al., 2026). Laragon adalah perangkat lunak yang bersifat *open source* (terbuka) yang dapat mendukung banyak sekali sistem operasi di mana Laragon bertugas sebagai server virtual atau sering disebut sebagai *localhost* (Talitha, 2024). Laragon juga adalah sebuah aplikasi mirip seperti XAMPP, namun didesain untuk kebutuhan *developer* PHP yang menggunakan *framework* Laravel. *Service* yang *include* dalam Laragon seperti Apache, MySQL, PHP *Server*, Memchaced, Redis, Composer, Xdebug,

PhpMyAdmin, Cmder, dan lainnya. Aplikasi ini sangat cocok digunakan oleh seorang *developer* PHP yang menggunakan *framework* Laravel karena akan mempermudah dalam melakukan pengembangan aplikasi (Suci & Rismayati, 2024).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, Laragon adalah perangkat lunak *open-source* yang berfungsi sebagai penyedia lingkungan pengembangan lokal (*local development environment*) atau server virtual pada sistem operasi Windows.

### 3.12.1. Mysql

MySQL adalah salah satu Sistem Manajemen Basis Data Relasional (RDBMS) yang paling populer dan banyak digunakan, terutama dalam pengembangan aplikasi berbasis web. Popularitas MySQL tidak lepas dari sejumlah keunggulan yang dimilikinya, seperti bersifat *open source* (gratis untuk digunakan), kemudahan dalam pengelolaan data, serta tingkat keamanan yang cukup baik untuk berbagai skala aplikasi. Manipulasi data dalam MySQL mencakup berbagai operasi dasar seperti menambahkan data baru (*INSERT*), memperbarui data yang sudah ada (*UPDATE*), dan menghapus data yang tidak lagi dibutuhkan (*DELETE*)(Talitha, 2024). MySQL (*My Structure Query Language*) merupakan basis data yang paling digemari dikalangan *programmer* web, dengan alasan bahwa program ini merupakan basis data yang sangat kuat dan cukup stabil untuk digunakan sebagai media penyimpanan data. Sebagai sebuah basis data server yang mampu untuk memenajemen basis data dengan baik, mysql terhitung merupakan basis data yang paling digemari dan paling banyak digunakan dibandingkan dengan basis data lainnya. Selain mysql masih terdapat beberapa jenis basis data server yang juga memiliki kemampuan yang juga tidak bisa dianggap enteng, basis data itu adalah Oracle dan PostgreSQL (Noviantoro et al., 2022).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, MySQL adalah Sistem Manajemen Basis Data Relasional (*RDBMS*) yang sangat populer dan paling banyak digemari di kalangan *programmer* untuk pengembangan aplikasi

berbasis web. Popularitasnya didukung oleh sifatnya yang *open source,* kemudahan dalam pengelolaan data, tingkat keamanan yang baik, serta kemampuannya dalam melakukan berbagai operasi manipulasi data dasar seperti *insert, update,* dan *delete.*

Sistem manajemen basis data MySQL diterapkan dalam skripsi ini untuk menyimpan data historis *Near-Earth Objects* yang telah ditarik dari API NASA. Struktur tabel pada MySQL dirancang secara relasional untuk memastikan integritas data elemen orbital tetap terjaga, sehingga model RNN dapat mengakses data latihan (*training data*) dengan cepat dan terstruktur.

### 3.13. Bahasa Pemrograman

Bahasa pemrograman adalah cara bagi pemrogram (pengembang) untuk berkomunikasi dengan komputer. Bahasa pemrograman terdiri dari seperangkat aturan yang memungkinkan nilai *string* diubah menjadi berbagai hasil kode mesin, atau dalam kasus pemrograman visual serta elemen grafis (Wali et al., 2023). Bahasa pemrograman dengan kerangka kerja web (*framework*) adalah kombinasi bahasa pemrograman dan kerangka kerja yang digunakan untuk aplikasi web dengan lebih efisien (Kurniawan et al., 2023). Bahasa pemrograman adalah bahasa yang digunakan para pengguna perangkat lunak untuk berbicara dengan komputer. Bahasa mesin dan bahasa *assembly* merupakan contoh bahasa pemrograman tingkat rendah, sedangkan BASIC FOR TRAN, COBOL, PASCAL, C++, dan DELPHI merupakan contoh bahasa pemrograman tingkat tinggi. Selain model-model ini, ada banyak bahasa pemrograman lainnya (Aulia, 2024).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, bahasa pemrograman adalah sekumpulan aturan yang digunakan oleh pengembang untuk berkomunikasi dan memberikan instruksi kepada komputer, baik untuk menghasilkan kode mesin maupun elemen grafis.

### 3.13.1. Php

PHP adalah bahasa pemrograman yang digunakan untuk kebutuhan membangun web yang biasa dijalankan untuk mengolah informasi di internet (Hendra et al., 2026). PHP atau kependekan dari *Hypertext Preprocessor* adalah salah satu bahasa pemrograman *open source* yang sangat cocok atau dikhususkan untuk pengembangan web dan dapat ditanamkan pada sebuah skripsi HTML. Bahasa PHP dapat dikatakan menggambarkan beberapa bahasa pemrograman seperti C, Java, dan Perl serta mudah untuk dipelajari. PHP merupakan bahasa *scripting server-side*, dimana pemrosesan datanya dilakukan pada sisi server (Noviantoro et al., 2022). PHP merupakan bahasa pemrograman untuk membuat web yang bersifat *server-side scripting*. PHP memungkinkan untuk membuat halaman web yang bersifat dinamis. Sistem manajemen basis data yang sering digunakan bersama PHP adalah MySQL namun PHP juga mendukung sistem manajemen *database* Oracle, Microsoft Access, Interbase, d-base, PostgreSQL, dan sebagainya (Adrianto, 2021).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, PHP (*Hypertext Preprocessor*) adalah bahasa pemrograman *open source* bertipe *server-side scripting* yang dikhususkan untuk membangun halaman *website* dinamis.

Dalam pengembangan SINEO, bahasa pemrograman PHP digunakan sebagai mesin pengolah logika di sisi server (*server-side*). Penulis memanfaatkan PHP untuk menjembatani komunikasi antara antarmuka web dengan basis data, sehingga setiap proses kalkulasi data elemen orbital dan autentikasi pengguna dapat berjalan secara dinamis dan responsif.

### 3.13.2. Python

Python adalah bahasa pemrograman yang diciptakan oleh Guido van Rossum dan pertama kali dirilis pada tahun 1991. Bahasa ini dikenal dengan sintaksnya yang sederhana dan mudah dipahami, serta mendukung berbagai paradigma pemrograman seperti prosedural, berorientasi objek, dan fungsional. Python bersifat *multiplatform* dan dapat dijalankan di berbagai sistem operasi seperti Windows, Linux, dan macOS (Nugraha, 2025). Python merupakan bahasa

pemrograman dinamis dan serba guna, serta memiliki sistem pengolahan sampah otomatis. Python juga sering dijelaskan dengan istilah "*batteries included*", yang berarti bahwa bahasa pemrograman ini dilengkapi dengan beragam modul (*module*) dalam pustaka (*library*) standar yang dapat digunakan untuk berbagai tugas (Prasetya, 2024). Python adalah bahasa terpopuler kedua untuk analitik dan ilmu data, python mendukung banyak sekali beban kerja pemrosesan data di berbagai organisasi di seluruh dunia. Sementara itu, pustaka Python seperti OpenCV untuk visi komputer dan *TensorFlow* untuk jaringan saraf digunakan dalam ribuan proyek pembelajaran mesin setiap hari (Saabith et al., 2019).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, python adalah bahasa pemrograman tingkat tinggi yang dinamis, serbaguna, dan dirancang dengan sintaks yang sederhana sehingga mudah dipahami oleh pengembang.

Python digunakan secara khusus dalam penelitian ini untuk membangun arsitektur model *Recurrent Neural Network* (RNN). Penulis memilih Python karena ketersediaan pustaka (*library*) yang mumpuni untuk pengolahan data saintifik, yang kemudian hasilnya diintegrasikan ke dalam sistem utama berbasis Laravel untuk ditampilkan pada *dashboard* monitoring.

### 3.14. *Framework*

*Framework* merupakan sekumpulan fungsi-fungsi atau sebuah prosedur dan *class-class* tertentu yang dikembangkan dengan tujuan tertentu yang dapat dipakai oleh *programmer* untuk bisa lebih mudah dan cepat dalam menyelesaikan pekerjaan seorang *programmer* akan lebih ringkas, tanpa membuat sebuah fungsi maupun *class* tertentu dari awal (Suci & Rismayati, 2024). *Framework* adalah sebuah struktur konseptual dasar yang digunakan untuk memecahkan sebuah permasalahan atau isu-isu kompleks. Adapun keuntungan menggunakan *framework* adalah menghemat waktu pengembangan, *reuse of code*, bantuan komunitas (Gibran et al., 2024).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *framework* adalah sebuah struktur konseptual dasar yang berisi sekumpulan fungsi, prosedur,

dan kelas-kelas siap pakai untuk membantu pemrogram memecahkan masalah kompleks secara lebih efisien. Penggunaan *framework* memungkinkan proses pengembangan aplikasi menjadi lebih ringkas dan cepat karena pemrogram tidak perlu membangun kode dari nol, melainkan dapat memanfaatkan prinsip *reuse of code*, menghemat waktu pengembangan, serta mendapatkan dukungan dari bantuan komunitas

### 3.14.1. Laravel

Laravel merupakan salah satu *framework* web yang berbasis PHP dan dikembangkan secara *open source*, Laravel dikembangkan oleh Taylor Otwell dan digunakan untuk mengembanganken aplikasi berbasis web yang menerapkan sebuah pola yaitu MVC. Struktrur MVC yang diterapkan Laravel ini agak berbeda dari MVC yang pada umumnya. Pada Laravel memiliki fitur *routing* yang digunakan untuk menghubungkan antara *request user* dan sebuah *controller* yang menerimanya. Sehingga *controller* tidak bisa langsung dapat menerima sebuah *request* tertentu (Suci & Rismayati, 2024). Laravel adalah struktur PHP yang terkenal dan secara luas diaplikasikan untuk mendorong aplikasi elektronik mulai dari *project* yang kecil maupun besar diseluruh dunia. *Framework* ini banyak di implementasikan karena memiliki keunggulan baik dari sisi fitur, kinerja dan skalabilitasnya. *Framework Laravel* memiliki struktur MVC (*Model View Controller*) didalamnya. MVC adalah prosedur dalam aplikasi yang mengisolasi beberapa data dari tampilan berdasarkan bagian-bagiannya, misalnya UI (*User Interface*), *Controller* dan *Data Manipulation*. Dengan digunakannya struktur *Model View Controller* (MVC) dapat membuat Laravel lebih mudah untuk dipahami dan mempersingkat proses pembuatan *web application prototype* (Gibran et al., 2024).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, Laravel adalah *framework* berbasis PHP yang dikembangkan oleh Taylor Otwell secara *open source* untuk membangun aplikasi *web* berskala kecil maupun besar. *Framework* ini sangat populer karena keunggulannya pada sisi fitur, kinerja, dan skalabilitas, serta kemampuannya dalam mempercepat proses pembuatan prototipe

aplikasi. Laravel menerapkan pola arsitektur MVC (*Model View Controller*) yang memisahkan antara manipulasi data, logika pengontrol, dan tampilan antarmuka. Berbeda dengan struktur MVC konvensional, Laravel menggunakan fitur *routing* sebagai jembatan wajib untuk menghubungkan permintaan (*request*) pengguna menuju *controller* yang sesuai.

Penerapan *framework* Laravel 12 dalam penelitian ini bertujuan untuk mempercepat pembuatan prototipe aplikasi web dengan struktur MVC yang rapi. Fitur *routing* pada Laravel dimanfaatkan sebagai jembatan untuk mengelola permintaan pengguna dan mengintegrasikan hasil perhitungan dari skrip Python ke dalam tampilan antarmuka *dashboard* secara dinamis.

### 3.15. *Visual Studio Code* (VS *Code*)

*Visual Studio Code* adalah sebuah teks editor ringan dan handal yang dibuat oleh Microsoft untuk sistem operasi *multiplatform*, artinya tersedia juga untuk versi Linux, Mac, dan Windows. Teks editor ini secara langsung mendukung bahasa pemrograman Javascript, Typescript, dan Node. Js, serta bahasa pemrograman lainnya dengan bantuan *plugin* yang dapat dipasang via *marketplace Visual Studio Code* seperti: C++, C#, Python, Go, Java, PHP, dst (Ningsih et al., 2022). *Visual Studio Code* adalah kode editor sumber yang dikembangkan oleh Microsoft untuk Windows, Linux dan macOS. Ini termasuk dukungan untuk *debugging*, kontrol *git* yang tertanam dan GitHub, penyorotan sintaksis, penyelesaian kode cerdas, *snippet*, dan *refactoring* kode. Ini sangat dapat disesuaikan, memungkinkan pengguna untuk mengubah tema, pintasan *keyboard*, preferensi,dan menginstal ekstensi yang menambah fungsionalitas tambahan (Aviyah & Nurvianti, 2024).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *Visual Studio Code* (VS *Code*) adalah perangkat lunak teks editor sumber terbuka (*source code editor*) yang ringan, handal, dan bersifat *multiplatform* (tersedia untuk Windows, Linux, dan macOS).

Visual Studio Code digunakan sebagai lingkungan pengembangan utama (*Integrated Development Environment*) dalam menyusun kode program SINEO.

Fitur-fitur pendukung di dalamnya memudahkan penulis dalam mengelola skrip PHP dan Python secara bersamaan, serta mempercepat proses pencarian kesalahan (*debugging*) pada aplikasi.

### 3.16. *Black Box Testing*

Pengujian *black box* adalah pengujian yang hanya menguji bagian luar dari perangkat lunak, contohnya seperti desain antarmuka. Hal tersebut merupakan salah satu alasan pengujian ini layak digunakan untuk menguji luaran suatu perangkat lunak (Parlika et al., 2020). *Black box testing* juga disebut pengujian perilaku, di mana struktur internal, logika perangkat lunak yang sedang diuji tidak diketahui oleh analis. Pengujian ini didasarkan pada spesifikasi persyaratan dan tidak perlu menganalisis kode. Pada dasarnya dilakukan dari sudut pandang pengguna akhir. Ini membantu mengidentifikasi spesifikasi yang tidak lengkap dan tidak terduga, sehingga dapat diperbaiki kemudian (Agarwal & Saxena, 2017). Pengujian *black box* dilakukan dengan hanya mengamati hasil eksekusi program tanpa melihat isi dari program tersebut (Saputra, 2018).

Berdasarkan sumber-sumber tersebut didapatkan kesimpulan bahwa, *black box testing* (pengujian kotak hitam) adalah metode pengujian perangkat lunak yang berfokus sepenuhnya pada fungsionalitas dan luaran (*output*) sistem dari sudut pandang pengguna akhir.

Metode *Black Box Testing* diterapkan pada tahap akhir pengembangan SINEO untuk memvalidasi fungsionalitas sistem dari sudut pandang pengguna. Penulis memfokuskan pengujian pada aspek masukan dan keluaran, seperti memastikan fitur sinkronisasi data NASA berjalan benar dan hasil prediksi AI tampil sesuai dengan format yang diharapkan tanpa harus menguji logika internal algoritma secara mendetail.

### 3.17. *System Usability Scale* (SUS)

*System Usability Scale* (SUS) merupakan metode evaluasi usabilitas yang dikembangkan oleh John Brooke pada tahun 1996. Metode ini digunakan untuk mengukur tingkat kebergunaan (*usability*), efisiensi antarmuka, serta kepuasan

subjektif pengguna akhir terhadap suatu produk perangkat lunak secara kuantitatif. SUS dikenal secara luas dalam rekayasa sistem informasi karena instrumennya yang terstandarisasi, efisien, dan memiliki tingkat keandalan (*reliability*) yang tinggi meskipun dievaluasi menggunakan sampel responden dalam jumlah yang relatif terbatas (Welda et al., 2020).

Instrumen pengujian usabilitas metode SUS terdiri dari 10 butir pertanyaan terstandarisasi yang disajikan secara berselang-seling antara pernyataan bernada positif (butir ganjil) dan pernyataan bernada negatif (butir genap). Penyusunan item pertanyaan berselang-seling ini bertujuan untuk meminimalisir bias respons dari responden sewaktu mengisi kuesioner. Setiap butir instrumen dinilai menggunakan skala *Likert* 5 tingkat, yaitu Sangat Tidak Setuju (STS = 1), Tidak Setuju (TS = 2), Kurang Setuju (KS = 3), Setuju (S = 4), dan Sangat Setuju (SS = 5).

### 3.17.1. Prosedur Perhitungan Skor SUS

Menurut (Welda et al., 2020) kalkulasi skor untuk setiap lembar kuesioner yang diisi oleh responden diolah menggunakan aturan komputasi baku dari metode SUS. Pada butir pertanyaan bernomor ganjil (P1, P3, P5, P7, P9), skor kontribusi diperoleh dengan mengurangi nilai skala jawaban responden dengan angka 1 (Skor = Skala - 1). Sebaliknya, pada butir pertanyaan bernomor genap (P2, P4, P6, P8, P10), skor kontribusi diperoleh dengan mengurangi angka 5 dengan nilai skala jawaban responden (Skor = 5 - Skala). Total akumulasi skor kontribusi dari ke-10 pertanyaan (yang berada dalam rentang 0–40) kemudian dikalikan dengan faktor pengali sebesar 2,5 untuk mengonversikan nilai ke dalam skala standar SUS bernilai 0 hingga 100. Prosedur kalkulasi skor untuk setiap lembar kuesioner yang diisi oleh responden diolah berdasarkan aturan komputasi matematis dari metode SUS.

### 3.17.2. Interpretasi Hasil Evaluasi SUS

Menurut (Welda et al., 2020) untuk memberikan pemaknaan terhadap nilai rata-rata skor akhir SUS yang diperoleh, dilakukan komparasi berdasarkan kriteria standar

penilaian usabilitas yang dikembangkan. Interpretasi hasil evaluasi diukur berdasarkan empat parameter utama:

1. *Acceptability Ranges* (Tingkat Penerimaan): Mengukur tingkat penerimaan pengguna terhadap sistem, yang dibagi menjadi tiga kategori utama: *Not Acceptable* (< 60), *Marginal* (60 - 70), dan *Acceptable* (> 70).
2. *Grade Scale* (Skala Kelayakan): Mengelompokkan peringkat kelayakan sistem ke dalam skala huruf: *Grade F* (< 51), Grade D (51 – 67,9), Grade C (68 – 73,9), *Grade B* (74 – 80,2), dan *Grade A* (> 80,3).
3. *Adjective Rating* (Predikat Kualitatif): Menentukan predikat kata sifat kualitas antarmuka berdasarkan persepsi pengguna, mulai dari *Worst Imaginable*, *Poor*, *OK*, *Good, Excellent*, hingga *Best Imaginable.*
4. *Percentile Rank Grade* (Peringkat Persentil): Memetakan posisi kualitas sistem berdasarkan distribusi skor persentil relatif terhadap aplikasi web secara umum.

Penerapan teori *System Usability Scale* (SUS) ini bertindak sebagai landasan metodologis pada hasil dan pembahasan untuk mengevaluasi antarmuka *platform* SINEO secara empiris, guna membuktikan bahwa integrasi visualisasi *Traffic Light System* (TLS) dan penyederhanaan data astronomi NASA mampu menyajikan informasi yang efisien, mudah dipahami, serta memberikan kepuasan yang tinggi bagi penggunanya.