BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1. Tinjauan Pustaka

Penelitian *Deep Learning* menggunakan *TensorFlow Lite* telah dilakukan dengan tujuan mengklasifikasikan gambar yang dapat diimplementasikan ke dalam perangkat bergerak. Penelitian sebelumnya menunjukan bahwa *Deep Learning* adalah teknologi yang dapat memecahkan masalah dengan cara berpikir seperti otak manusia. Dengan menggunakan pendekatan ini maka performa aplikasi pengenalan visual seperti klasifikasi dan deteksi gambar dapat meningkat secara nyata.

Penelitian yang berjudul Deteksi Jenis Sayuran dengan Tensorflow Dengan Metode *Convolutional Neural Network* oleh Agung Rizqi Hidayat dan Veronica Lusiana pada tahun 2022 bertujuan untuk melakukan klasifikasi jenis sayuran dengan data sebanyak 2550 gambar yang mencakup 17 jenis sayuran. Dalam penelitian ini, *TensorFlow* digunakan untuk mengintegrasikan model CNN sehingga dapat dijalankan pada perangkat bergerak, yang memungkinkan untuk deteksi jenis sayuran secara *realtime*. Hasil penelitian menunjukan bahwa model tersebut mencapai tingkat akurasi sebesar 86% [8]. Meskipun penelitian ini membuktikan kemampuan model CNN dalam mengklasifikasikan jenis sayuran dengan akurasi yang baik, namun penelitian ini hanya fokus pada pembuatan model dan tidak memberikan rincian tentang pembuatan aplikasi yang terkait.

Penelitian berikutnya yang berjudul Rancang Bangun Aplikasi Edukasi Tuberkulosis Menggunakan Metode Scrum oleh Sri Puji Utami, Kyky Eviyanti, Wening Sari dan Sri Chusri Haryanti pada tahun 2022 bertujuan untuk menciptakan aplikasi Yes TBCare berbasis Android untuk penyuluhan kepada pasien TBC. Aplikasi yang dibangun melalui metode *scrum* tersebut telah diuji menggunakan *black box testing* dan berhasil memperoleh validasi sebesar 100%. Semua fitur dalam aplikasi berfungsi dengan baik dan pengujian menggunakan *UEQ* (*User Experience Questionnaire*) menunjukan hasil yang positif dari

pengguna dengan nilai diatas 0,8 [9]. Penelitian ini membuktikan bahwa dengan metode *scrum*, dapat menghasilkan aplikasi yang memberikan edukasi tentang Tuberkulosis yang dimana dalam pengembangannya, perubahan dapat dilakukan sesuai dengan keinginan pengguna yang disampaikan dalam *sprint review,* tetapi pada penelitian tersebut tidak melakukan pengujian menggunakan *heuristic evaluation* agar mendapatkan hasil pengujian langsung dari ahli dibidangnya.

Penelitian lain yang berjudul Perancangan Aplikasi Mobil Derek Berbasis Android Menggunakan Metode Scrum oleh Kukuh Primadito Raharjo, Gita Fadila Fitriana, dan Novian Adi Prasetyo pada tahun 2022 bertujuan untuk memanggil mobil derek terhadap kendaraan yang mengalami kerusakan mesin dengan menggunakan aplikasi pencarian mobil derek yang akan memudahkan bagi pelanggan dan supir derek. Aplikasi yang dibuat dengan metode *scrum* ini memungkinkan pengguna untuk melakukan pemesanan mobil derek dengan fitur-fitur seperti menampilkan peta untuk menentukan lokasi penjemputan, lokasi tujuan, lokasi mobil derek dan menampilkan informasi ketersedian mobil derek di sekitar pelanggan. Aplikasi yang dibangun akan diuji menggunakan metode *black box*, dengan persentase hasil pengujian sebesar 99,16% untuk layak digunakan dan nilai rata-rata dengan System *Usability Scale* (SUS) sbesar 76,6% [10]. Penelitian ini membuktikan bahwa dengan metode *scrum* dapat menghasilkan sebuah aplikasi yang dapat membuat pelanggan puas terhadap fitur yang diberikan, tetapi pada penelitian tersebut tidak melakukan pengujian menggunakan *heuristic evaluation* agar mendapatkan hasil pengujian langsung dari ahli dibidangnya.

Penelitian selanjutnya yang berjudul Pengujian Black Box Pada Aplikasi Pembelajaran Bahasa Mandarin Berbasis Android oleh Putri Saman dan Chanifah Indah Ratnasari pada tahun 2022 bertujuan untuk melakukan pengujian aplikasi pembelajaran bahasa Mandarin berbasis Android menggunakan *black box* dengan teknik *Boundary Value Analysis* dengan hasil setiap skenario berstatus berhasil [11]. Pada penelitian ini membuktikan bahwa dengan menggunakan metode *black box*, penguji dapat dengan mudah mengevaluasi alur fungsi aplikasi dengan hasil yang diharapkan dan dapat memberikan saran perbaikan yang dapat meningkatkan

kemudahan penggunaan aplikasi. Namun, pada penelitian ini hanya berfokus dengan satu metode pengujian dan tidak melakukan pengujian yang lain agar mendapatkan hasil yang lebih valid lagi.

Penelitian selanjutnya yang berjudul *Development of An Android Based Real-Time Image Recognition Application Using Convolutional Neural Network Algorithm: Addressing Challenges in Mother Tongue-Based of Multilingual Education* oleh Jasten Keneth D. Trecene pada tahun 2019 membahas tentang pembuatan aplikasi Android menggunakan konsep algoritma *Convolutional Neural Network* (CNN) untuk deteksi sayuran secara *real-time*. Penelitian ini bertujuan untuk memberikan materi tambahan kepada guru MTB-MLE dikarenakan kurangnya materi yang ditulis dalam bahasa ibu dan kosakata dalam mengenali sayuran lokal. Aplikasi tersebut diintegrasikan dengan Tensorflow dengan tujuan untuk dapat dijalankan pada perangkat bergerak, dan berhasil menghasilkan model dengan akurasi rata-rata 92,8% [12]. Penelitian ini mengimplementasikan model CNN yang mampu mengenali sayuran lokal dengan tingkat akurasi yang tinggi. Namun, penelitian lebih berfokus pada pembuatan model dan tidak secara rinci menjelaskan proses pembangunan aplikasi Android tersebut.

Dengan memperhatikan penjabaran di atas, maka diperoleh ringkasan penelitian yang berkesesuaian seperti yang ditunjukkan pada Tabel 2.1 berikut ini:

Tabel 2.1 Penelitian terdahulu

No	Judul	Peneliti	Metode	Hasil
1	Deteksi Jenis Sayuran dengan Tensorflow Dengan Metode Convolutional Neural Network [8]	Agung Rizqi Hidayat dan Veronica Lusiana	*Convolutional Neural Network* (CNN)	Penelitian yang telah dilakukan untuk membuat aplikasi Android yang dapat mendeteksi jenis sayuran yang diintegrasikan melalui *TensorFlow* menunjukan hasil yang cukup baik dengan nilai akurasi tertinggi sebesar 86%
2	Rancang Bangun Aplikasi Edukasi Tuberkulosis Menggunakan Metode Scrum [9]	Sri Puji Utami, Kyky Eviyanti, Wening Sari dan Sri Chusri Haryanti	Scrum	Penelitian yang telah dilakukan untuk membuat aplikasi Android yang dapat memberikan edukasi mengenai Tuberkulosis menunjukan hasil pengujian yang positif. Pengujian menggunakan *black box* menyatakan bahwa aplikasi tersebut valid 100% dan semua fitur aplikasi berfungsi dengan baik serta pengujian menggunakan *UEQ* mendapatkan nilai diatas 0.8
3	Perancangan Aplikasi Mobil Derek Berbasis Android Menggunakan Metode Scrum [10]	Kukuh Primadito Rahajo, Gita Fadila Fitriana, dan Novian Adi Prasetyo	Scrum	Penelitian yang telah dilakukan untuk membuat aplikasi Android yang dapat melakukan pencarian mobil derek yang terintegrasi dengan *maps* menunjukan hasil pengujian yang sangat positif. Pengujian menggunakan *black box* menghasilkan persentase sebesar 99,16%, menunjukan bahwa aplikasi tersebut berfungsi sangat baik. Selain itu, pengujian menggunakan *System Usability Scale* (SUS) menghasilkan nilai rata-rata sebesar 76,6%

No	Judul	Peneliti	Metode	Hasil
4	Pengujian Black Box Pada Aplikasi Pembelajaran Bahasa Mandarin Berbasis Android [11]	Putri Saman dan Chanifah Indah Ratnasari	*Black Box*	Penelitian yang dilakukan untuk menguji aplikasi pembelajaran bahasa Mandarin berbasis Android menunjukan hasil yang bagus. Pengujian dilakukan menggunakan *black box* dengan teknik *Boundary Value Analysis*, di mana setiap skenario pengujian berhasil
5	*Development of An Android Based Real-Time Image Recognition Application Using Convolutional Neural Network Algorithm: Addressing Challenges in Mother Tongue-Based of Multilingual Education* [12]	Jasten Keneth D. Trecene	*Convolutional Neural Network* (CNN)	Penelitian yang dilakukan menggunakan metode *Convolutional Neural Network* (CNN menunjukan hasil yang baik dalam memprediksi jenis sayuran. Model CNN yang dikembangkan mencapai tingkat akurasi sebesar 92,8%

Berdasarkan data yang tercantum dalam Tabel 2.1, dapat disimpulkan bahwa penggunaan metode *Convolutional Neural Network* menghasilkan performa yang sangat baik dalam melakukan klasifikasi gambar dengan tingkat akurasi yang tinggi. Metode ini juga dapat dengan mudah diintegrasikan ke dalam perangkat bergerak (*mobile device*) menggunakan *TensorFlow Lite*. Disamping itu, penerapan metode *scrum* dalam pengembangan aplikasi memungkinkan adanya fleksibilitas untuk menyesuaikan dengan keinginan pengguna. Hal ini memastikan bahwa aplikasi yang dikembangkan dapat memenuhi kebutuhan dan harapan pengguna. Penelitian yang menjadi fokus utama adalah penelitian pertama yang tercantum pada tabel karena menggunakan metode yang serupa dan diintegrasikan menggunakan *TensorFlow*. Perbedaan terletak pada objek yang akan dideteksi dan studi kasus yang dilakukan.

2.2. Landasan Teori

Dalam penelitian ini ada beberapa landasan teori yang digunakan, di antaranya sebagai berikut ini.

2.2.1. Visi Komputer

Visi komputer atau *computer vision* adalah sebuah metode yang di mana komputer atau mesin memiliki kemampuan untuk melihat dan memproses informasi visual. Dalam Visi komputer, data yang berasal dari foto, kamera atau gambar diubah menjadi hasil pengambilan keputusan yang memiliki tujuan spesifik. Tujuan utama dari visi komputer adalah untuk memungkinkan mesin atau komputer meniru kemampuan perseptual otak dan mata manusia bahkan melampaui kemampuannya dalam beberapa konteks [13].

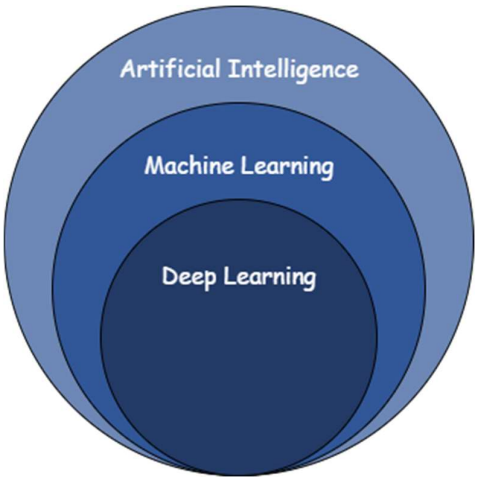
Proses transformasi yang dilakukan dalam visi komputer memungkinkan data yang dimasukan, seperti foto atau gambar menyediakan informasi yang dapat digunakan dalam pengambilan keputusan terkait gambar tersebut. Namun, terdapat perbedaan dengan manusia dalam hal pemahaman, penelitian dan perbandingan informasi objek secara langsung menggunakan pengalaman yang diperoleh selama hidup didunia. Pada sistem penglihatan mesin komputer, informasi yang diterima hanyalah kumpulan angka yang berasal dari media input data seperti disket atau kamera [13].

2.2.2. *Deep Learning*

Deep Learning adalah sebuah teknik yang terinspirasi dari struktur otak manusia yang termasuk ke dalam cabang ilmu *Artificial Intelligence* dan *Machine Learning*. Teknik ini memungkinkan sistem untuk belajar dan beradaptasi dengan data dalam jumlah yang besar [14]. Sebelum membahas topik *Deep Learning* lebih lanjut, akan dibahas mengenai *Artificial Intelligence* dan *Machine Learning* terlebih dahulu.

Kecerdasan buatan (*Artificial Intelligence* atau AI) adalah salah satu bidang ilmu komputer yang mempelajari bagaimana mesin dapat melakukan tugas seperti yang dilakukan oleh manusia. Menurut [15] menunjukan bahwa dalam tugas-tugas yang membutuhkan analisis tingkat tinggi dan kompleksitas rendah dengan tingkat ketidakpastian, kecerdasan buatan mampu mengungguli.

Selanjutnya yaitu *Machine Learning* (ML). Sebelumnya, *Artificial Intelligence* hanya menggambarkan konsep umum suatu tindakan tanpa rincian yang lebih lanjut, *Machine Learning* merupakan pengembangan dari *Artificial Intelligence* yang memungkinkan komputer belajar tanpa perlu diprogram secara eksplisit [14]. Pada *Machine Learning*, dibutuhkan data sebagai model untuk proses pembelajaran.



Gambar 2.1 Hierarki AI, ML, dan *deep learning*

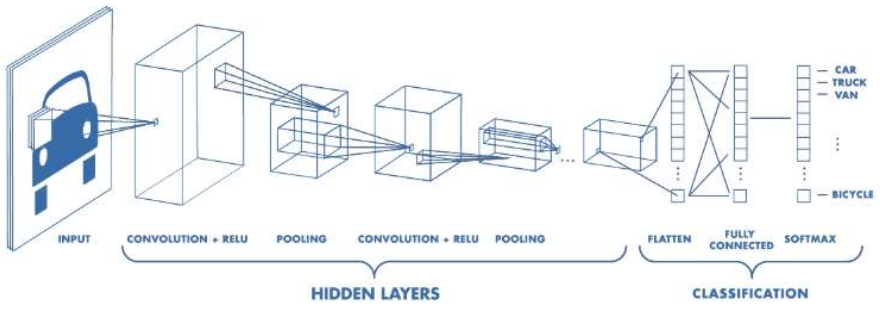
Setelah mengerti dengan konsep *Artificial Intelligence* dan *Machine Learning*, terdapat bidang yang lebih dalam yaitu *Deep Learning*. *Deep Learning* adalah perkembangan dari konsep Jaringan Syaraf Tiruan (*Neural Network* atau

NN) atau yang dikenal juga sebagai *Artificial Neural Network* (ANN). Model ini terinspirasi oleh struktur otak manusia. Pada Gambar 2.1, terlihat hierarki *Artificial Intelligence* (AI), *Machine Learning* (ML) dan *Deep Learning*, yang menunjukan bahwa *Deep Learning* termasuk dalam bagian *Machine Learning*, sementara *Machine Learning* sendiri adalah bagian dari *Artificial Intelligence*.

2.2.3. *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan salah satu bagian penting dari *Deep Neural Network* yang digunakan untuk melakukan pengenalan gambar. Metode ini termasuk hasil pengembangan dari *Multi-Layer Perceptron* (MLP). Namun, perbedaannya terletak pada kemampuan CNN untuk menerima data dua dimensi, khususnya gambar, sementara MLP hanya menerima data masukan satu dimensi [16].

CNN merupakan sebuah konstruksi matematika yang umumnya terdiri dari tiga jenis lapisan (*layer*), yaitu *Convolutional Layer*, *Pooling Layer*, dan *Fully Connected Layer* [17].



Gambar 2.2 Arsitektur CNN [17]

Pada Gambar 2.2, proses layer CNN dibagi menjadi dua bagian utama yaitu *Hidden Layer* dan *Classification*. Pada *Classification* akan bertugas untuk mengklasifikasikan tiap data yang telah disiapkan dan akan diekstraksi menjadi beberapa bagian yaitu *flatten*, *Fully Connected Layer* dan *softmax*. Sementara pada *Hidden Layer* berguna sebagai tempat translasi untuk mengubah *input* menjadi *features* berdasarkan ciri atau keunikan pada *input* yang dimana akan

diubah menjadi angka-angka pada *vector*. Terdapat dua lapisan pada *Hidden Layer* yaitu lapisan *Pooling* dan *Convolutional Layer* [8].

2.2.4. Android

Android merupakan sebuah sistem operasi yang dirancang oleh Google untuk berbagai perangkat seperti *tablet*, *smartphone*, jam tangan, televisi dan mobil. Android telah menjadi pilihan sistem operasi bagi banyak produsen perangkat elektronik, yang menggunakannya pada berbagai produk yang mereka hasilkan. Selain itu, Android juga mempunyai toko aplikasi atau yang lebih sering dikenal dengan *playstore* yang memiliki jumlah pengguna lebih dari 3 miliar.

Android 13 merupakan versi terbaru dari Android yang dirilis pada 16 agustus 2022. Sejak awal peluncurannya pada tahun 2008 dengan versi 1.0, Android telah mengalami banyak perkembangan. Namun, pada saat itu versi 1.0 belum resmi digunakan. Perkembangan Android sendiri memiliki sejarah yang resmi diluncurkan oleh Google, yang tertera pada Tabel 2.2.

Tabel 2.2 Sejarah versi dari Android

Versi	Nama Kode	Tanggal Rilis	API Level	DVM/ ART	Fitur Terbaru
13	13	16 Agustus 2022	33	ART	-
12L	12	7 Maret 2022	32	ART	*Modifications to the user interface to tailor it to larger screens and improvements specific for Chromebooks, tablets, foldable phones and desktop-sized screens*
12	12	4 Oktober 2021	31	ART	a. *Easier Wi-Fi sharing* b. *Material You* c. AVIF *image support.* d. *Scrolling Screenshot*
11	11	8 September 2020	30	ART	a. *Predictive Tool* b. *Chat Bubbles* c. *Device Control* d. *One-time Permission*

Versi	Nama Kode	Tanggal Rilis	API Level	DVM/ ART	Fitur Terbaru
					e. *Screen Recorder*
10	10	3 September 2019	29	ART	a. *Dark Theme* b. *Live Action* c. *Sound Amplifier* d. *Privacy & Security* e. *Digital Wellbeing* f. *Smart Reply*
9	*Pie*	6 Agustus 2018	28	ART	a. *Adaptive Battery* b. *Adaptive Brightness*
8.0 – 8.1	*Oreo*	25 Oktober 2017	26 - 27	ART	*Picture-in-Picture*
7.1 – 7.1.2	*Nougat*	22 Agustus 2016	24 - 25	ART	a. *GIF Keyboard* b. *Multi-window*
6.0 – 6.0.1	*Marshmallow*	5 Oktober 2015	23	ART	a. *Permissions* b. *Battery* c. *Now On Tap*
5.1 – 5.1.1	*Lollipop*	12 November 2014	21 - 22	ART	a. *Multiscreen* b. *Notification* c. *Material Design*
4.4 – 4.4.4	*Kitkat*	31 Oktober 2013	19 - 20	DVM (and ART 1.6.0)	a. *Voice: Ok Google* b. *Smart Dialer* c. *Immersive Design*
4.1 – 4.3.1	*Jellybean*	9 Juli 2012	18	DVM	a. *Account Switching* b. *Google Now* c. *Actionable Notifications*
4.0 – 4.6	*Ice Cream Sandwich*	19 Oktober 2011	15	DVM	a. *Android Beam* b. *Data Usage Control* c. *Custom Home Screen*
3.0 – 3.2.6	*HoneyComb*	22 Februari 2011	11 - 13	DVM	a. *System Bar* b. *Quick Settings* c. *Tablet-Friendly Design*
2.3 - 2.3.7	*Gingerbread*	9 Februari 2011	9 - 10	DVM	a. *Battery Management* b. *NFC* c. *Gaming APIs*
2.2 – 2.23	*Froyo*	20 Mei 2010	8	DVM	a. *Dalvik JIT* b. *Portable Hotspot* c. *Voice Action*

Versi	Nama Kode	Tanggal Rilis	API Level	DVM/ ART	Fitur Terbaru
2.0 – 2.1	*Eclair*	26 Oktober 2009	5	-	a. *Speech-to-Text* b. *Home Screen Customization* c. *Google Maps Navigation*
1.6	*Donut*	15 September 2009	4	-	a. *Quick Search Box* b. *Android Market* c. *Screen Size Diversity*
1.5	*Cupcake*	27 April 2009	3	-	-

Menurut [18], Android telah mengalami perkembangan yang signifikan bukan hanya sebagai sistem operasi untuk perangkat telepon pintar. Namun, terdapat beberapa faktor yang menjadi alasan mengapa Android dipilih sebagai platform pengembangan aplikasi:

a. Android memiliki keunggulan sebagai sistem operasi *open source*. Hal ini menunjukan bahwa produk mendukung untuk dikembangkan dengan aman, sehingga banyak pengembang komunitas yang tertarik.
b. Android memiliki keunggulan dengan adanya Google Play Store. Hal ini memberikan kemudahan bagi pengembang dalam mendistribusikan aplikasi mereka ke pasar yang memiliki miliaran pengguna.
c. Android juga menyediakan *Integrated Development Environment* (IDE) resmi yang bernama Android Studio. IDE ini didasarkan pada Intellij IDEA, sebuah perusahaan yang berasal dari Rusia dan secara langsung didukung oleh Google.
d. Android mempunyai *Development Kit* yaitu *Software Developer Kit* (SDK) yang didalamnya terdapat berbagai *tools* sepert *sample code*, *software libraries*, *emulator*, *debugger*, dokumentasi dan tutorial.

Platform Android mempunyai sebuah IDE resmi yang sudah dibahas pada poin diatas, yakni Android Studio. Android Studio adalah *code editor* yang dilengkapi dengan fitur-fitur yang dapat meningkatkan efisiensi dalam pengembangan aplikasi Android, sehingga menjadikannya sebagai *code editor*

yang *powerful*. Android Studio dapat digunakan dengan spesifikasi sistem sebagai berikut ini.

Tabel 2.3 Spesifikasi sistem Android Studio

Windows
1. Disarankan memiliki RAM 8 GB atau lebih 2. Resolusi layar minimal harus mencapai 1280 x 800 3. Disarankan menggunakan prosesor Intel i5 atau yang lebih tinggi seperti seri U atau yg lebih canggih 4. Diperlukan ruang disk minimal sebesar 4 GB yang tersedia
Mac
1. Sistem operasi yang disarankan adalah MacOS® 10.14 (Mojave) atau versi yang lebih baru 2. Diperlukan chip berbasis ARM, atau Intel Core generasi ke-2 atau yang lebih baru dengan dukungan Hypervisor.Framework 3. Disarankan memiliki RAM 8 GB atau lebih 4. Diperlukan ruang disk minimal sebesar 8 GB yang tersedia (IDE + Android SDK + Android Emulator) 5. Resolusi layar minimal harus mencapai 1280 x 800
Linux
1. Dibutuhkan sistem operasi Linux 64-bit yang mendukung GNOME, KDE, atau Unity DE, GNU C Library (glibc) 2.31 atau yang lebih baru 2. Perlu menggunakan arsitektur CPU x86_64; Intel Core generasi ke-2 atau yang lebih baru, atau prosesor AMD yang mendukung AMD Virtualization (AMD-V) dan SSSE3 3. Disarankan memiliki RAM 8 GB atau lebih 4. Diperlukan ruang disk minimal sebesar 8 GB yang tersedia (IDE + Android SDK + Android Emulator) 5. Resolusi layar minimal harus mencapai 1280 x 800
Chrome OS
1. Disarankan memiliki RAM 8 GB atau lebih 2. Diperlukan ruang disk minimal sebesar 4 GB yang tersedia 3. Resolusi layar minimal harus mencapai 1280 x 800 4. Disarankan menggunakan prosesor Intel i5 atau yang lebih tinggi seperti seri U atau yg lebih canggih Perangkat yang direkomendasikan 1. Acer: Chromebook 13/Spin 13, Chromebox CXI3, Chromebook 712 [C871] 2. Lenovo: Yoga C630 Chromebook, Flex 5 Chromebook 3. ASUS: Chromebox 3, Chromebook Flip C436FA 4. HP: Chromebook x360 14, Chromebox G2, Chromebook x360 14c 5. Dell: Inspiron Chromebook 14, Latitude 5300 2-in-1 Chromebook Enterprise, Latitude 5400 Chromebook Enterprise 6. CTL: Chromebox CBx1 7. ViewSonic: NMP660 Chromebox

2.2.5. Kotlin

Beberapa bahasa pemrograman yang dapat dipakai untuk mengembangkan aplikasi Android, antara lain Java, C++, Kotlin, dan Dart yang merupakan bahasa terbaru. Pada Google I/O 2019, Google mengumumkan bahwa Kotlin resmi diakui sebagai bahasa pemrograman untuk mengembangkan aplikasi Android. Hal ini menandakan bahwa Kotlin akan mendapatkan lebih banyak perhatian dan dukungan dari Google dalam pengembangan aplikasi Android [19].

Kotlin merupakan bahasa pemrograman yang memiliki beberapa kelebihan yang sangat mempermudah proses pengembangan perangkat lunak. Kotlin juga termasuk ke dalam bahasa yang ekspresif dan ringkas yang dapat mengurangi kesalahan kode yang sering terjadi dan membuatnya mudah diintegrasikan ke dalam aplikasi yang ada. Selain itu, terdapat beberapa karakteristik yang dimiliki oleh Kotlin yaitu [20]:

a. *Modern and Concise*

Kotlin terkenal sebagai bahasa pemrograman modern yang ringkas, memungkinkan penulisan kode yang lebih singkat dibandingkan dengan bahasa yang lain.

Gambar 2.3 Perbandingan Kotlin dan Java

b. *Pragmatic*

Kotlin merupakan hasil pengembangan berdasarkan permasalahan-permasalahan yang sering dihadapi oleh para pengembang di JetBrains, yang terus memperbaharui fitur-fitur yang ada untuk memberikan solusi praktis bagi seorang pengembang.

c. *Safe*

Kotlin memiliki kemampuan untuk mencegah kesalahan dengan melakukan pemeriksaan saat kompilasi. Selain itu, Kotlin juga mengatasi masalah *NullPointerException* (NPE) atau yang sering dijuluki sebagai "*The billion dollar mistake*" dengan membedakan objek yang dapat bernilai *null* atau tidak dapat bernilai null pada saat objek tersebut dibuat.

d. *Statically Typed*

Dalam Kotlin pada saat kompilasi, setiap ekspresi yang ada didalam program dapat dikenali dan dapat memungkinkan untuk tidak menuliskan tipe variabel secara eksplisit.

e. *Free and Open-Sources*

Kotlin juga termasuk ke dalam kategori perangkat lunak *open-source* yang memberikan manfaat besar bagi para pengembang, sehingga banyak pengembang yang ikut berkontribusi yang membuat Kotlin menjadi kian *powerful*.

2.2.6. Python

Python adalah bahasa pemrograman serbaguna yang dikembangkan oleh Guido van Rossum. Python dapat digunakan di berbagai platform. Python memiliki beberapa kelebihan diantaranya dapat menangani kesalahan (*exception handling*), sintaks mudah dibaca dan dimengerti (*readability*), sehingga menjadikan Python sebagai bahasa pemrograman yang sangat mudah dipahami dan fleksibel [21].

2.2.7. Firebase

Firebase adalah platform pengembangan aplikasi *web* dan *mobile*. Awalnya Firebase dikembangkan oleh sebuah perusahaan *startup* bernama Envolve buatan James Tampilin dan Andrew Lee pada tahun 2011, kemudian

pada tahun 2014 Google mengumumkan untuk mengakuisisi Firebase [22]. Pada September 2022 Firebase telah memiliki 24 produk.

Dalam pengembangan perangkat lunak, Firebase menawarkan beberapa produk populer seperti Firebase *Authentication*, Firebase *Realtime Database*, Firebase *Cloud Firestore*, Firebase *Cloud Storage* dan Firebase *Machine Learning* yang dapat digunakan untuk aplikasi yang membutuhkan teknologi *Machine Learning*.

Firebase *Authentication* merupakan produk Firebase untuk mempermudah pembuatan sistem autentikasi yang aman. Produk ini menyediakan beberapa metode autentikasi, seperti email/sandi, nomor telepon dan autentikasi menggunakan pihak ketiga seperti Google, Play Games, Facebook, Apple, Github, Microsoft, Twitter dan Yahoo. Selanjutnya yaitu Firebase *Realtime Database* adalah sebuah produk Firebase yang digunakan untuk menyimpan dan menyinkronkan data secara *real-time* antar pengguna dengan aturan *NoSQL*. Selain *Realtime Database*, Firebase juga memiliki database lain yaitu Firebase *Cloud Firestore*. Perbedaan kedua database tersebut tergantung dari kebutuhan pengguna, jika pengguna memerlukan *query* yang sedikit rumit maka *Cloud Firestore* sangat cocok digunakan dan sebaliknya jika tidak memerlukan *query* maka dapat menggunakan *Realtime Database*. Selanjutnya yaitu Firebase *Cloud Storage* merupakan sebuah produk Firebase untuk menyimpan dan menyajikan konten seperti foto dan video kepada pengguna dengan cepat dan mudah [23].

Google meluncurkan Firebase ML Kit sebagai alat untuk mengembangkan aplikasi yang menggunakan teknologi *Machine Learning*. Pada tahun 2018, Google meluncurkan ML Kit acara tahunannya yaitu Google I/O 2018 [24]. Seiring berjalannya waktu, terjadi perubahan nama pada produk Firebase ML Kit ini menjadi Firebase *Machine Learning* dan ML Kit menjadi produknya sendiri [25].

Firebase *Machine Learning* memiliki API yang berfungsi baik di *cloud* atau di perangkat. Dalam Firebase ML terdapat 3 jenis yang berbeda yaitu *custom models*, *Cloud* AutoML *Vision Edge* dan *Cloud Vision* APIs. *Custom Models*, Firebase ML hanya mendownload model saat diperlukan dan secara otomatis

memperbarui ke versi terbaru. *Cloud* AutoML *Vision Edge* digunakan untuk pelabelan gambar atau model deteksi objek yang lebih khusus. Selanjutnya *Cloud Vision* APIs dapat digunakan untuk mengenali teks, melabeli gambar dan mengenali tempat terkenal. Selain itu, untuk mengelola model yang telah dibuat sehingga dapat digunakan dalam Firebase *Machine Learning*, Google juga memiliki produk lain yaitu *TensorFlow* [26].

2.2.8. *TensorFlow Lite*

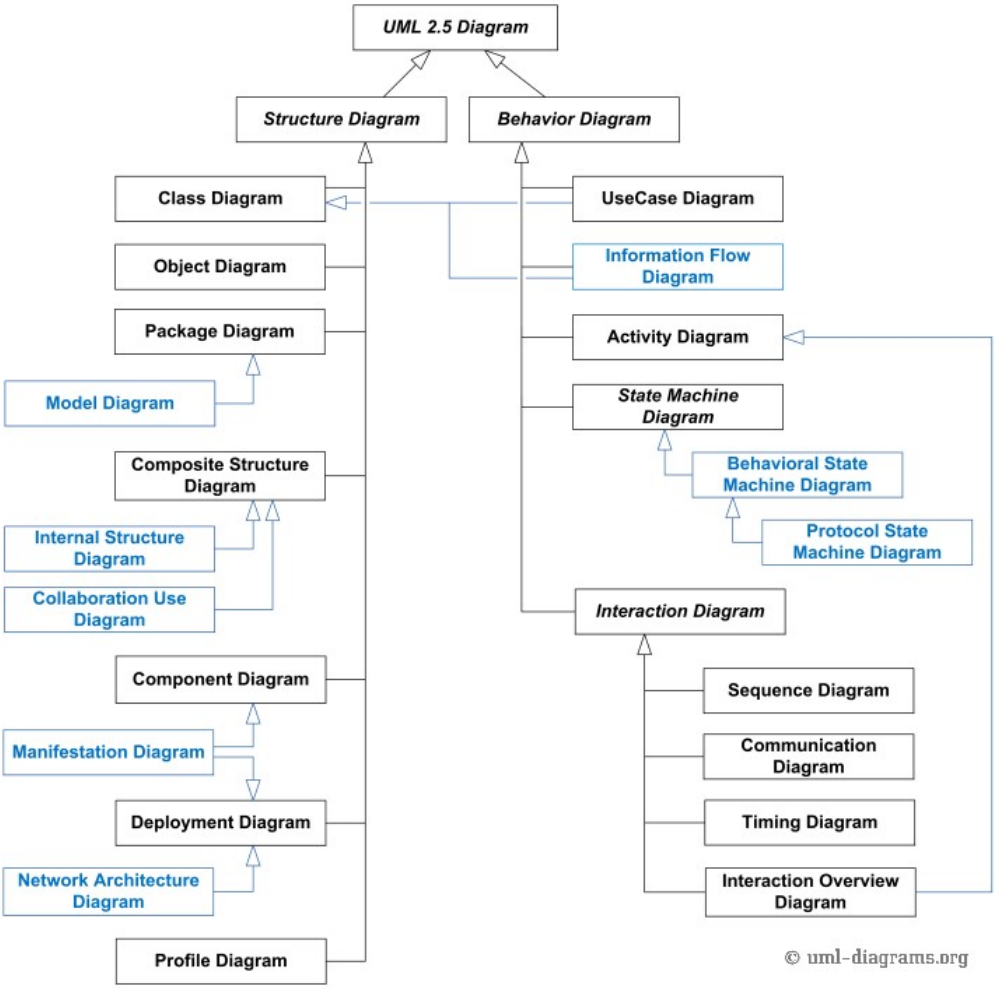
TensorFlow Lite adalah sebuah *platform end-to-end* yang dikembangkan oleh Google untuk keperluan *machine learning*. Platform ini dirancang khusus untuk membantu pengembang menjalankan model *machine learning* di perangkat seluler. *TensorFlow Lite* memiliki beberapa fitur utama salah satunya adalah pengoptimalan untuk pembelajaran mesin di perangkat seluler [27]. Selain itu, *TensorFlow Lite* dapat dihubungkan dengan Firebase *Machine Learning*. Seluruh alur kerja *TensorFlow Lite*, dari tahap awal pengembangan hingga tahap penyebaran ke perangkat, termasuk tahap pengoptimalan juga dijelaskan pada Gambar 2.4 berikut.

Gambar 2.4 Proses kerja tensorFlow [28]

Pada tahap awal, pilihan dilakukan untuk menggunakan model baru dalam proses ini, atau dalam situasi tertentu, memungkinkan untuk melatih kembali model yang sudah ada. Selanjutnya, *TensorFlow Lite Converter* akan digunakan untuk mengonversi model *TensorFlow Lite* menjadi format *compressed flat buffer*. Setelah berhasil mengubah model tersebut, langkah selanjutnya yaitu mengambil *file* .tflite yang sudah terkompresi dan mengunggahnya ke Firebase, yang akan terhubung dengan perangkat seluler atau ditanamkan secara langsung pada perangkat [28].

2.2.9. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah sebuah bahasa yang digunakan untuk membuat *blueprint* atau rancangan sistem. Tujuan dari penggunaan UML adalah untuk memvisualisasikan, mengembangkan dan mendokumentasikan sistem dengan tujuan untuk memudahkan pengembangan perangkat lunak dan memenuhi kebutuhan pengguna secara efektif, tepat dan lengkap [29]. Pada tahun 2015, Diagram UML versi 2.5 dirilis, yang mengklasifikasikan jenis diagram menjadi dua kategori yaitu *structure diagram* dan *behavior diagram*, sebagaimana ditampilkan pada Gambar 2.5 [30].



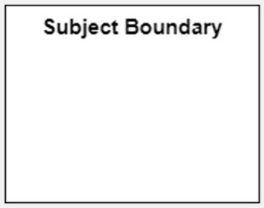
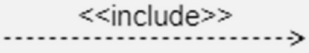
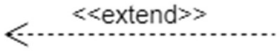
Gambar 2.5 Klasifikasi digram UML [30]

Dalam penelitian ini, menggunakan diagram UML yang hanya berfokus pada kategori *behavior diagram* meliputi *Use Case Diagram*, *Activity Diagram*, dan *Sequence Diagram*.

a. *Use Case Diagram*

Use Case diagram merupakan jenis diagram yang digunakan untuk menjelaskan operasi fundamental sistem informasi. Use Case menggambarkan interaksi sistem bisnis dan sekitarnya [31]. Adapun elemen dari *Use Case Diagram* ini meliputi:

Tabel 2.4 Simbol dan penjelasan pada *use case diagram*

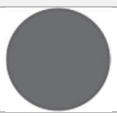
Notasi	Nama	Deskripsi
	*Use case*	*Use case* memiliki peran besar dalam bagaimana sebuah sistem berfungsi. *Use case* biasanya dilabeli dengan kata kerja – frasa kata benda
	*Actor*	*Actor* merujuk pada elemen yang mewakili pengguna, entitas atau sistem lain yang terlibat dalam interaksi dengan sistem yang sedang berlangsung
	*Subject Boundary*	*Subject boundary* adalah elemen yang memuat nama sistem yang ditempatkan di dalam atau di bagian atas *boundary* atau bisa juga disebut elemen yang menyatakan ruang lingkup sistem yang dibuat
	Association Relationship	*Association Relationship* merupakan elemen penghubung antara *use case* dan *actor*
	*Include Relationship*	*Include Relationship* digunakan untuk memasukan satu *use case* di dalam *use case* lainnya
	*Extend Relationship*	*Extend Relationship* digunakan untuk

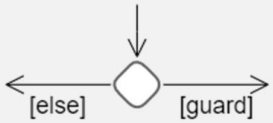
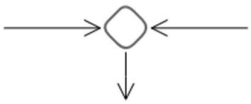
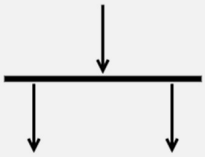
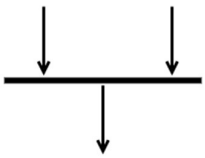
Notasi	Nama	Deskripsi
		memperluas *use case* dengan memasukan perilaku opsional
	Generalization Relationship	*Generalization Relationship* digunakan untuk mewakili *use case* khusus yang merupakan variasi dari *use case* yang lebih umum

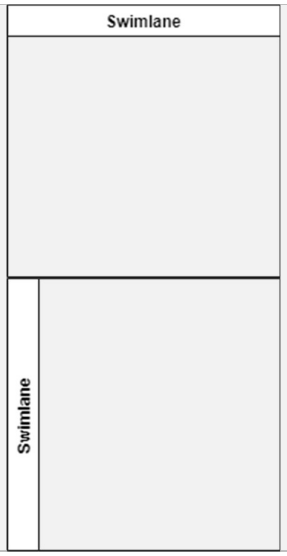
b. *Activity Diagram*

Activity Diagram adalah jenis diagram yang digunakan untuk memvisualisasikan tindakan utama dan hubungan antara tindakan dalam suatu proses [31]. Adapun elemen dari *Activity Diagram* ini meliputi:

Tabel 2.5 Simbol dan penjelasan pada *activity diagram*

Notasi	Nama	Deskripsi
	*Action*	*Action* merupakan sebuah elemen yang memiliki perilaku sederhana dan tidak diuraikan
	*Activity*	*Activity* merupakan elemen yang berfungsi untuk merepresentasikan sekumpulan tindakan (*action*)
	*Object Node*	*Object Node* merupakan elemen yang berfungsi untuk merepresentasikan sebuah objek yang terhubung dengan sekumpulan aliran objek
	Control Flow	*Control Flow* adalah elemen yang berfungsi untuk menunjukan urutan eksekusi
	Object Flow	*Object Flow* merupakan elemen yang berfungsi untuk menunjukan *flow* suatu objek dari satu aktivitas ke aktivitas lainnya.
	*Initial Node*	*Initial Node* merupakan elemen yang merepresentasikan

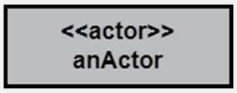
Notasi	Nama	Deskripsi
		kondisi awal dari serangkaian tindakan atau kegiatan
	*Final-activity Node*	*Final-activity Node* merupakan elemen yang berfungsi untuk menghentikan aliran kendali kontrol dan aliran objek dalam suatu sebuah aktivitas
	*Final-flow Node*	*Final-flow Node* merupakan elemen yang berfungsi untuk menghentikan aliran kendali atau aliran objek tertentu dalam suatu konteks
	*Decision Node*	*Decision Node* merupakan elemen yang berfungsi untuk menggambarkan kondisi pengujian dengan tujuan memastikan bahwa aliran objek atau aliran kontrol hanya mengikuti satu jalur saja
	*Merge Node*	*Merge Node* merupakan elemen yang berfungsi untuk menggabungkan kembali berbagai jalur keputusan yang dibentuk menggunakan *Decision Node*
	*Fork Node*	*Fork Node* merupakan elemen yang berfungsi untuk membagi perilaku ke dalam sejumlah aktivitas yang berjalan secara paralel atau bersamaan dari aktivitas utama
	*Join Node*	*Join Node* merupakan elemen yang berfungsi untuk menggabungkan kembali serangkaian aliran aktivitas yang berjalan secara bersamaan atau secara paralel

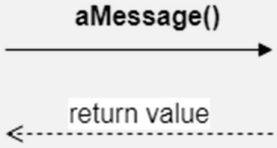
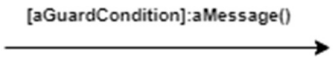
Notasi	Nama	Deskripsi
	*Swimlane*	*Swimlane* merupakan elemen berfungsi untuk membagi diagram aktivitas menjadi baris dan kolom, dengan tujuan menetapkan kegiatan individu kepada individu atau objek yang bertanggung jawab dalam melaksanakan aktivitas tersebut

c. *Sequence Diagram*

Sequence Diagram adalah jenis diagram yang memvisualisasikan pesan antara objek-objek yang terlibat dalam sebuah untuk *use case* atau skenario tertentu dari waktu ke waktu [31]. Adapun elemen dari *Sequence Diagram* ini meliputi:

Tabel 2.6 Simbol dan penjelasan pada *sequence diagram*

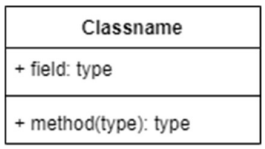
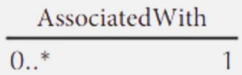
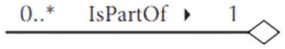
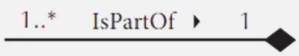
Notasi	Nama	Deskripsi
	*Actor*	*Actor* adalah entitas, baik berupa orang atau sistem yang memperoleh dari dan berada diluar sistem
	*Object*	*Object* adalah elemen yang terlibat dalam urutan tindakan dengan dengan mengirim atau menerima pesan
	*Lifeline*	*Lifeline* adalah elemen yang menunjukan keberadaan atau eksistensi suatu objek selama urutan waktu tertentu

Notasi	Nama	Deskripsi
	*Execution Occurrence*	*Execution Occurrence* atau kejadian eksekusi adalah elemen yang ditempatkan dibagian atas *lifeline* dalam sebuah diagram untuk menunjukan waktu suatu objek dalam mengirim atau menerima pesan dalam urutan waktu yang ditentukan
	*Message*	*Message* adalah elemen yang berfungsi untuk mengirimkan informasi atau pesan dari satu objek ke objek lainnya
	*Guard Condition*	*Guard Condition* merupakan elemen tes yang harus dipenuhi sebelum suatu pesan dapat dikirimkan
	*Object Destruction*	*Object Destruction* merupakan elemen yang ditempatkan di bagian ujung *lifeline* dalam sebuah diagram untuk menunjukan bahwa objek tersebut akan berhenti atau dihancurkan pada titik tertentu
	*Frame*	*Frame* merupakan elemen yang digunakan untuk menunjukan konteks *sequence diagram*

d. Class Diagram

Class Diagram adalah jenis diagram yang menggambarkan kelas-kelas dan hubungan antar kelas yang tidak berubah dalam sistem dari waktu ke waktu [31]. Adapun elemen dari *Class Diagram* ini meliputi:

Notasi	Nama	Deskripsi
	Generalization	*Generalization* adalah suatu bentuk relasi yang terbentuk antara beberapa kelas

Notasi	Nama	Deskripsi
	*Class*	*Class* adalah sebuah komponen yang mewakili tipe entitas, seperti orang, tempat, atau benda, yang diperlukan oleh sistem untuk mengumpulkan data dan menyimpan data atau informasi
	*Association*	*Association* merupakan hubungan antara beberapa kelas atau sebuah kelas dan dirinya sendiri
	*Aggregation*	*Aggregation* adalah elemen yang mewakili bagian logis dari hubungan antara beberapa kelas atau hubungan antara sebuah kelas dengan dirinya sendiri
	*Composition*	*Composition* adalah elemen mewakili bagian fisik dari hubungan antara beberapa kelas atau hubungan antara sebuah kelas dengan dirinya sendiri

2.2.10. Pengujian Aplikasi

Pengujian memiliki peran yang sangat krusial dalam pengembangan aplikasi, terutama dalam memverifikasi bahwa kode yang telah dibuat dapat diukur dan dikelola dengan mudah di masa mendatang. Dalam konteks pengujian, terdapat dua tujuan utama. Pertama, untuk memperoleh informasi mengenai kualitas dari aplikasi tersebut. Kedua, untuk memverifikasi bahwa aplikasi berjalan sesuai dengan kebutuhan yang telah ditentukan [18]. Di platform Android, terdapat *test code* yang digunakan untuk menguji sebuah aplikasi. *Test code* adalah sekumpulan kode untuk melakukan sejumlah proses pengujian untuk memastikannya berfungsi sebagaimana mestinya. Selain melakukan pengujian menggunakan *test code*, pengujian langsung dengan melibatkan ahli dalam bidangnya juga dilakukan untuk memastikan bahwa setiap fungsi dalam aplikasi

berjalan dengan benar. Dalam, penelitian ini, pengujian menggunakan *test code* mencakup pengujian *black box* untuk menguji *User Interface*, sementara pengujian langsung yang dilakukan oleh ahli dalam bidangnya menggunakan *Heuristic Evaluation*. Penjelasan mengenai tahapan pengujian tersebut dapat dilihat dibawah ini.

a. Pengujian UI (UI *Testing*)

Pengujian UI merupakan sebuah pengujian yang masuk dalam kategori pengujian *black box*. Tujuan pengujian ini untuk menguji sesuai dengan kondisi *user* atau pengguna ketika berinteraksi pada sebuah aplikasi. Dalam pengujian UI ini, peneliti menggunakan Espresso. Espresso adalah sebuah *framework* pengujian yang tersedia pada Android *Testing Support Library*. Ada beberapa keuntungan ketika menggunakan Espresso yaitu *framework* ini otomatis melakukan sinkronisasi antara sebuah skenario *test* dengan target aplikasi yang akan diuji. Selain itu, Espresso juga memastikan komponen aplikasi seperti *activity* dijalankan terlebih dahulu sebelum *test*-nya berjalan serta mampu mendeteksi setiap proses *asynchronous* yang berjalan di dalam aplikasi yang diuji [18]. Contoh implementasi dari Espresso, terlihat pada Gambar 2.6

Gambar 2.6 Implementasi *library* Espresso

Pada gambar Gambar 2.6 terdapat tiga komponen utama dari Espresso yaitu:

1) *ViewMatchers(onView(ViewMatcher))* digunakan untuk menemukan elemen atau komponen antarmuka yang diuji.
2) *ViewActions(perform(ViewAction))* digunakan untuk memberikan *event* untuk melakukan sebuah aksi pada komponen antarmuka yang diuji.
3) *ViewAssertions* digunakan untuk melakukan pemeriksaan terhadap sebuah kondisi atau *state* dari komponen yang diuji.

b. Pengujian *Heuristic Evaluation*

Heuristic Evaluation merupakan metode yang digunakan untuk mengidentifikasi dan menemukan masalah dalam desain antarmuka pengguna yang berkaitan dengan kegunaan. Masalah-masalah tersebut dapat diatasi melalui proses desain yang berulang. Dalam *Heuristic Evaluation* terdapat 10 prinsip yang dapat dilihat pada tabel Tabel 2.7 [32].

Tabel 2.7 Prinsip *heuristic evaluation* [32]

No	Prinsip	Keterangan
1	*Visibility of system status*	Pengguna harus mendapatkan umpan balik visual yang jelas tentang apa yang terjadi dalam sistem
2	*Match between system and the real world*	Desain UI harus menggunakan istilah, *icon* dan konvensi yang sudah familiar bagi pengguna agar mudah dipahami
3	*User control and freedom*	Pengguna harus memiliki kemampuan untuk mengendalikan sistem dan dapat membatalkan atau mengubah tindakan yang sudah dilakukan
4	*Consistency and standards*	UI harus konsisten dan mengikuti standar yang berlaku agar pengguna tidak bingung saat berpindah antarmuka
5	*Error prevention*	Sistem harus didesain untuk mencegah kesalahan yang bisa terjadi
6	*Recognition rather than recall*	Desain UI harus menggunakan elemen yang mudah dikenali daripada mengharuskan pengguna mengingat banyak hal
7	*Flexibility and efficient of use*	Sistem harus dapat digunakan dengan efisien oleh pengguna dengan berbagai tingkat keahlian
8	*Aesthetic and minimalist design*	Antarmuka harus menarik secara visual dan tidak mengganggu dengan elemen yang tidak perlu
9	*Help user recognize, dialogue, and recovers from errors*	Sistem harus memberikan pesan yang jelas dan membantu pengguna ketika terjadi kesalahan agar dapat membantu pengguna untuk memperbaikinya
10	*Help and documentation*	Sistem harus menyediakan bantuan yang mudah diakses seperti petunjuk atau panduan pengguna untuk membantu pengguna dalam memahami sistem

Dalam melakukan evaluasi aplikasi menggunakan metode *heuristic evaluation*, kehadiran seorang *evaluator* yang merupakan ahli dalam bidang antarmuka pengguna (*User Interface*) dan pengalaman pengguna (*User*

Experience). Menurut [33], satu *evaluator* memiliki kemampuan untuk mendeteksi masalah *usability* sebesar 35%. Dengan kata lain, semakin banyak *evaluator* yang terlibat, semakin banyak masalah yang dapat diidentifikasi. Namun, secara ideal, jumlah *evaluator* yang optimal adalah antara 3-5 orang. Jika melebihi jumlah tersebut, kemungkinan terjadi penemuan masalah yang serupa secara berulang dan biaya yang dikeluarkan juga akan meningkat [32]. Pada penelitian ini, untuk mengevaluasi tingkat keparahan masalah *usability* yang ditemukan selama analisis, akan digunakan *severity rating* [34]. Menurut [33], terdapat skala *severity rating* yang dapat mengurutkan masalah dari tingkat keparahan terendah hingga tertinggi yang tertera pada Tabel 2.8.

Tabel 2.8 Skala *severity rating* [33]

Nilai	Keterangan
0	***Don't Agree*****:** Sistem atau antarmuka tidak ada masalah
1	***Cosmetic Problem*****:** Terdapat masalah hanya mempengaruhi aspek estetika atau tampilan visual sistem, tanpa mengganggu fungsionalitas atau penggunaan sistem
2	***Minor Usability Problem*****:** Terdapat masalah yang memiliki dampak kecil pada pengalaman pengguna, namun masih dapat diatasi dan tidak menghambat pengguna dalam mencapai tujuan mereka
3	***Major Usability Problem*****:** Terdapat masalah yang memiliki signifikan pada pengalaman pengguna yang menyebabkan ketidaknyamanan atau kesulitan yang cukup berarti dalam penggunaan sistem
4	***Usability Catastrophe*****:** Terdapat masalah yang memiliki dampak serius pada pengalaman pengguna, menghambat pengguna dalam mencapai tujuan mereka atau bahkan menyebabkan kesalahan serius atau kegagalan sistem

2.2.11. *Agile Software Development Methods*

Kent Beck dan 16 orang lainnya dikenal sebagai pencetus ide *Agile Software Development Methods*, yang mereka gambarkan sebagai pendekatan dalam membangun perangkat lunak dengan melakukan dan membantu orang lain dalam proses pembangunan. *Agile Software Development Methods* merujuk pada sekelompok metodologi yang berfokus pada pendekatan iteratif atau berulang, dimana persyaratan dan solusi dikembangkan melalui kolaborasi antar anggota tim [35].

Pengembangan sistem perangkat lunak dapat dilakukan dengan dua metode utama yaitu metode tradisional (*Heavy*) dan metode *agile*. Pengembangan perangkat lunak tradisional memiliki beberapa metode seperti metode *spiral*,

metode *iterative and incremental*, metode *waterfall*, metode *evolutionary* dan lain sebagainya. Untuk pengembangan perangkat lunak *agile* memiliki beberapa metode seperti *Adaptive Software Development* (ASD), *Dynamic System Development Model* (DSDM), *Extreme Programming* (XP) dan Scrum [36]. Berikut merupakan tabel perbandingan pengembangan perangkat lunak tradisional dan pengembangan perangkat lunak *agile*.

Tabel 2.9 Perbandingan pengembangan perangkat lunak tradisional dan *agile*

Parameter	Metode Tradisional	Metode Agile
Kemudahan Modifikasi	Sulit	Mudah
Pendekatan Pembangunan	Prediktif	Adaptif
Orientasi Pembangunan	Berorientasi Proses	Berorientasi Pelanggan
Ukuran Proyek	Besar	Sedang atau Kecil
Skala Perencanaan	Jangka Panjang	Jangka Pendek
Gaya Manajemen	Kontrol dan Perintah	Kerjasama dan Kepemimpinan
Learning	Belajar Terus Menerus sambil Berkembang	Belajar adalah sekunder dari Perkembangan
Dokumentasi	Tinggi	Rendah
Tipe Organisasi	Pendapatan Tinggi	Pendapatan sedang atau rendah
Nomor Organisasi Para Karyawan	Besar	Kecil
Anggaran	Besar	Kecil
Jumlah Tim	Banyak	Satu
Ukuran Tim	Sedang	Kecil

Adapun untuk perbandingan metode pengembangan perangkat lunak *agile* sebagai berikut:

Tabel 2.10 Perbandingan metode pengembangan dalam model *agile*

Karakteristik	***Extreme Programming***	***Dynamic Systems Development Method***	***Adaptive Software Development***	***Scrum***
Pendekatan Pengembangan	Berkala dan Bertahap	Berkala	Berkala	Berkala dan bertahap
Rekomendasi Waktu Iterasi	Satu sampai 6 minggu	80% solusi dalam 20% waktu total	4 hingga 8 minggu	2 hingga 4 minggu
Proyek Tim	Kurang dari 20 orang	Dapat digunakan dalam semua skala tim	5 hingga 9 anggota tim	Dapat digunakan dalam semua skala tim

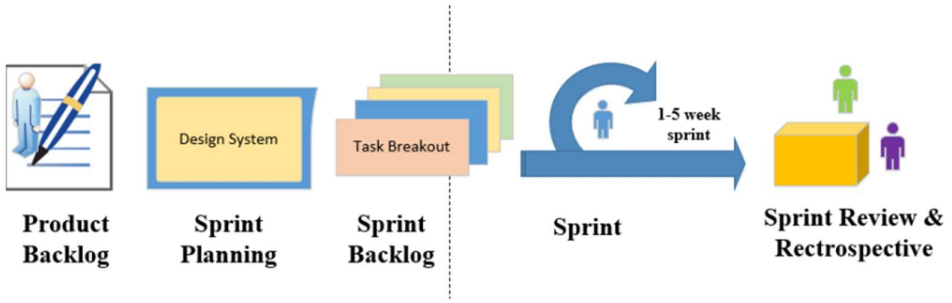
Karakteristik	*Extreme Programming*	*Dynamic Systems Development Method*	*Adaptive Software Development*	*Scrum*
Komunikasi Tim	Informal, pertemuan setiap hari	Berdasarkan dokumentasi	Informal, secara tatap muka	Informal, pertemuan setiap hari
Spesialisasi	*TDD*, *User stories*, *refactoring*	*Prototyping*	*Learning Cycle*	*Sprint*, *Product backlog*, *Sprint backlog*, *Planning*, *Scrum master*
Kelebihan	Ruang kerja terbuka, pelanggan sebagai bagian dari tim, praktik yang terdefinisi dengan baik dan umpan balik	Prioritas kebutuhan, proyek manajemen yang efisien	Pengembangan komponen beresiko tinggi didahulukan	Komunikasi dan kolaborasi yang tinggi
Kekurangan	Dokumentasi yang lemah, disiplin yang kurang, kehadiran pelanggan adalah yang utama	Dokumentasi yang kompleks	Metode dokumentasi yang buruk	Dokumentasi yang lemah, kontrol yang buruk atas proyek

Pada tabel Tabel 2.10, merupakan perbandingan pengembangan metode dalam model *agile*. Metode Scrum cocok digunakan dalam pengembangan perangkat lunak yang dapat diaplikasi pada proyek dengan segala ukuran, ukuran tim dapat menyesuaikan, waktu rekomendasi untuk suatu proyek yaitu 2 hingga 4 minggu dan memiliki komunikasi dan kolaborasi yang tinggi.

2.2.12. *Scrum*

Metode *scrum* merupakan salah satu metode yang termasuk dalam metode *Agile* yang pertama kali diperkenalkan oleh Ken Schwaber pada tahun 1997 [37]. Berdasarkan metode *agile*, *scrum* mampu memberikan nilai dan manfaat yang optimal dalam pengembangan perangkat lunak [38]. Metode ini mengusung konsep yang berulang dalam prosesnya, dengan tujuan untuk mengembangkan produk atau layanan yang inovatif [39].

Di dalam *scrum* terdapat 3 peran utama, yaitu *product owner* (pemilik produk), *scrum master*, dan *development team* (tim pengembang). *Product Owner* (PO) adalah seorang ahli yang memiliki pengetahuan mendalam tentang produk yang sedang dikembangkan. *Development team* adalah tim berisikan individu yang mampu mengorganisir diri mereka sendiri dan berperan sebagai tim multifungsi dalam pengembangan perangkat lunak. *Scrum master* adalah orang yang bertanggung jawab dalam memimpin tim *scrum* ke arah yang benar, membantu mereka mencapai tujuan dan mengatasi hambatan-hambatan yang mungkin menghalangi kinerja mereka dalam menciptakan perangkat lunak yang berkualitas [38].



Gambar 2.7 Proses metode *scrum* [40]

Pada gambar Gambar 2.7 merupakan prinsip proses dalam metode *scrum*. Alur kerja dalam metode *scrum* yang pertama dilakukan adalah seluruh tim menentukan *scrum building block* yang disebut dengan *sprint*. *Sprint* merupakan periode waktu dengan durasi satu hingga empat minggu. *Development team* akan fokus pada pencapaian target dalam setiap *sprint*, dan setiap *sprint* diakhiri dengan *review* yang bertujuan untuk menampilkan hasil yang sudah dibuat dalam sebuah rapat tim. Dalam *scrum* terdapat beberapa fase yaitu [40] :

a. *Product Backlog*

   Product Backlog merupakan pengumpulan daftar kebutuhan yang berisi detail fitur-fitur yang akan dikerjakan berdasarkan persyaratan yang diperoleh dari pengumpulan data.

b. *Sprint Planning*

Sprint Planning merupakan kegiatan perencanaan atau penjadwalan dengan diadakannya rapat oleh *scrum team* untuk mengevaluasi *product backlog*. Perancangan sistem yang akan dikembangkan dilakukan pada tahap ini.

c. *Sprint Backlog*

Sprint Backlog merupakan daftar kebutuhan dari *product backlog* dan akan dikerjakan pada fase *sprint*. Daftar ini sudah terbagi menjadi beberapa bagian untuk kemudian dikerjakan oleh tim.

d. *Sprint*

Sprint merupakan tahapan dimana *developer team* untuk mengembangkan perangkat lunak untuk mencapai kebutuhan sesuai dengan *backlog* dan harus selesai dalam jangan waktu yang telah ditentukan. Selain itu, Pada tahap juga ini akan dilakukan *daily scrum* yaitu aktivitas harian di dalam *sprint* yang dilakukan oleh *scrum team* untuk memeriksa apa yang telah dikerjakan.

e. *Sprint Review* dan *Sprint Retrospective*

Pada tahap ini, akan dilakukan *review* aplikasi untuk melihat kemajuan yang telah dicapai dan menyesuaikan *product backlog* jika dibutuhkan. Selanjutnya, dilakukan *sprint retrospective* untuk mendapatkan umpan balik mengenai kebutuhan fungsional yang telah diulas. Jika ada perubahan, perubahan tersebut akan ditambahkan ke dalam *backlog*. Jika tidak ada perubahan, aplikasi siap untuk diuji.