

REKAYASA PERANGKAT LUNAK PENGOLAHAN KLISE FOTO MENJADI DIGITAL

TUGAS AKHIR
Diajukan untuk Memenuhi Salah Satu Syarat Kelulusan
Program Pendidikan Sarjana

Oleh:
Andreas Fernando Stevanus
2018130001



**JURUSAN INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER-LIKMI
BANDUNG
2021**

REKAYASA PERANGKAT LUNAK PENGOLAHAN KLISE FOTO MENJADI DIGITAL

Oleh:
Andreas Fernando Stevanus
2018130001

Bandung, 19 Agustus 2022
Menyetujui,

Jenisa Felisa, S.T., M.Kom.
Ko-Pembimbing

Dhanny Setiawan, S.T., M.T.
Pembimbing

Dhanny Setiawan, S.T., M.T.
Ketua Jurusan

**JURUSAN INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER-LIKMI
BANDUNG
2021**

ABSTRAK

Dalam masa modern foto bukan lagi hal yang sulit untuk kita simpan. Bahkan saat ini hampir semua orang hanya menyimpan gambar dalam bentuk *soft copy*. Mungkin juga banyak orang yang memiliki klise foto lama dan ingin mengembalikan momen yang ada di dalamnya dalam bentuk *soft copy*. Namun seiring perkembangan teknologi tempat sangat sulit dijumpai tempat yang menerima klise foto lama untuk dijadikan *soft copy*. Hal tersebut menjadi dasar terbentuknya aplikasi ini, dimana pengguna tidak perlu lagi mencari tempat khusus klise foto untuk dijadikan *soft copy*.

Hanya dengan menggunakan ponsel yang telah terinstall aplikasi PENGOLAHAN KLISE FOTO MENJADI DIGITAL ini bisa menjadi solusi ketika dibutuhkan untuk mencetak ataupun mengubah dan menyimpan klise foto menjadi bentuk digital, sehingga lebih mudah untuk dilihat karena disimpan dalam bentuk *soft copy*. Penulisan aplikasi dilakukan dengan menggunakan bahasa pemrograman *java* dan juga menggunakan algoritma *invert* citra negatif pada setiap *pixel*. Pada pembuatan aplikasi ini juga akan diterapkan metode *prototyping* dimana pengguna dapat memiliki gambaran awal mengenai perangkat yang akan dibuat dan pengguna dapat juga menguji aplikasi sebelum nantinya akan dirilis. Aplikasi ini akan berjalan pada *device android 9.0* dan berdasarkan penelitian yang telah dilakukan aplikasi dapat menampilkan gambar berwarna dan dapat melakukan editing sederhana.

KATA PENGANTAR

Puji serta syukur penulis panjatkan kepada Tuhan Yesus Kristus, karena atas berkat serta rahmat-Nya penulis dapat menyelesaikan seluruh rangkaian tugas akhir ini. Dapat menjalani perkuliahan di STMIK LIKMI dengan lancar sebagai mahasiswa jurusan informatika dengan bidang keahlian Rekayasa Perangkat Lunak membuat penulis semakin bersyukur kepada Tuhan. Masa perkuliahan di STMIK LIKMI yang dijalani oleh penulis dengan tekun membuahkan hasil terutama dalam ilmu maupun karakter. Semoga semua ilmu yang sudah penulis dapatkan di kampus ini dapat menjadi bekal yang berguna pada kemudian hari.

Pembuatan tugas akhir ini pun tentunya tidak dapat diselesaikan bila tanpa adanya ketekunan, kerja keras, serta dukungan dari banyak pihak. Terimakasih yang sebesar-besarnya penulis kepada:

1. Bapak Dhanny Setiawan, S.T., M.T. sebagai dosen pembimbing yang terus mendukung dan juga membimbing seluruh tugas akhir yang penulis kerjakan. Seluruh saran dan juga masukan yang menjadi inspirasi penulis dalam menyelesaikan tugas akhir.
2. Ibu Jenisa Felisa, S.T., M.Kom. sebagai ko-pembimbing yang selalu bersedia dalam menjawab semua pertanyaan seputar tugas akhir perkuliahan.
3. Terima kasih juga penulis ucapkan kepada seluruh dosen STMIK LIKMI yang sudah memberikan seluruh ilmu dan juga tenaga dalam mendidik penulis selama menempuh Pendidikan di kampus ini.
4. Tidak lupa juga penulis ucapkan terima kasih kepada keluarga terutama Ayah, Ibu, dan Adik yang selalu memberikan dukungan baik secara moril dan juga materil.
5. Terima kasih juga kepada dr. Irma Permanasari Gani, Sp. KJ yang telah membantu dalam menangani penulis selama berada dalam kondisi kejiwaan yang tidak baik.
6. Maria Putri Fransischa yang selalu memberikan dukungan dan selalu menemani dalam proses pengerjaan tugas akhir ini.
7. Seluruh anggota fotografi analog bogor yang sudah memberikan waktu dan kesempatan untuk belajar lebih dalam mengenai klise foto.

8. Seluruh anggota komunitas fotografi hantu yang sudah menemani dalam proses penyusunan tugas akhir.
9. Seluruh teman dan kerabat spiritual yang sudah memberi dukungan untuk penulis.
10. Stephen Nathanael yang sudah memberikan saran dan juga masukan mengenai penggunaan *Android Studio*.
11. Jimmy Dwiwana yang sudah memberikan info mengenai editing foto dan juga *color correction*.

Tidak terlepas dari segala keterbatasan dan juga kekurangan yang mungkin ada dalam tugas akhir ini, penulis menerima kritik dan juga saran bila ada yang harus diperbaiki ataupun ditambahkan di kemudian waktu. Harapan penulis kiranya tugas akhir ini dapat dimanfaatkan oleh banyak pihak, penulis ucapkan terima kasih, kiranya Tuhan Yesus Kristus serta Bunda Maria memberkati kita semua.

DAFTAR ISI

ABSTRAK	i
KATA PENGANTAR	ii
DAFTAR ISI	iv
DAFTAR TABEL	viii
DAFTAR GAMBAR	x
DAFTAR SIMBOL	xii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	3
1.5 Kegunaan Hasil	3
1.6 Sistematika Penulisan.....	4
BAB II LANDASAN TEORI	5
2.1 Pengertian Rekayasa Perangkat Lunak	5
2.2 Pemrograman Berorientasi Objek.....	6
2.3 Metode Pengembangan <i>Prototyping</i>	7
2.4 Fotografi dan Kamera Analog	8
2.5 Citra Digital.....	9
2.6 Citra Film Negatif	11
2.7 Slider RGB	12
2.8 <i>Brightness</i> dan <i>Contras</i>	13
2.9 Unified Modelling Language (UML)	13
2.10 Android	14
2.11 Android Studio.....	15
2.12 Kotlin	16
2.13 <i>Java</i>	18

BAB III ANALISIS DAN PERANCANGAN PERANGKAT LUNAK.....	20
3.1 Gambaran Umum Perangkat Lunak	20
3.2 Spesifikasi Kebutuhan Perangkat Lunak	21
3.2.1 Kebutuhan Non-Fungsional	21
3.2.2 Kebutuhan Fungsional	21
3.3 Use Case Diagram	22
3.4 Skenario Use Case	22
3.4.1 Skenario Use Case Convert RGB	22
3.4.2 Skenario Use Case Tutorial	24
3.4.3 Skenario Use Case Simpan Gambar	25
3.5 Class Diagram	26
3.6 Activity Diagram	27
3.6.1 Activity Diagram Convert RGB	27
3.6.2 Activity Diagram Tutorial	28
3.6.3 Activity Diagram Simpan Gambar	29
3.7 Sequence Diagram	30
3.7.1 Sequence Diagram Convert Image	30
3.7.2 Sequence Diagram Tutorial	31
3.7.3 Sequence Diagram Simpan Gambar	31
3.8 Rancangan Antarmuka	32
3.8.1 Rancangan Antarmuka Home	32
3.8.2 Rancangan Antarmuka Kamera	33
3.8.3 Rancangan Antarmuka Pick From Gallery	34
3.8.4 Rancangan Antarmuka Crop Gambar	35
3.8.5 Rancangan Antarmuka Ubah Kecerahan	36
3.8.6 Rancangan Antarmuka Edit RGB	37
3.8.7 Rancangan Antarmuka Ubah Kontras	38
3.8.8 Rancangan Antarmuka Save	39
BAB IV IMPLEMENTASI DAN PENGUJIAN PERANGKAT LUNAK	40

4.1	Spesifikasi Kebutuhan Perangkat Keras.....	40
4.2	Pengujian Perangkat Keras	40
4.3	Implementasi Antarmuka	41
4.3.1	Pengujian Antarmuka Home	41
4.3.2	Pengujian Antarmuka Kamera	42
4.3.3	Pengujian Antarmuka <i>View Image</i>	43
4.3.4	Pengujian Antarmuka <i>Pick From Gallery</i>	44
4.3.5	Pengujian Antarmuka <i>Crop Image</i>	45
4.3.6	Pengujian Antarmuka Ubah Kecerahan.....	46
4.3.7	Pengujian Antarmuka Edit RGB.....	47
4.3.8	Pengujian Antarmuka Ubah Kontras.....	48
4.3.9	Pengujian Antarmuka <i>Save</i>	49
4.4	Pengujian Fungsi	50
4.4.1	Pengujian <i>Button Convert Image</i>	50
4.4.2	Pengujian <i>Button Pick From Gallery</i>	51
4.4.3	Pengujian <i>Button Tutorial</i>	52
4.4.4	Pengujian <i>Button Capture</i> Pada Kamera.....	52
4.4.5	Pengujian Fitur <i>Crop</i> Gambar	53
4.4.6	Pengujian Fitur <i>Brightness</i>	54
4.4.7	Pengujian <i>Slider RGB</i>	55
4.4.8	Pengujian Fitur Kontras.....	56
4.4.9	Pengujian <i>Button Save</i>	57
4.5	Pengujian Berdasarkan Kamera dan Histogram	58
BAB V Kesimpulan dan Saran		60
5.1	Kesimpulan	60
5.2	Saran.....	60
DAFTAR PUSTAKA		61

LAMPIRAN	63
----------------	----

DAFTAR TABEL

Tabel 2.1 Tabel Warna 8bit Bagi Setiap Pixel	11
Tabel 2.2 Versi Android.....	15
Tabel 3.1 Skenario Use Case Covert RGB.....	23
Tabel 3.2 Skenario Alternatif I Use Case Convert RGB	23
Tabel 3.3 Skenario Alternatif II Use Case Covert RGB	24
Tabel 3.4 Skenario Alternatif III Use Case Convert RGB	24
Tabel 3.5 Skenario Use Case Tutorial	24
Tabel 3.6 Skenario Use Case Simpan Gambar.....	25
Tabel 3.7 Skenario Alternatif I Use Case Simpan Gambar	25
Tabel 4.1 Spesifikasi Perangkat Keras Ponsel	40
Tabel 4.2 Pengujian Perangkat Keras	40
Tabel 4.3 Skenario Pengujian	50
Tabel 4.4 Hasil Uji Skenario Button Convert Image	50
Tabel 4.5 Skenario Pengujian Button Pick From Gallery.....	51
Tabel 4.6 Hasil Uji Skenario Button Pick From Gallery	51
Tabel 4.7 Skenario Pengujian Button Tutorial	52
Tabel 4.8 Hasil Uji Skenario Button Tutorial	52
Tabel 4.9 Skenario Pengujian Button Capture Pada Kamera	53
Tabel 4.10 Hasil Uji Skenario Button Capture Pada Kamera	53
Tabel 4.11 Skenario Pengujian Fitur Crop Gambar.....	53
Tabel 4.12 Hasil Uji Skenario Crop Gambar	54
Tabel 4.13 Skenario Pengujian Fitur Brightness.....	54
Tabel 4.14 Hasil Uji Skenario Fitur Brightness	54
Tabel 4.15 Skenario Pengujian Slider RGB.....	55
Tabel 4.16 Hasil Uji Skenario Slider RGB.....	55
Tabel 4.17 Skenario Pengujian Fitur Kontras	56
Tabel 4.18 Hasil Uji Skenario Fitur Kontras	56

Tabel 4.19 Skenario Pengujian Button Save	57
Tabel 4.20 Hasil Uji Skenario Button Save	57
Tabel 4.21 Pengujian Berdasarkan Kamera dan Histogram	58

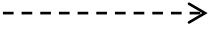
DAFTAR GAMBAR

Gambar 2.1 Citra Warna Grayscale.....	10
Gambar 2.2 Transformasi Citra	12
Gambar 2.3 Stuktur File Android Studio	16
Gambar 3.1 Use Case Color It.....	22
Gambar 3.2 Class Diagram Color It.....	26
Gambar 3.3 Activity Diagram Convert RGB.....	27
Gambar 3.4 Activity Diagram Tutorial	28
Gambar 3.5 Activity Diagram Simpan Gambar.....	29
Gambar 3.6 Sequence Diagram Convert Image	30
Gambar 3.7 Sequence Diagram Tutorial	31
Gambar 3.8 Sequence Diagram Simpan Gambar	31
Gambar 3.9 Rancangan Antarmuka Home.....	32
Gambar 3.10 Rancangan Antarmuka Kamera.....	33
Gambar 3.11 Rancangan Antarmuka Pick From Gallery	34
Gambar 3.12 Rancangan Antarmuka Crop Gambar	35
Gambar 3.13 Rancangan Antarmuka Ubah Kecerahan	36
Gambar 3.14 Rancangan Antarmuka Edit RGB	37
Gambar 3.15 Rancangan Antarmuka Ubah Kontras	38
Gambar 3.16 Rancangan Antarmuka Save	39
Gambar 4.1 Pengujian Antarmuka Home	41
Gambar 4.2 Pengujian Antarmuka Kamera	42
Gambar 4.3 Pengujian Antarmuka View Image.....	43
Gambar 4.4 Pengujian Antarmuka Pick From Gallery	44
Gambar 4.5 Pengujian Antarmuka Crop Image.....	45
Gambar 4.6 Pengujian Antarmuka Ubah Kecerahan	46
Gambar 4.7 Pengujian Antarmuka Edit RGB.....	47
Gambar 4.8 Pengujian Antarmuka Ubah Kontras	48

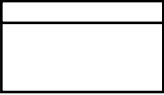
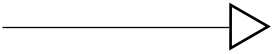
Gambar 4.9 Pengujian Antarmuka Save	49
---	----

DAFTAR SIMBOL

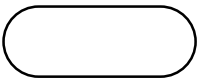
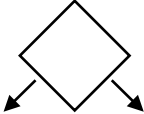
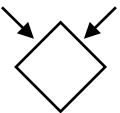
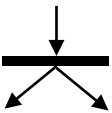
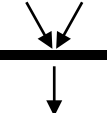
Use Case Diagram

Simbol	Keterangan
<i>Use Case</i> 	Menggambarkan fungsionalitas yang terdapat pada sistem.
<i>Actor/Aktor</i> 	Merupakan abstrak dari orang yang mengoperasikan sistem. Actor juga dibagi sesuai dengan tugasnya masing-masing.
<i>Asosiasi</i> 	Asosiasi merupakan hubungan yang terbentuk antara actor dengan use case. Asosiasi juga menjadi penjelas untuk aksi yang dilakukan oleh aktor.
<i>Include</i> 	<i>Include</i> merupakan hubungan yang terjadi antar use case. Sebagai contoh seorang yang akan login ke suatu aplikasi harus disertai dengan verifikasi OTP.
<i>Extend</i> 	<i>Extend</i> merupakan hubungan yang terjadi antar use case dimana hubungan tersebut boleh ada ataupun tidak. Sebagai contoh seseorang yang sudah melakukan pemesanan barang bisa saja memesan Kembali dan juga bisa tidak.
<i>System Boundary</i> 	<i>System Boundary</i> digunakan sebagai pemisah antara aktor dan use case. Bagian yang berada di dalam <i>boundary</i> merupakan pekerjaan yang dilakukan oleh sistem.

Class Diagram

Simbol	Keterangan
<i>Class</i> 	<i>Class</i> merupakan himpunan dari objek yang terbagi atas atribut dan juga operasi.
<i>Association</i> 	<i>Association</i> menggambarkan relasi yang terjadi antar kelas, bahwa kelas tersebut menjadi referensi kelas lainnya.
<i>Generalization</i> 	<i>Generalization</i> menggambarkan relasi khusus yang terjadi antar kelas.

Activity Diagram

Simbol	Keterangan
<i>Start State</i> 	<i>Start State</i> merupakan simbol yang dipakai ketika memulai suatu aktivitas.
<i>End State</i> 	<i>End State</i> merupakan simbol yang digunakan pada saat berakhirnya suatu aktivitas.
<i>Activity</i> 	<i>Activity</i> adalah aktivitas yang dilakukan dan menggambarkan suatu proses.
<i>Desicion Points</i> 	<i>Desicion Points</i> menggambarkan pilihan benar atau salah (<i>if</i>).
<i>Decision Merge</i> 	<i>Decision Merge</i> merupakan penggabungan dari <i>Decision Points</i> yang terbentuk.
<i>Message</i> 	<i>Message</i> merupakan pesan yang terbentuk sebagai timbal balik antar <i>activity</i> .
<i>Fork</i> 	<i>Fork</i> merupakan kegiatan yang dilaksanakan secara paralel dan pemecah kegiatan.
<i>Join</i> 	<i>Join</i> merupakan titik temu atau penggabungan aktivitas yang sudah dipecah menggunakan <i>fork</i> .

Sequence Diagram

Simbol	Keterangan
<i>Lifeline</i> 	<i>Lifeline</i> menyatakan keberadaan objek dalam satu waktu.
<i>Actor</i> 	Peran <i>actor</i> pada <i>sequence diagram</i> sama seperti perannya pada <i>use case diagram</i> , dimana <i>actor</i> merupakan komunikator antara pengguna dan objek pada sistem.

Simbol	Keterangan
<i>Activation</i> 	<i>Activation</i> merupakan indikasi bahwa suatu objek akan melakukan suatu aksi yang mungkin dilakukan oleh pengguna.
<i>Object</i> 	<i>Object</i> merupakan <i>instance</i> dari sebuah kelas yang kemudian dituliskan secara horizontal.
<i>Message</i> 	<i>Message</i> merupakan anak yang menjelaskan suatu interaksi antar objek.
<i>Reply</i> 	<i>Reply</i> merupakan timbaal balik yang diberikan oleh objek kepada objek yang memberi <i>Message</i> .

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada saat sekarang masyarakat banyak sekali yang menyukai dunia foto. Banyak sekali tujuan diambilnya sebuah foto, baik foto untuk mengabadikan informasi, foto untuk sebuah pengamatan, dan masih banyak lagi. Dengan semakin pesatnya teknologi kini untuk mengambil sebuah foto tidak perlu lagi untuk membawa kamera khusus untuk foto seperti analog kamera ataupun kamera digital.

Dengan hanya menggunakan ponsel pintar saja sudah dapat melakukan pengambilan gambar dengan mudah. Dengan kecanggihan yang sudah ada, kini tidak perlu lagi seperti dahulu dimana seorang fotografer harus memikirkan mengenai pencahayaan yang cukup tidak lebih dan tidak kurang, karena jika cahaya yang terlalu besar masuk dalam optik akan menimbulkan klise roll terbakar atau putih seluruhnya.

Namun tak jarang juga pada zaman sekarang masih ada orang yang menyimpan klise foto di rumah. Klise foto ini sulit untuk disimpan dalam bentuk digital, oleh sebab itu pemilik foto tersebut hanya menyimpannya saja klise foto tersebut. Klise foto tidak seperti foto zaman sekarang sangat mudah untuk dilihat dan sudah berwarna, melihat foto yang diambil dengan menggunakan perangkat digital jauh lebih mudah dibandingkan mencetak foto zaman dahulu dimana kita harus mencetak terlebih dahulu agar bisa menampilkan gambar yang berwarna. Pengambilan foto pada zaman dahulu masih menggunakan kamera analog, dimana kamera analog merupakan kamera yang digunakan untuk pengambilan gambar sebelum adanya kamera digital. Perbedaan antara keduanya adalah jika kamera digital menggunakan memori untuk menyimpan gambar dan dapat langsung dilihat melalui layar yang ada, sedangkan kamera analog menggunakan klise sebagai media penyimpanan gambar dan tidak dapat langsung dilihat begitu saja.

Tempat mencetak foto digital mudah sekali dijumpai tidak seperti kamera analog yang masih menggunakan klise. Survey yang dilakukan oleh *Ilford Photo* pada tahun 2018 lalu menunjukkan bahwa 32,8% masyarakat kembali menggunakan kamera analog setelah

berhenti dan beristirahat dari kamera analog (Ilford Photo, 2019). Kesimpulan pada hasil survei tersebut menunjukkan bahwa masih banyak pengguna kamera analog yang aktif.

Salah satu komunitas yang masih aktif dalam pemotretan dengan kamera analog adalah komunitas dari Analog Bogor, komunitas ini seringkali melakukan perjalanan yang terbilang cukup jauh untuk mengambil gambar dengan kamera analog. Sayangnya perkembangan komunitas ini tidak disertai dengan cukupnya sarana, seperti sudah mahalnya klise foto yang ada dan juga sulitnya melihat hasil foto tersebut dari warna negatif menjadi foto yang berwarna. Dengan dibuatnya aplikasi "**Pengolahan Klise Foto Menjadi Digital**" ini bisa membantu mengembalikan informasi yang tersimpan pada film analog tersebut. Sehingga para pemilik klise foto tidak kesulitan untuk melihat klise foto dimasa sekarang dengan citra analog yang ada, pengguna bisa langsung menjadikan klise foto sebagai file digital dan menampilkan warna.

1.2 Rumusan Masalah

Terdapat beberapa masalah yang terdapat pada penelitian diantaranya, yaitu:

1. Bagaimana foto negatif dapat menampilkan warna dengan mudah?
2. Bagaimana mengolah hasil tangkapan kamera klise negatif?

1.3 Tujuan

Untuk memenuhi syarat kelulusan program sarjana informatika penulis akan membuat program yang berjudul "**Pengolahan Klise Foto Menjadi Digital**". Aplikasi yang penulis buat adalah untuk membantu pemilik klise foto agar pemilik klise foto dapat menjadikan klise foto negatif menjadi berwarna. Selain itu, tujuan lainnya adalah untuk mempermudah pencetakan foto, sehingga pengguna aplikasi tidak perlu lagi mencari tempat khusus yang mencetak klise foto analog. Adanya "**Pengolahan Klise Foto Menjadi Digital**" membantu pengguna agar pencetakan akan lebih mudah dengan hanya menggunakan media-media digital yang ada.

1.4 Batasan Masalah

Aplikasi "**Pengolahan Klise Foto Menjadi Digital**" memiliki beberapa batasan diantaranya:

1. Mampu berjalan pada *android* minimal versi 9.0
2. Klise foto yang digunakan merupakan klise foto yang sudah dicuci.
3. Pengguna dapat memakai kamera untuk melakukan convert atau dengan memilih gambar pada penyimpanan perangkat.
4. Pengguna harus mengikuti cara penggunaan aplikasi dengan menyaksikan tutorial pada menu tutorial.

1.5 Kegunaan Hasil

Kegunaan aplikasi "**Pengolahan Klise Foto Menjadi Digital**" adalah untuk menjadikan sebuah foto yang berada dalam klise foto menjadi sebuah foto yang dapat diakses melalui media digital. Aplikasi ini juga dapat menampilkan gambar yang terdapat pada film analog serta memberi warna pada foto dalam bentuk digital. Sehingga mempermudah pemilik klise foto dalam melakukan pencetakan foto. Setelah melakukan perubahan dari klise foto menjadi digital, pemilik bisa mencetaknya melalui printer yang ada atau dapat mencetaknya di toko pencetakan foto.

Selain kegunaan aplikasi terdapat pula beberapa kegunaan dari hasil karya tulis tugas akhir ini diantaranya :

1. Bagi Penulis

Bagi penulis hasil karya tulis ini bermanfaat untuk menerapkan beberapa materi yang telah dipelajari dan melatih penulis dalam menganalisa permasalahan yang ada secara kritis serta mencari penyelesaiannya.

2. Bagi Pembaca

Bagi pembaca hasil karya ilmiah ini dapat digunakan sebagai bahan bacaan untuk pengetahuan dan untuk menjadi sumber acuan ketika mengkaji pada bidang yang sama.

1.6 Sistematika Penulisan

Untuk Menyusun tugas akhir ini dibutuhkan penyajian sebagai berikut:

1. BAB I PENDAHULUAN

Bab ini akan menjelaskan mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, kegunaan hasil dan juga sistematika dalam penulisan.

2. BAB II LANDASAN TEORI

Bab ini menjelaskan rangkuman mengenai teori-teori mengenai pengertian perangkat lunak, penjelasan mengenai pemrograman berorientasi objek, metode pengembangan prototyping, fotografi dan kamera analog, citra digital, citra film negative, Slider RGB, Unified Modelling Language (UML), penjelasan mengenai android, software android studio, dan Bahasa pemrograman java.

3. BAB III ANALISIS DAN PERANCANGAN PERANGKAT LUNAK

Bab ini menjelaskan mengenai gambaran dari aplikasi pengolahan film analog, uraian mengenai analisis dan perancangan yang digunakan dalam membangun aplikasi ini. Perancangan meliputi *use case diagram*, *activity diagram*, *class diagram*, dan juga antarmuka aplikasi.

4. BAB IV IMPLEMENTASI DAN PENGUJIAN PERANGKAT LUNAK

Bab ini berisikan hasil dari analisis dan juga perancangan yang sudah dibuat dan dapat digunakan. Bab ini juga menjelaskan mengenai pengujian yang akan dilakukan. Pengujian tersebut meliputi pengujian seperti navigasi, fungsi aplikasi dan perangkat yang kompatibel pada android.

5. BAB V KESIMPULAN DAN SARAN

Pada bab ini akan disampaikan kesimpulan yang sudah didapatkan selama proses pembangunan aplikasi ini dan juga akan disampaikan mengenai saran akan aplikasi yang sejenis agar dapat dikembangkan.

BAB II

LANDASAN TEORI

2.1 Pengertian Rekayasa Perangkat Lunak

Pada buku yang berjudul *“Software Engineering A Practitioner’s Approach, Seventh Edition”* karya Roger S. Pressman menyatakan bahwa:

“Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.” (Pressman, 2011)

Pada buku tersebut IEEE (*Institute of Electrical and Electronics Engineer*) juga turut memaparkan bahwa:

“Software Engineering : (1) The application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is the application of engineering to software. (2) The study of approaches as in (1).” (Pressman, 2011)

Menurut Frank F. Tsui dalam buku yang berjudul *“Essentials of Software Engineering”* dituliskan bahwa

“Software Engineering is engineer discipline whose focus in the cost effective development of high quality software system” (Tsui, Karam, & Bernal, 2016)

Dari kutipan di atas dapat dikatakan bahwa rekayasa perangkat lunak merupakan suatu pekerjaan yang berfokus untuk melakukan pengembangan terhadap perangkat lunak yang berkualitas dan menghemat biaya.

Berdasarkan ketiga penjelasan tersebut dapat disimpulkan bahwa rekayasa perangkat lunak merupakan suatu bidang keahlian dengan menggunakan pendekatan sistematis untuk menciptakan, melakukan perawatan serta menata pengembangan perangkat-perangkat lunak.

2.2 Pemrograman Berorientasi Objek

Data berorientasi objek juga dijelaskan oleh Matt Weisfeld melalui bukunya yang berjudul *"The Object-Oriented Thought Process"*, bahwa:

"Data stored within an object represents the state of object. In Object Oriented Programming technology, this data is called attributes." (Weisfeld, 2013)

Data yang tersimpan di dalam objek tersebut disebut juga sebagai atribut, di mana atribut tersebut merepresentasikan sebuah keadaan dari objek tersebut.

Menurut (Raharjo, Budi, Heryanto, & Haryono, 2012) pada buku yang berjudul *"Mudah Belajar Java"* dijelaskan bahwa:

"pemrograman berorientasi objek sendiri merupakan sebuah inti dari pemrograman java di mana semua program pada java merupakan sebuah objek."

Menurut artikel (Beal, 2021) pada website *"Webopedia"* dijelaskan bahwa:

Object-Oriented Programming refer to a type of computer programming in which programmers define the data type of a data structure and also the types of operations that can be applied to the data structure. The data structure is becomes an object that includes both data and functions. In addition, programmers can create relationships between one object to the another objects. The example is objects can inherit characteristics from other objects.

Arti dari kutipan di atas adalah *programmer* akan menentukan tipe data dari struktur data dan juga menentukan fungsi yang dapat diterapkan pada struktur data. Cara ini dilakukan dan menjadikan struktur data sebagai objek yang memiliki fungsi sehingga *programmer* dapat membuat hubungan antara satu objek dengan objek yang lainnya sebagai contoh objek pada sebuah program dapat mewarisi karakteristik dari objek yang lainnya.

Pemrograman berorientasi objek pada dasarnya merupakan pemodelan dan juga interaksi dari objek tersebut, yang akhirnya menghasilkan suatu sistem berdasarkan objek yang ada dan juga dari interaksi antar objeknya.

2.3 Metode Pengembangan *Prototyping*

Metode pengembangan perangkat lunak memiliki beberapa macam, beberapa di antaranya adalah metode *prototyping*, metode *spiral*, *waterfall*, *agile* dan lain sebagainya. Pada kasus ini penulis akan menggunakan metode pengembangan *prototyping*. Menurut Dicoding Indonesia (Setiawan, 2021) dijelaskan bahwa metode *prototyping* merupakan metode yang memungkinkan untuk pengguna memiliki gambaran awal tentang perangkat lunak yang akan dikembangkan lalu pengguna sendiri dapat menguji pada awal aplikasi sebelum nantinya akan dirilis. Metode ini juga digunakan untuk mengembangkan model menjadi sebuah perangkat lunak yang final. Pada metode ini terdapat beberapa tahapan yang harus dilakukan dalam pengembangannya.

1. Analisa Kebutuhan.

Analisa kebutuhan merupakan tahap awal yang dilakukan untuk menentukan hal apa saja yang harus ada dan diperlukan oleh aplikasi yang akan dibuat.

2. *Design*.

Setelah semua kebutuhan ditentukan tahap selanjutnya adalah menentukan *design* sederhana berdasarkan kebutuhan yang telah ditentukan sebelumnya.

3. Pembuatan *Prototype*.

Setelah *design* terbentuk maka selanjutnya *prototype* dari aplikasi mulai untuk dibuat.

4. Evaluasi *Prototype*.

Setelah *prototype* selesai hasil dari *prototype* akan dipresentasikan kepada klien untuk dapat dikomentari apakah aplikasi sudah sesuai atau belum.

5. Perbaikan *Prototype*.

Jika terdapat revisi yang diberikan oleh klien maka *prototype* akan diperbaiki dan dipresentasikan ulang kepada klien. Pada tahap 4 dan 5 bisa saja terjadi perulangan sampai *prototype* sesuai dan tidak ada revisi.

6. Penggunaan Sistem.

Setelah tahap 4 dan 5 selesai maka hasil dari *prototype* sudah dapat dirilis dan siap dipakai.

2.4 Fotografi dan Kamera Analog

Menurut buku “Fotografi” milik (Sudarma, 2014) dijelaskan bahwa:

Media foto sendiri merupakan suatu media komunikasi, yaitu sebuah media yang dapat dipakai untuk menyampaikan sebuah pesan kepada masyarakat luas. Selain itu foto juga dapat diartikan sebagai media untuk mendokumentasikan suatu kejadian atau peristiwa yang tidak akan bisa terlupakan. Seperti contoh momen bahagia bersama keluarga maupun momen duka yang juga sering kali didokumentasikan untuk bisa dilihat kembali.

Lalu pada buku yang berjudul “Jurnal Foto Suatu Pengantar” milik (Gani & Kusumalestari, 2013) dijelaskan bahwa:

Pengertian dari fotografi sebagai teknik merupakan cara untuk dapat mengambil sebuah objek dengan benar. Cara tersebut termasuk dalam teknik pencahayaan, pengolahan gambar, dan semua yang berkaitan dengan fotografi. Fotografi juga merupakan suatu karya seni di mana di dalamnya terkandung nilai estetika yang mencerminkan isi pikiran dan juga pesan yang akan disampaikan oleh fotografer melalui fotonya.

Kesimpulannya adalah bahwa foto merupakan suatu karya seni di mana di dalamnya terdapat berbagai macam unsur dan juga teknik. Selain merupakan suatu karya seni foto sendiri bisa menjadi media sebagai media informasi, foto juga bisa sebagai dokumentasi untuk momen suka maupun duka dalam kehidupan yang nantinya dapat kita lihat dan kita kenang.

Kamera analog sendiri merupakan kamera yang banyak digunakan pada dunia fotografi jauh sebelum adanya kamera digital. Fotografi sendiri berasal dari kata “*photo*” dan “*graph*”. Menurut kamus *Oxford* “*photo*” sendiri berarti sebuah gambar yang dibuat dari sebuah pengambilan gambar dengan kamera baik secara digital dengan memori maupun dengan klise di dalam kamera. Sedangkan “*graph*” berarti sebuah *diagram* yang terdiri atas garis dan menunjukkan dua atau lebih angka di dalamnya. (Darmawan & Wikayanto, 2018)

Fotografi analog sendiri berarti sebuah proses pengambilan citra dengan cara merekam suatu momen dan memanfaatkan proses kimiawi tertentu dalam memanfaatkan *roll* klise. Film sendiri suatu lapisan yang terbuat dari plastik dan sangat peka terhadap cahaya. Klise sendiri dapat dimanfaatkan sebagai media untuk pengambilan gambar

maupun pengambilan video sebelum adanya kamera digital. (Darmawan & Wikayanto, 2018)

Terdapat beberapa teknik pengambilan gambar pada citra negatif untuk bisa diolah menjadi digital salah satunya yaitu dengan cara sebagai berikut :

1. Pertama tama perlu disiapkan satu buah kotak dengan penutup pada bagian atasnya.
2. Dalam kotak tersebut sebisa mungkin ditutupi karton berwarna hitam untuk mengurangi pantulan cahaya yang terjadi.
3. Selain memberi karton hitam, pada bagian bawah kotak diberikan cahaya berwarna putih yang mengarah ke bagian atas kotak.
4. Bagian penutup kotak diberikan lubang berbentuk persegi Panjang dengan ukuran tinggi 3.7 dan lebar 2.5.
5. Saat kotak sudah selesai dibuat, klise foto negatif dapat langsung disimpan padaa bagian tutup kotaak yang diberi lubang (disesuaikan posisi klise dan lubangnya).
6. Jika sudah sesuai, gambar pada klise akan terlihat dan bisa langsung di foto dengan menggunakan kamera ponsel.

2.5 Citra Digital

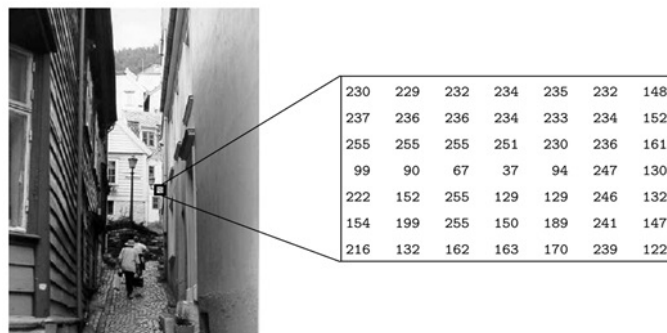
Citra digital sendiri merupakan suatu citra (gambar) yang dapat diolah baik dari segi warna, bentuk dan sebagainya. Pada sebuah gambar *greyscale* dengan ukuran 150x150 *pixel* di mana pada setiap gambar sebenarnya memiliki kotak kecil berukuran sebesar 5x7 *pixel*, kotak kecil tersebut menyimpan banyak sekali warna di dalamnya namun yang terjadi pada komputer *pixel* warna tersebut disimpan dalam bentuk angka seperti pada gambar 2.1 setiap angka mewakili intensitas warna yang ada pada gambar tersebut (Saputra, Pranata, & Handani, 2016). Citra digital berdasarkan *pixel* terbagi atas 5 jenis di antaranya yaitu:

1. Citra Biner

Citra Biner merupakan gambar yang hanya memiliki dua kemungkinan nilai pada setiap *pixel*. Nilai untuk setiap *pixel* bisa berwarna hitam atau putih, citra biner banyak sekali ditemukan pada zaman dahulu. Pada saat ini citra biner lebih sering dikenal dengan nama *B&W color (black and white)* atau *monochrome color*. (Saputra, Pranata, & Handani, 2016)

2. Citra Grayscale

Citra *greyscale* adalah citra yang hanya memiliki satu nilai pada setiap *pixel* dengan kata lain warna dasar RGB memiliki nilai yang sama. Nilai tersebut digunakan untuk menentukan intensitas warna, warna yang ada pada *greyscale* yaitu dari hitam hingga menuju putih. Kedalaman warna pada *greyscale* adalah sebesar 8bit di mana pada 8bit hanya terdapat kombinasi warna sebanyak 256 warna ke abuan. (Saputra, Pranata, & Handani, 2016)



Gambar 2.1 Citra Warna *Grayscale*
(Saputra, Pranata, & Handani, 2016)

3. Citra Warna (8bit)

Berbeda dengan citra *greyscale* yang memiliki kedalaman sebesar 8bit dengan kombinasi warna ke abuan saja, citra warna 8bit memiliki kombinasi warna *red-green-blue* (RGB), di mana terdapat 256 kombinasi warna RGB di dalamnya. Pada warna 8bit ada kombinasi warna yang menggunakan *palette* warna di mana terdapat kombinasi warna dan *palet* warna ini termasuk yang paling sering dipakai. Namun selain dari *palette* warna ada juga yang menggunakan format 8bit untuk setiap *pixel* gambar. (Saputra, Pranata, & Handani, 2016)

Tabel 2.1 Tabel Warna 8bit Bagi Setiap *Pixel*

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	G	G	G	B	B

(Saputra, Pranata, & Handani, 2016)

4. Citra Warna (16bit)

Pada citra warna 16bit terdapat kombinasi warna sebanyak 65.536. Pada citra warna 16bit setiap *pixel* gambar akan diwakili oleh 2 *byte memory*. Untuk masing-masing warna pada komponen merah dan juga biru memiliki jumlah sebesar 10bit warna, di mana pada setiap warna memiliki nilai sebesar 5bit. Namun berbeda dengan warna hijau yang pada citra warna 16 bit memiliki nilai sebesar 6bit, hal ini merupakan dasar dari terbentuknya 16bit di mana kombinasi warna merah dan biru bernilai 10bit dijumlahkan dengan warna hijau yang bernilai 6bit hingga terbentuklah citra warna 16bit. Pada citra warna 16bit warna hijau memiliki nilai lebih besar yaitu sebesar 6bit, hal ini terjadi karena manusia pada dasarnya lebih sensitif terhadap warna hijau. (Saputra, Pranata, & Handani, 2016)

5. Citra Warna (24bit)

Citra warna 24bit memiliki setidaknya 16.777.216 warna di dalamnya. Citra warna 24bit mewakili setiap *pixel* di dalamnya, citra warna 24bit ini sudah sangat cukup untuk memvisualisasikan setiap warna yang ada dan mampu dilihat oleh manusia. Manusia pada dasarnya hanya mampu untuk membedakan 10 juta warna saja. Untuk citra warna 24bit dibagi menjadi merah, biru dan hijau yang masing-masingnya memiliki nilai sebesar 8bit warna.

2.6 Citra Film Negatif

Citra negatif atau *invert image* merupakan citra yang berkebalikan dengan citra aslinya, sama halnya dengan klise foto lama di mana hasil pengambilan gambar disimpan dalam citra negatif. Jika terdapat sebuah citra yang mempunyai jumlah *gray level* (L) dengan jangkauan/*range* sebesar (0 hingga L-1), maka citra negatif didapatkan dari transformasi negatif menggunakan persamaan (Saputra, Pranata, & Handani, 2016):

$$s = L - 1 - r$$

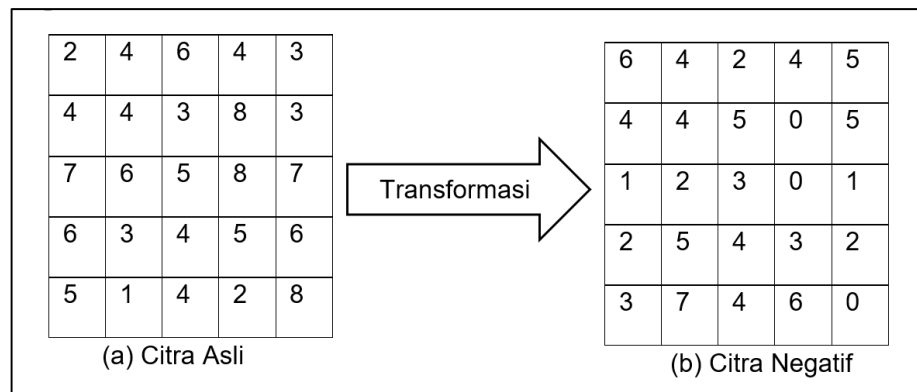
Keterangan :

s = citra dari transformasi negatif

L = banyaknya *gray level* pada citra digital

r = citra asli

Contoh : jika L = 9, maka :



Gambar 2.2 Transformasi Citra

(Saputra, Pranata, & Handani, 2016)

2.7 Slider RGB

RGB adalah singkatan bagi warna *red*, *green*, dan *blue* sebagai warna dasar pada suatu citra digital. Pada Slider RGB setiap slider akan mewakili setiap warna baik itu merah, hijau dan juga biru. Slider RGB digunakan oleh pengguna sebagai perantara untuk menyesuaikan warna primer yang diinginkan oleh pengguna dengan cara menggeser kontrol pada slider yang warnanya akan ditingkatkan ataupun dikurangi. Biasanya penggunaan Slider RGB sendiri lebih sering dipakai oleh orang yang memiliki ide-ide kreatif dan ingin mencoba sesuatu yang baru pada citra digital yang ada. Slider RGB juga bisa menjadi kontrol untuk keseimbangan warna citra digital yang akan diolah (Mcgregor, 2017). Rumus dalam perubahan warna RGB adalah sebagai berikut :

$$\text{Output} = \text{Input} + R$$

Output = Hasil akhir perubahan warna dengan maksimal output 225

Input = Besar warna yang akan diterapkan pada gambar

R = Warna asli pada citra

2.8 *Brightness* dan *Contras*

Kecerahan atau *Brightness* merupakan kata lain yang digunakan untuk menghitung intensitas cahaya. Pada sebuah citra *brightness* yang terdapat pada (*pixel*) gambar bukan merupakan *brightness* yang sesungguhnya, karena *brightness* yang terdapat pada citra merupakan rata-rata *brightness* yang didapatkan dari sekeliling ruangan yang melingkupinya. Pada *brightness* level tertinggi berada pada tingkat 10^{10} . Rumus untuk menghitung *brightness* dan juga kontras adalah sebagai berikut :

$$I_N = (I - I_{min}) \frac{225}{I_{max} - I_{min}}$$

I_n = intensitas pixel yang baru

I_{min} = intensitas terkecil sebelum diperoleh hasil yang baru

I_{max} = Intensitas terbesar pada citra

2.9 Unified Modelling Language (UML)

Unified Modelling Language (UML) adalah suatu bahasa yang digunakan untuk mendefinisikan secara visual, selain itu (*UML*) juga dipakai untuk menentukan sebuah spesifikasi, rancangan, model, dan juga sebagai dokumentasi aspek yang dibutuhkan dari sebuah sistem. (Herpendi, 2016) Pada (*UML*) terdapat beberapa bagian atau yang biasa disebut dengan *diagram* antara lain:

1. *Use Case Diagram*

Use Case Diagram merupakan sebuah pemodelan yang digunakan mendeskripsikan interaksi yang ada antara aktor dengan sistem. Pada *Use Case Diagram* terdapat banyak sekali simbol yang mewakili setiap aksi maupun pengguna, pada *diagram* ini juga terdapat garis relasi yang saling terhubung sesuai dengan ketentuannya. Berikut simbol yang biasanya terdapat pada *Use Case Diagram*. (Herpendi, 2016)

2. *Class Diagram*

Class Diagram merupakan gambaran mengenai hubungan antar kelas yang ada pada sebuah sistem. Pada *diagram* kelas ini terdapat aturan dan juga tanggung jawab yang harus bisa dilaksanakan oleh sistem. Selain itu kelas *diagram* juga menggambarkan atribut

dan juga operasi yang terdapat pada suatu sistem. Adapun hubungan antar kelas dilambangkan menggunakan angka. (Urva & Siregar, 2015)

3. *Sequence Diagram*

Sequence Diagram merupakan suatu gambaran suatu objek pada *Use Case*, dengan cara mendeskripsikan aksi pada saat objek mulai berjalan, mengirim pesan dan menerima umpan balik antar objek yang ada. Pada *Sequence Diagram* terdapat beberapa simbol yang digunakan untuk mendeskripsikan aksi pada objek antara lain sebagai berikut. (Urva & Siregar, 2015)

4. *Activity Diagram*

Activity Diagram merupakan aliran dari suatu aktivitas menuju ke aktivitas lainnya pada suatu sistem. *Activity Diagram* merupakan *diagram* yang penting saat kita memiliki suatu sistem, karena *diagram* ini memodelkan fungsi yang ada pada suatu sistem. Pada *diagram* ini juga terdapat beberapa simbol yang dipakai seperti yang terdapat pada tabel di bawah ini. (Muslihudin & Oktafiano, 2016)

2.10 Android

Android atau lebih tepatnya *Android Incorporated* adalah perusahaan yang awalnya membuat dan memperkenalkan *android* sebelum akhirnya *android* dibeli dan tergabung Bersama perusahaan *Google Incorporated* pada tahun 2005. *Android Incorporated* didirikan oleh Andy Rubin pada bulan Oktober 2003 di Palo Alto, California, Amerika Serikat. Pada awalnya perusahaan *Android Incorporated* mengembangkan hanya untuk kepentingan kamera saja, namun karena pasar yang terlalu sempit akhirnya *Android Incorporated* mulai masuk ke dunia ponsel pintar yang kita kenal hingga saat ini. Kata *android* sendiri awalnya terbentuk dari panggilan yang diberikan oleh rekan kerja Andi Rubin karena kecintaan Andi Rubin pada robot. Lalu pada tahun 2008 HTC meluncurkan ponsel pertama yang menggunakan *android* setelah sebelumnya pada tahun 2007 *Google Incorporated* mengumumkan pengembangan sistem operasi *android*. Dan hingga saat ini *android* masih terus berkembang dan memiliki versi lebih dari 10 di antaranya adalah seperti pada tabel di bawah. (Javapoint, 2021)

Tabel 2.2 Versi Android
(Javapoint, 2021)

Nama	Versi	API	Tanggal Rilis
No codename	1.0	1	September 23, 2008
No codename	1.1	2	February 9, 2009
Cupcake	1.5	3	April 27, 2009
Donut	1.6	4	September 15, 2009
Eclair	2.0 - 2.1	5 - 7	October 26, 2009
Froyo	2.2 - 2.2.3	8	May 20, 2010
Gingerbread	2.3 - 2.3.7	9 - 10	December 6, 2010
Honeycomb	3.0 - 3.2.6	11 - 13	February 22, 2011
Ice Cream Sandwich	4.0 - 4.0.4	14 - 15	October 18, 2011
Jelly Bean	4.1 - 4.3.1	16 - 18	July 9, 2012
KitKat	4.4 - 4.4.4	19 - 20	October 31, 2013
Lollipop	5.0 - 5.1.1	21- 22	November 12, 2014
Marshmallow	6.0 - 6.0.1	23	October 5, 2015
Nougat	7.0	24	August 22, 2016
Nougat	7.1.0 - 7.1.2	25	October 4, 2016
Oreo	8.0	26	August 21, 2017
Oreo	8.1	27	December 5, 2017
Pie	9.0	28	August 6, 2018
Android 10	10.0	29	September 3, 2019
Android 11	11	30	September 8, 2020

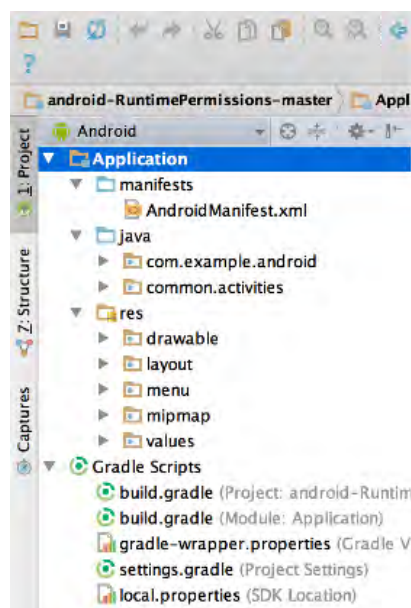
2.11 Android Studio

Android Studio adalah *Integrated Development Environment (IDE)* merupakan *software* di mana seseorang dapat membuat aplikasi sendiri. Selain sebagai alat untuk mengedit baris kode *android studio* juga memiliki banyak fitur di dalamnya, fitur tersebut berguna untuk meningkatkan produktivitas para *developer*. Bahasa yang digunakan oleh *android studio* sendiri terbilang umum yakni *Java* dan *Kotlin*. Sebagai editor baris kode yang dikembangkan oleh *IntelliJ IDE*, kini *android studio* menggantikan editor kode pertamanya yakni *Eclipse Android Development Tools (Eclipse ADT)*. Berikut merupakan beberapa fitur yang dimiliki oleh *android studio*. (Android Developer, 2021)

1. Memiliki *open source build system* yang lebih fleksibel.
2. Memiliki emulator yang mudah untuk digunakan.
3. Dapat digunakan bagi berbagai jenis android.
4. Lebih mudah saat mencoba menjalankan aplikasi karena tidak perlu dalam bentuk APK terlebih dahulu.

5. Lebih mudah dalam membuat *repository* karena dapat terhubung langsung dengan *GitHub*.
6. Alat dan juga kerangka pengujian yang luas.
7. Dapat mengetahui kinerja, kompatibilitas versi, dan juga masalah lainnya dengan menggunakan *lint tools*.
8. Integrasi *Google Cloud Messaging* dan *App Engine* yang menjadi dukungan bawaan dalam *Google Cloud Platform*.

Di dalam *android studio* sendiri terdapat modul untuk setiap proyeknya, seperti modul aplikasi, modul *library*, dan juga modul *Google Cloud* yang isinya merupakan *file* kode yang telah kita buat dan juga *file* yang terbentuk secara otomatis oleh *android studio* (Android Developer, 2021). Secara *default* *android studio* menampilkan *file* yang merupakan proyek yang telah dibuat. Semua *file* akan terlihat seperti pada gambar di bawah di mana seluruh *file* terbagi dalam masing masing fungsi secara *default*.



Gambar 2.3
Struktur *File Android Studio*
(Android Developer, 2021)

2.12 Kotlin

Kotlin sendiri merupakan salah satu bahasa pemrograman yang diketik dengan statis yang menargetkan *Java Virtual Machine (JVM)*, *Native*, *JavaScript*, dan juga *Android*.

Bahasa pemrograman Kotlin sendiri dikembangkan oleh JetBrains di mana proses proyek Kotlin sendiri dimulai pada tahun 2010. Kotlin resmi merilis Kotlin versi 1,0 pada bulan Februari 2016 di mana Kotlin merupakan Bahasa pemrograman *open source* (Ramadoni, 2017). Kotlin sendiri menawarkan beberapa fitur dalam mengembangkan aplikasi *Android*. Berikut ini merupakan fitur yang ada pada Kotlin:

1. Kompatibilitas

Kotlin sendiri sepenuhnya kompatibel dengan *Java Development Kit (JDK) 6*, sehingga dapat berjalan walau pada perangkat *Android* versi lama. Kotlin juga dapat digunakan pada *Android Studio* dan kompatibel dengan *Android Build System*.

2. Peforma

Aplikasi yang dibangun dengan Bahasa Kotlin dapat memiliki peforma yang mirip *Java*, hal ini disebabkan karena Kotlin memiliki *bytecode* yang mirip dengan *Java*. Meskipun Bahasa Kotlin memiliki peforma yang mirip dengan *Java*, bila dilakukan perbandingan seringkali program yang dibuat dengan Kotlin berjalan lebih cepat dibandingkan yang dibuat dengan Bahasa *Java*, ini disebabkan karena Kotlin memiliki dukungan fungsi *inline*.

3. Interoperabilitas

Kotlin sepenuhnya dapat beroperasi dengan *Java*, sehingga memungkinkan *developer* untuk menggunakan seluruh *Android library* yang ada dalam aplikasi, termasuk pengolahan anotasi, sehingga *databinding* dan *dagger* dapat berjalan.

4. *Footprint*

Kotlin sendiri memiliki *runtime library* yang sangat padat, sehingga kedepannya dapat dikurangi melalui penggunaan *ProGuard*. Pada aplikasi sesungguhnya, *runtime* yang terdapat pada Kotlin hanya menambahkan beberapa metode dan ukuran apk kurang dari 100Kb

5. Waktu Kompilasi

Kotlin juga mendukung kompilasi inkramental yang sangat efisien, sehingga ada beberapa tambahan *overhead* untuk *clean build*, pada tahap ini biasanya Kotlin cenderung lebih cepat dibandingkan dengan *Java*.

6. *Learning Curve*

Developer yang lebih dulu menggunakan *Java* akan cenderung lebih cepat menguasai Bahasa Kotlin. *Automated Java to Kotlin Converter* yang terdapat pada *plugin* Kotlin akan sangat membantu *developer Java*.

2.13 *Java*

Java merupakan salah satu bahasa pemrograman yang sering dipakai oleh *developer* khususnya dalam bidang pembuatan aplikasi *android*. Salah satu faktor yang menyebabkan *java* begitu populer karena *java* bisa dijalankan di berbagai macam sistem operasi termasuk juga telepon genggam. Bahasa pemrograman *java* pertama kali diciptakan oleh James Gosling saat masih berada di *Sun Microsystem Incorporated* yang merupakan salah satu produsen semikonduktor dan juga perangkat lunak di *Santa Clara, California* yang pada saat 2010 yang kemudian dibeli oleh *Oracle Corporation*. *Java* sendiri merupakan pengembangan dari Bahasa lainnya yaitu Bahasa C++, oleh karena Bahasa *java* mengadopsi banyak Bahasa C dan juga C++ saat ini *java* menjadi sangat luas penggunaannya mulai dari pengembangan aplikasi *mobile* maupun *web*. *Java* merupakan Bahasa pemrograman *multiplatform* yang sifatnya mengadopsi sistem berorientasi objek di mana *java* juga memiliki banyak *library* di dalamnya. (Nofriadi, 2015)

Pada Bahasa pemrograman *java* terdapat *Java Virtual Machine* yang merupakan komponen penting agar pemrograman yang dibuat dengan *java* dapat berjalan pada sebuah sistem operasi. Pada *java virtual machine* terdapat juga dua pembagian di dalamnya. Pembagian tersebut adalah *Java Runtime Environment (JRE)* dan *Java Development Kit (JDK)*. *Java Runtime Environment* digunakan untuk menjalankan aplikasi *Windows* yang dibuat menggunakan *java*. Jika kita tidak memasang *JRE* maka sudah pasti aplikasi *Windows* yang kita buat tidak akan bisa berjalan dengan semestinya. (Nofriadi, 2015)

Berbeda dengan *JRE*, *Java Development Kit (JDK)* merupakan sekumpulan *tools* yang bisa kita gunakan dalam membuat aplikasi menggunakan *java*. Di dalam *JDK* sendiri terdapat beberapa komponen utama di dalamnya, yaitu: (Nofriadi, 2015)

1. *Compiler*

Digunakan untuk menerjemahkan Bahasa *java* yang telah kita buat biasanya ekstensi *file* akan berbentuk *.class*. (Nofriadi, 2015)

2. *Interpreter*

Interpreter berfungsi untuk menjalankan setiap baris perintah *non-grafis*. (Nofriadi, 2015)

3. *Applet*

Berfungsi untuk menguji *java applet* secara maksimal. *Applet* merupakan sebuah program kecil yang ditulis menggunakan Bahasa *java*. (Nofriadi, 2015)

4. *Debugger*

Berfungsi untuk mendeteksi masalah pada program ketika program dijalankan. (Nofriadi, 2015)

5. *Class File Disassembler*

Berfungsi untuk menguji setiap kelas, apakah kelas yang ada dapat berjalan dengan semestinya atau tidak. (Nofriadi, 2015)

6. *Header and Subfile Generator*

Untuk menghasilkan kode sumber sebagai implementasi *method*. (Nofriadi, 2015)

7. *Java Documentation Generating*

Berfungsi untuk menghasilkan dokumentasi langsung dari kode sumber *java*. (Nofriadi, 2015)

Setelah *Java Virtual Machine (JVM)* terdapat satu komponen *java* yang lainnya. Komponen tersebut bernama *Integrated Development Environment (IDE)*. *IDE* merupakan sebuah teks editor untuk mengetikkan sebuah skrip pemrograman *java*. Teks editor tersebut biasanya seperti *notepad*, *Jcreator*, *Netbeans*, *IntelliJ IDEA*, dan masih banyak lagi. (Nofriadi, 2015)

BAB III

ANALISIS DAN PERANCANGAN PERANGKAT LUNAK

3.1 Gambaran Umum Perangkat Lunak

Color It merupakan nama yang penulis pakai untuk proyek perangkat lunak ini. *Color It* merupakan aplikasi pengolahan gambar yang digunakan untuk mengubah warna negatif yang terdapat pada klise foto analog menjadi warna RGB. Aplikasi ini hanya dapat mengubah klise yang sudah dicuci dan berbentuk transparan. Jika dibandingkan dengan kondisi saat ini, di mana era sudah serba digital *Color It* bisa menjadi salah satu solusi untuk menampilkan foto menjadi bentuk digital sehingga pengguna bisa melihat foto tersebut dengan warna dan foto tersebut dapat disimpan dalam bentuk digital pada ponsel.

Aplikasi ini digunakan dengan cara melakukan pengambilan gambar pada klise foto menggunakan kamera ponsel maupun dengan melakukan *import* gambar dari *gallery* pengguna. Pengambilan gambar tetap harus dengan mengikuti prosedur yang sudah dijelaskan pada halaman tutorial. Prosedur yang terdapat pada tutorial digunakan untuk memaksimalkan pengambilan gambar dan memaksimalkan hasil yang akan tampil sesudah proses konversi. Prosedur yang harus diikuti tersebut meliputi membuat kotak pencahayaan, meminimalkan cahaya dari luar dan memberikan cahaya pada bagian bawah klise sebelum melakukan pengambilan gambar dengan ponsel. Aplikasi juga dapat melakukan konversi warna negatif dengan cara mengambil gambar negatif yang terdapat pada *gallery* untuk dikonversi menjadi citra berwarna.

Jika pengambilan gambar sudah dilakukan maka aplikasi *Color It* akan langsung menampilkan citra berwarna dari hasil konversi citra negatif. Pengguna juga dapat melakukan editing sederhana pada aplikasi seperti mengubah tingkat ketajaman warna *red, green, blue* serta menaikkan dan menurunkan transparansi menggunakan slider RGB yang tersedia pada aplikasi. Penggunaan *slider RGB* ini adalah untuk menyesuaikan warna yang sudah dikonversi oleh aplikasi *Color It*. Fitur lain yang terdapat pada aplikasi adalah *crop* yang dapat digunakan untuk memangkas gambar atau melakukan rotasi pada gambar, lalu ada juga fitur *brightness* dan juga *contras* untuk menaikkan dan menurunkan *brightness* dan ketajaman gambar. Setelah proses editing selesai maka pengguna dapat

menyimpan klise foto yang sudah dikonversi sehingga menjadi foto digital ke dalam penyimpanan ponsel.

3.2 Spesifikasi Kebutuhan Perangkat Lunak

Agar perangkat lunak ini dapat berjalan dengan semestinya dan sesuai dengan spesifikasinya, terdapat beberapa penunjang yang dibutuhkan di dalamnya. Berikut merupakan kebutuhan perangkat lunak baik dari segi fungsional dan juga dalam segi non-fungsional.

3.2.1 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan penjelasan mengenai kebutuhan perangkat lunak baik dari segi perangkat keras maupun dari perangkat lunak itu sendiri. Pada aplikasi “*Color It*” untuk dapat mengembangkan dan menjalankan aplikasi ini ada beberapa hal yang harus diperhatikan diantaranya:

1. Minimal *android* 9.
2. *Chipset dual core*.
3. *RAM* 2GB.
4. Kamera ponsel.

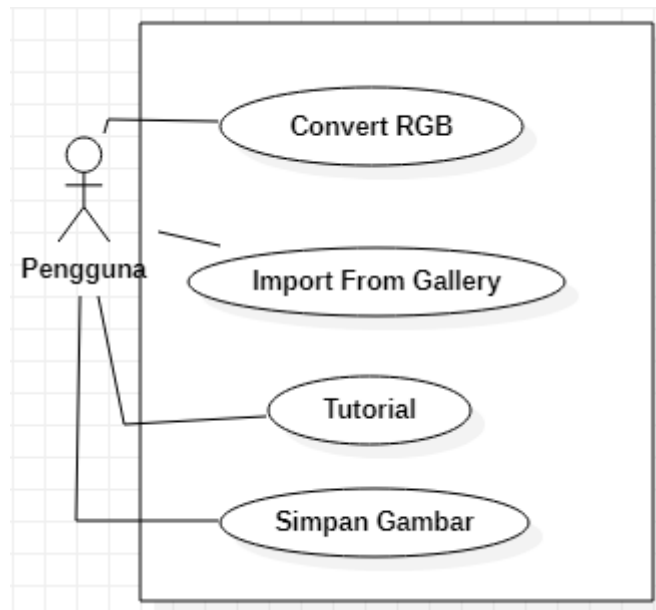
3.2.2 Kebutuhan Fungsional

Setelah kebutuhan non-fungsional terpenuhi selanjutnya terdapat beberapa kebutuhan fungsional yang terdapat dalam aplikasi “*Color It*” diantaranya:

1. Perangkat lunak dapat mengakses kamera ponsel
2. Perangkat lunak dapat memproses citra negatif menjadi positif (berwarna)
3. Perangkat lunak dapat bekerja secara *offline*
4. Perangkat lunak dapat menyimpan hasil pada saat pengguna melakukan *save*
5. Perangkat lunak dapat membuat *direktori* khusus bagi *file* yang sudah berwarna

3.3 Use Case Diagram

Use Case menggambarkan suatu interaksi yang terjadi antara aktor (pengguna) dan sistem, Pada sistem *Color It* pengguna dapat melakukan aktivitas mengubah warna citra negatif menjadi RGB dengan kamera, memotong gambar (*crop* gambar), mengubah warna dengan slider RGB, menyesuaikan kontras serta *brightness*, dan juga dapat menyimpan pada *direktori* ponsel.



Gambar 3.1 Use Case Color It

3.4 Skenario Use Case

Skenario *use case* ini menceritakan setiap aksi yang dilakukan oleh aktor kepada sistem dan juga menceritakan reaksi sistem terhadap aksi yang dilakukan oleh aktor. Aktor yang dimaksud adalah pengguna dan sistem merupakan aplikasi yang digunakan.

3.4.1 Skenario Use Case Convert RGB

Use Case : *Convert RGB*

Aktor : Pengguna

Kondisi awal : Halaman menu *Color It* sudah tampil.

Skenario normal : Pengguna menekan tombol *convert image* dan sistem langsung meminta izin penggunaan kamera serta memori internal untuk mengambil dan menyimpan gambar yang akan *diconvert* menjadi RGB.

Tabel 3.1 Skenario *Use Case Covert RGB*

Aktor	Sistem
1. Aktor menekan tombol <i>convert RGB</i>	
	2. Meminta izin penggunaan kamera dan penyimpanan
3. Mengizinkan penggunaan kamera dan penyimpanan	
	4. Membuka kamera
5. Mengambil gambar klise	
	6. Menampilkan hasil dan persetujuan pakai gambar atau foto ulang
7. Pengguna menggunakan gambar	
	8. Menampilkan hasil

Kondisi akhir : Sistem menampilkan gambar yang telah diambil.

Kondisi alternatif I : Pengguna tidak menggunakan gambar dan mengambil ulang gambar, lalu batal.

Tabel 3.2 Skenario Alternatif I *Use Case Convert RGB*

Aktor	Sistem
1. Aktor menekan tombol <i>convert RGB</i>	
	2. Meminta izin penggunaan kamera dan penyimpanan
3. Mengizinkan penggunaan kamera dan penyimpanan	
	4. Membuka kamera
5. Mengambil gambar klise	
	6. Menampilkan hasil dan persetujuan pakai gambar atau fotoulang
7. Pengguna memilih foto ulang	
	8. Membuka kamera
9. Pengguna menekan tombol back	
	10. Menampilkan halaman <i>home</i>

Kondisi akhir : Pengguna berada pada halaman *home*.

Kondisi alternatif II : Pengguna tidak memberikan izin penggunaan kamera dan penyimpanan.

Tabel 3.3 Skenario Alternatif II *Use Case Covert RGB*

Aktor	Sistem
1. Aktor menekan tombol <i>convert RGB</i>	
	2. Meminta izin penggunaan kamera dan penyimpanan
3. Tidak memberi izin penggunaan kamera dan penyimpanan	
	4. Menampilkan pesan <i>error</i> dan <i>home</i>

Kondisi akhir : Berada pada halaman *home Color It*.

Kondisi alternatif III : Pengguna tidak memberikan izin pada salah satu izin baik kamera atau penyimpanan.

Tabel 3.4 Skenario Alternatif III *Use Case Convert RGB*

Aktor	Sistem
1. Aktor menekan tombol <i>convert RGB</i>	
	2. Meminta izin penggunaan kamera dan penyimpanan
3. Tidak memberi izin penggunaan kamera atau penyimpanan	
	4. Menampilkan pesan <i>error</i> dan <i>home</i>

Kondisi akhir : Berada pada halaman *home*.

3.4.2 Skenario *Use Case Tutorial*

Use Case : Tutorial

Aktor : Pengguna

Kondisi awal : Menampilkan halaman menu utama.

Skenario normal : Pengguna menekan tombol *tutorial*.

Tabel 3.5 Skenario *Use Case Tutorial*

Aktor	Sistem
1. Menekan tombol <i>tutorial</i> .	
	2. Menampilkan halaman <i>YouTube</i> dari <i>Color It</i> .
3. Menekan tombol <i>play video tutorial</i> dan Kembali ke halaman <i>home</i> .	

3.4.3 Skenario Use Case Simpan Gambar

Use Case : Simpan Gambar

Aktor : Pengguna

Kondisi awal : Menampilkan foto yang telah *diconvert* dan telah diedit oleh pengguna dan sistem menampilkan tombol *next*.

Skenario normal : Pengguna menekan tombol *next* untuk masuk ke halaman *save*.

Pengguna juga dapat melakukan pengeditan sederhana pada halaman ini. Setelah selesai pengguna menekan tombol *save* untuk menyimpan gambar.

Tabel 3.6 Skenario Use Case Simpan Gambar

Aktor	Sistem
1. Menekan tombol <i>next</i>	
	2. Menampilkan halaman <i>save</i> dan fitur edit sederhana
3. Melakukan edit RGB, <i>crop</i> , <i>brightness</i> , serta <i>contrass</i> dan menekan tombol <i>save</i>	
	4. Menampilkan kotak dialog <i>save</i>
5. Melakukan <i>save</i>	
	6. Menampilkan <i>home</i> dengan <i>toast</i> bahwa gambar telah disimpan

Kondisi akhir : Menampilkan halaman *home*.

Skenario alternatif I : Pengguna menutup kotak dialog *save*.

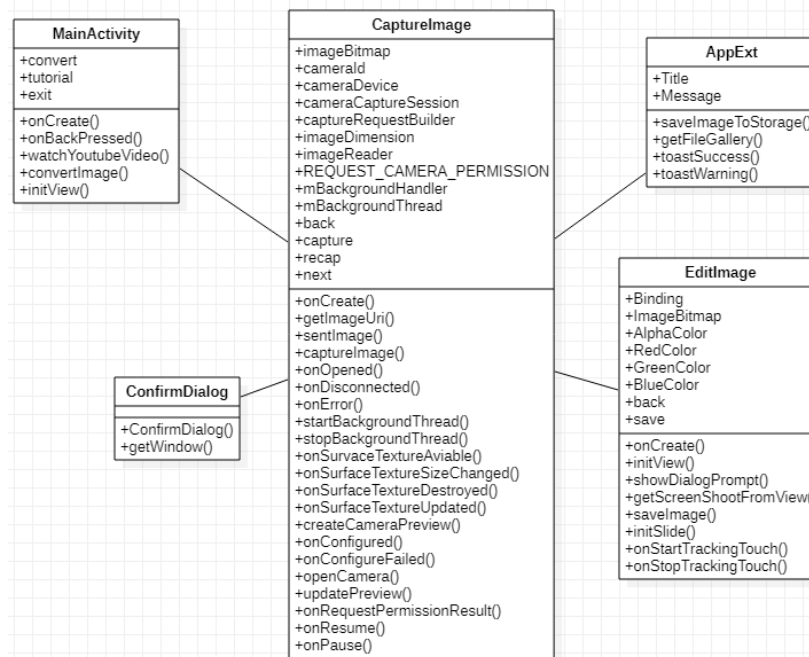
Tabel 3.7 Skenario Alternatif I Use Case Simpan Gambar

Aktor	Sistem
1. Menekan tombol <i>save</i>	
	2. Menampilkan kotak dialog <i>save</i>
3. Menutup kotak dialog	
	4. Menampilkan foto pada halaman <i>save</i>

Kondisi akhir : Menampilkan foto pada halaman *save*.

3.5 Class Diagram

Class diagram menggambarkan setiap kelas yang ada pada perangkat lunak, dan juga hubungan yang terjadi antar kelasnya yang nantinya dijadikan sebuah dasar pembangunan aplikasi *Color It* ini. Pada aplikasi *Color It* ini hanya terdapat sedikit kelas karena aplikasi ini masih sangat sederhana. Pada aplikasi ini terdapat kelas main menu yang menjadi pusat.



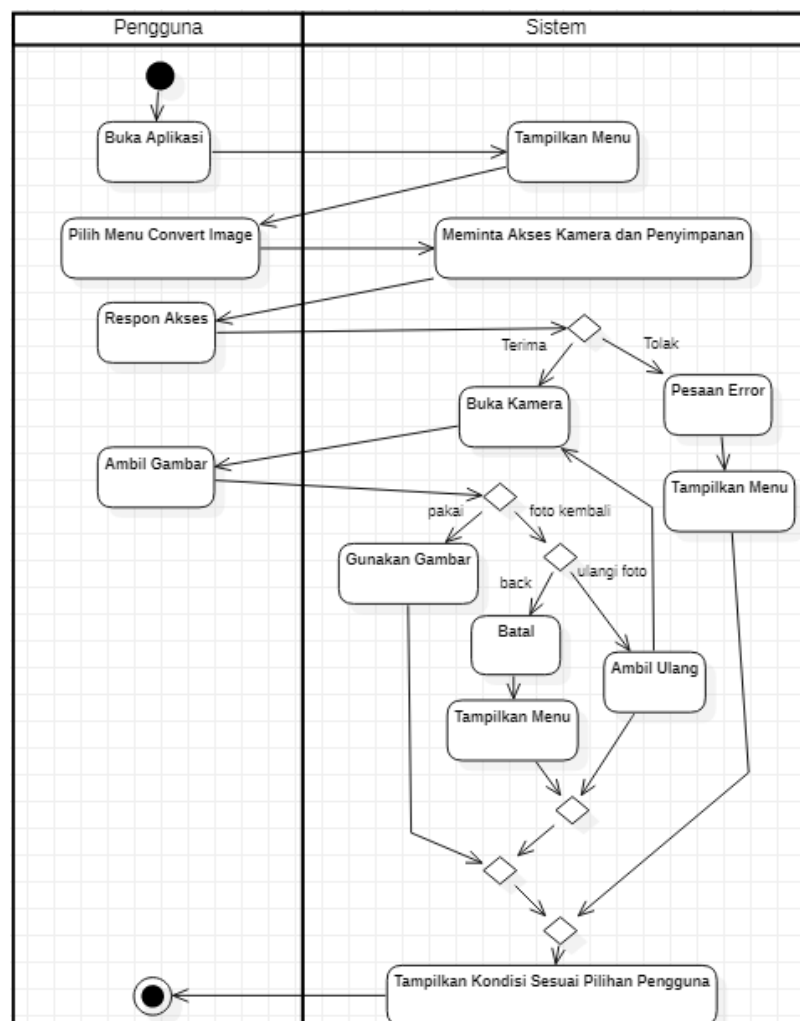
Gambar 3.2 Class Diagram *Color It*

3.6 Activity Diagram

Activity diagram merupakan gambaran suatu rangkaian kegiatan dari perangkat lunak yang dapat menunjukkan alur kontrol dari perangkat lunak tersebut. Berikut merupakan *activity diagram* yang dimiliki oleh aplikasi *Color It*.

3.6.1 Activity Diagram Convert RGB

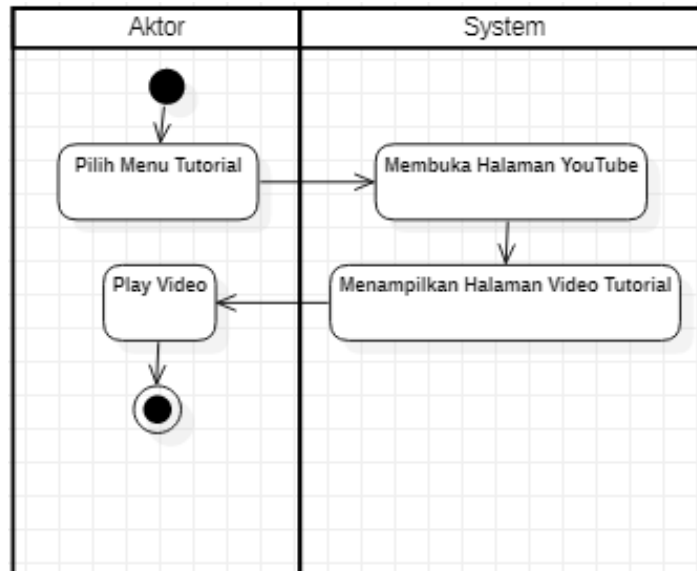
Pada tombol *convert* RGB aplikasi akan langsung meminta izin akses kamera. Setelah pengguna memberi izin kamera, pengguna bisa melakukan pengambilan gambar dan dapat mengulangi pengambilan gambar, setelah pengambilan gambar pengguna bisa mengkonfirmasi hasil gambar untuk selanjutnya dilakukan *convert* oleh sistem.



Gambar 3.3 Activity Diagram Convert RGB

3.6.2 Activity Diagram Tutorial

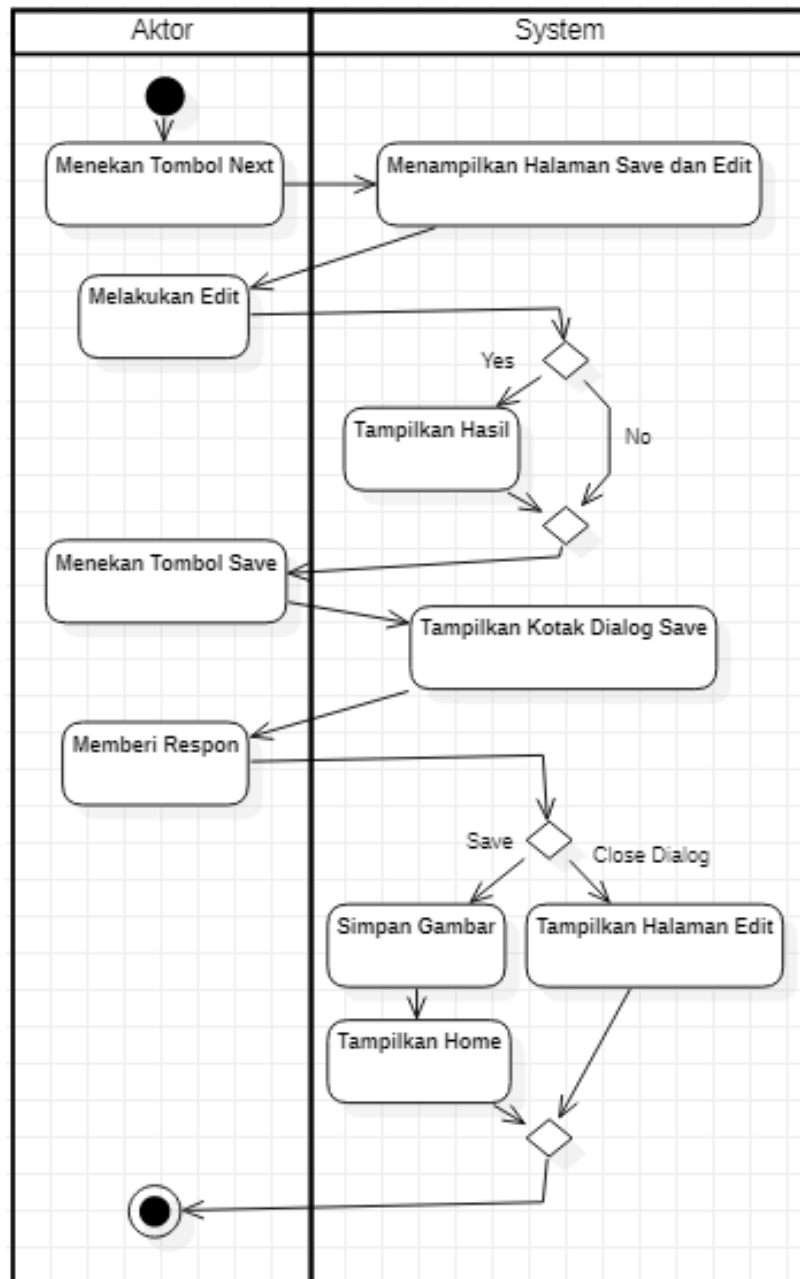
Pengguna dapat melihat bagaimana cara dan juga teknik dalam pengambilan gambar dengan aplikasi pada halaman *tutorial* dimana pengguna akan langsung diarahkan ke *Youtube* dari *Color It*.



Gambar 3.4 Activity Diagram Tutorial

3.6.3 Activity Diagram Simpan Gambar

Pengguna dapat menyimpan gambar yang telah *diconvert* dan dilakukan perubahan dengan Slider RGB (bila ada perubahan). Sebelum menyimpan sistem akan meminta akses ke memori internal ponsel, jika sudah diizinkan maka akan menyimpan gambar.

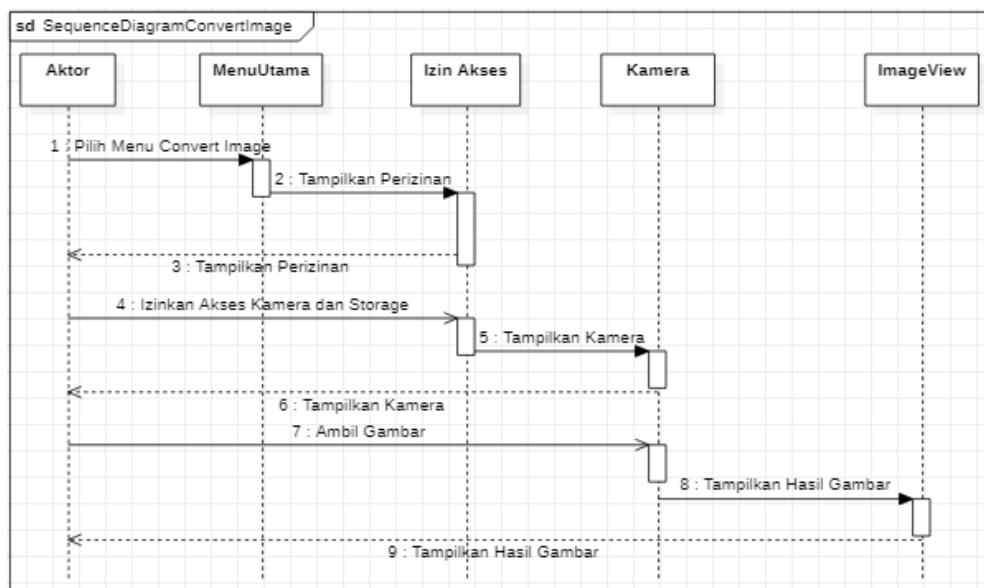


Gambar 3.5 Activity Diagram Simpan Gambar

3.7 Sequence Diagram

3.7.1 Sequence Diagram Convert Image

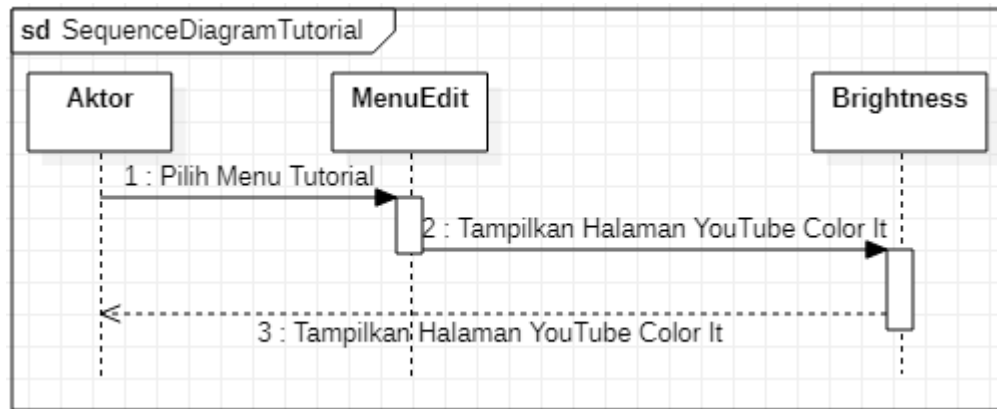
Pada *sequence diagram convert image* pengguna membuka aplikasi lalu memilih menu *Convert Image*. Setelah itu system akan merespon permintaan dari pengguna dengan menampilkan *layout* dari kamera. Kemudian pengguna bisa mengambil gambar dari kamera tersebut, akan ditampilkan hasil dari gambar yang diambil.



Gambar 3.6 Sequence Diagram Convert Image

3.7.2 Sequence Diagram Tutorial

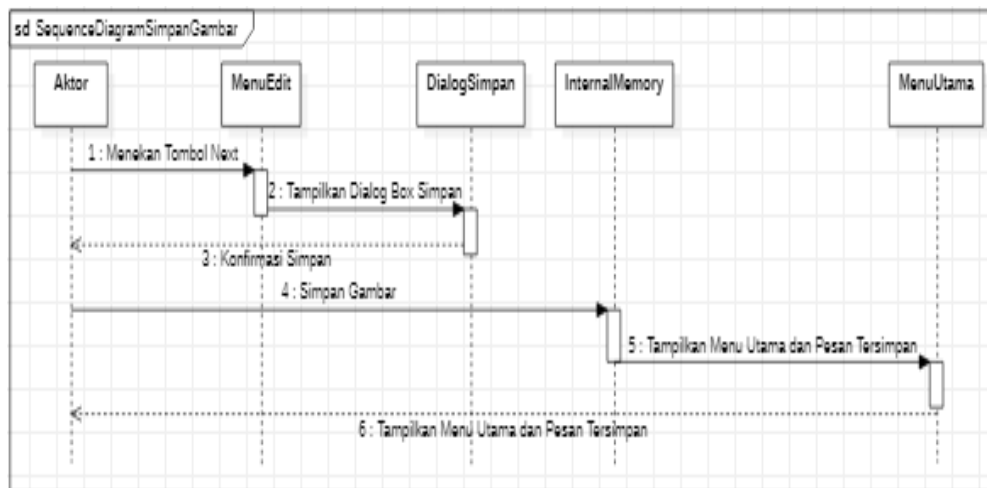
Pada *sequence diagram tutorial* pengguna dapat melihat video yang sudah dibuat oleh developer sebagai penjelasan dan arahan bagaimana cara menggunakan aplikasi *Color It*.



Gambar 3.7 Sequence Diagram Tutorial

3.7.3 Sequence Diagram Simpan Gambar

Pada *sequence diagram simpan gambar* pengguna dapat memilih simpan untuk menyimpan gambar pada ponsel. Jika pengguna menyimpan gambar maka sistem akan memproses gambar untuk dapat disimpan pada ponsel dan jika berhasil maka sistem akan memberikan informasi melalui pesan *toast* bahwa penyimpanan telah berhasil.



Gambar 3.8 Sequence Diagram Simpan Gambar

3.8 Rancangan Antarmuka

3.8.1 Rancangan Antarmuka *Home*

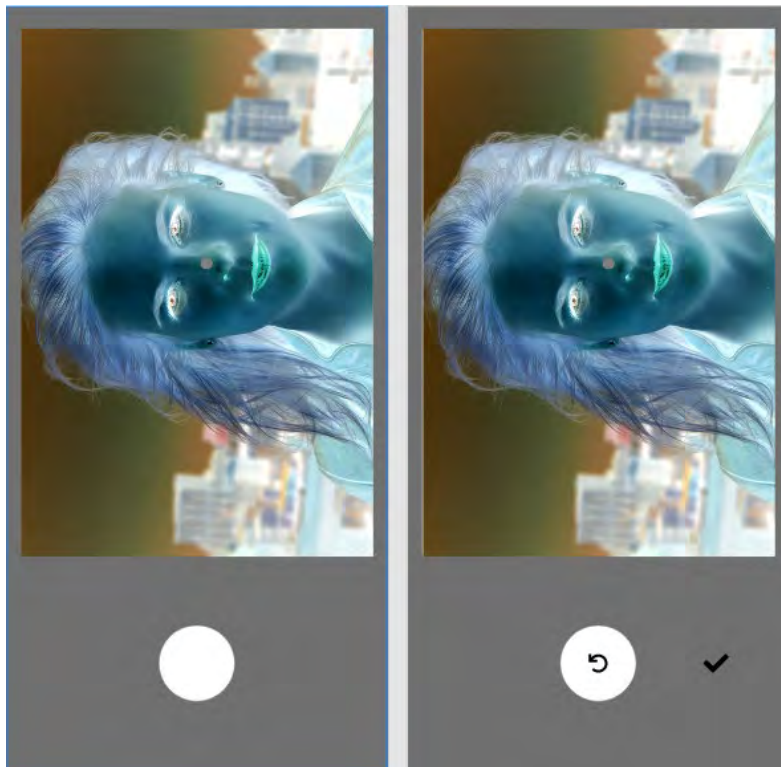
Pada rancangan antarmuka home terdiri dari empat bagian utama. Pada bagian tengah terdapat logo dari *Color It* yang digunakan sebagai branding dari aplikasi yang telah dibuat, lalu pada bagian bawah terdapat tiga buah *button* yang memiliki fungsinya masing-masing.



Gambar 3.9 Rancangan Antarmuka
Home

3.8.2 Rancangan Antarmuka Kamera

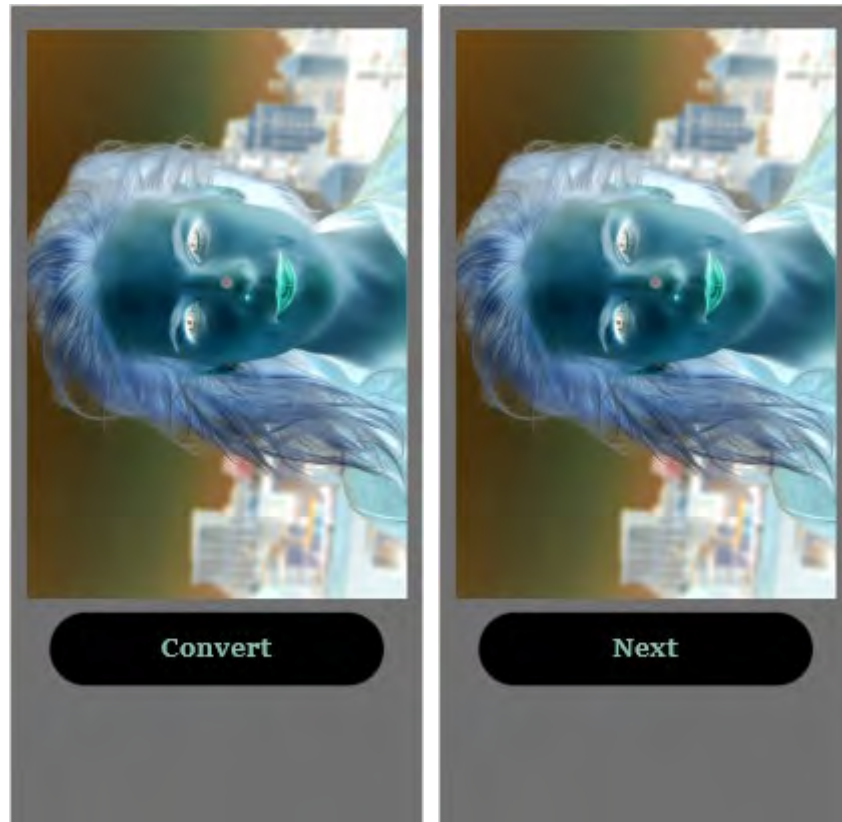
Pada rancangan antarmuka kamera terdiri dari dua buah *layout*. Pada *layout* pertama merupakan *layout* yang digunakan untuk menampilkan kamera secara live dengan satu buah tombol *capture* dibagian tengah bawahnya. Selanjutnya, pada *layout* kedua adalah *layout view* yang tampil ketika pengguna sudah mengambil gambar, untuk *layout view* sendiri tampilan yang diberikan tidak jauh berbeda dengan *layout* live kamera. Perbedaan yang ada pada *layout view* hanya terdapat pada bagian tombol di mana tombol *capture* dihilangkan dan digantikan dengan *recapture* pada bagian kiri bawah dan *checklist* pada bagian kanan bawah.



Gambar 3.10 Rancangan Antarmuka Kamera

3.8.3 Rancangan Antarmuka *Pick From Gallery*

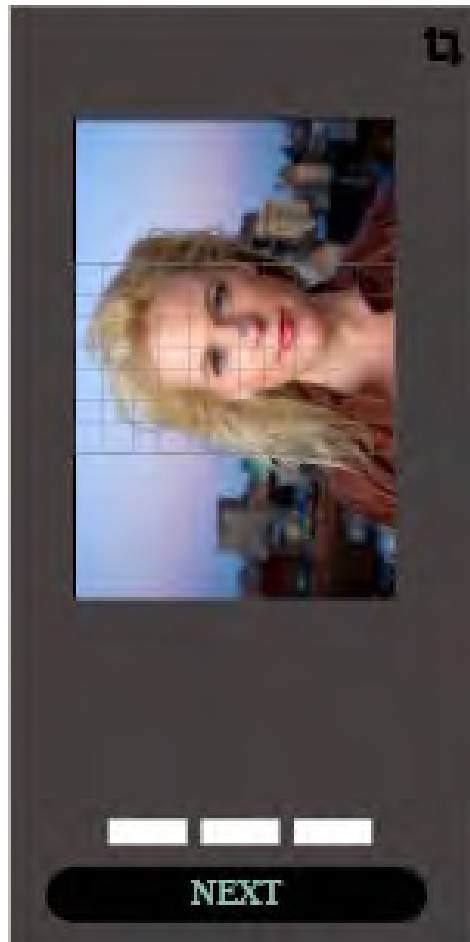
Setelah pengguna memilih gambar negatif yang akan dikonversi dari *gallery*, selanjutnya aplikasi akan menampilkan halaman *convert* gambar. Pada halaman ini hanya akan terdapat dua buah tombol seperti yang terdapat pada gambar dibawah.



Gambar 3.11 Rancangan Antarmuka *Pick From Gallery*

3.8.4 Rancangan Antarmuka *Crop* Gambar

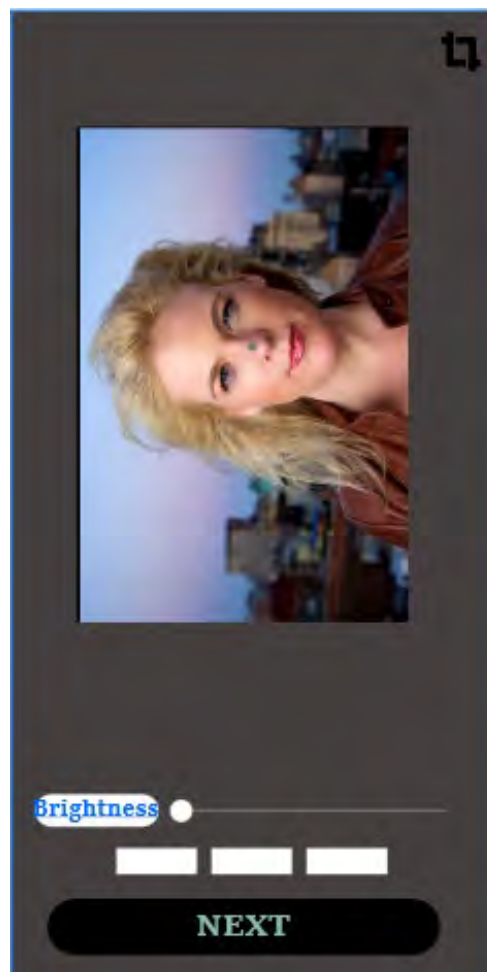
Pada halaman *crop* gambar terdapat grid dan juga kotak yang menjadi batasan bagian yang akan diambil jika pengguna menggunakan fitur *crop* gambar. Pada bagian bawah akan terdapat tiga buah tombol yang nantinya digunakan sebagai tempat untuk membuka fitur lain. Terdapat juga tombol *next* yang menjadi tombol untuk melakukan simpan gambar.



Gambar 3.12 Rancangan Antarmuka *Crop* Gambar

3.8.5 Rancangan Antarmuka Ubah Kecerahan

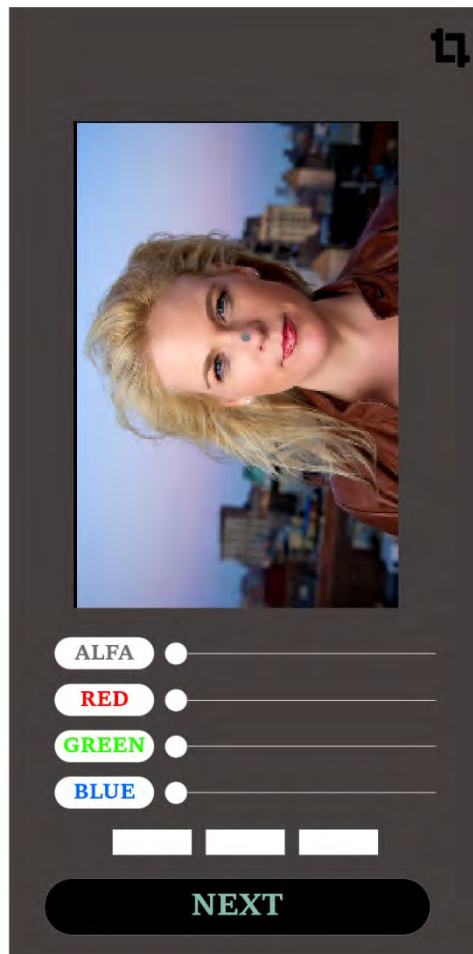
Pada halaman ubah kecerahan tidak ada perbedaan dengan rancangan antarmuka pada halaman *crop* gambar. Perbedaan yang ada hanya pada bagian grid yang dihilangkan dan digantikan dengan *slider brightness*. Pada bagian bawah tidak ada perubahan akan terdapat tiga buah tombol yang nantinya digunakan sebagai tempat untuk membuka fitur lain. Terdapat juga tombol *next* yang menjadi tombol untuk melakukan simpan gambar.



Gambar 3.13 Rancangan Antarmuka Ubah Kecerahan

3.8.6 Rancangan Antarmuka Edit RGB

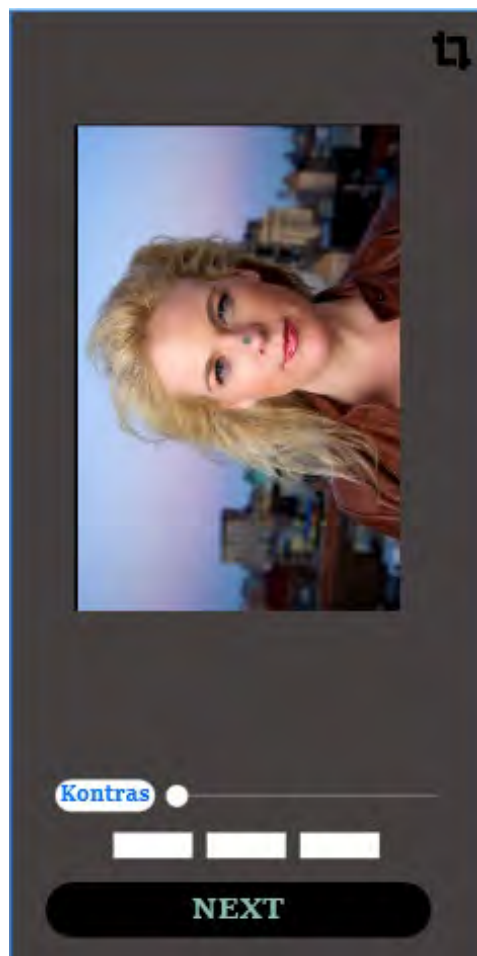
Pada halaman edit RGB terdapat satu buah gambar yang dipakai untuk menampilkan gambar RGB yang sudah selesai. Pada bagian bawah dari gambar terdapat empat buah *slider* yang digunakan sebagai pengatur tingkat kecerahan dari warna *alfa*, *red*, *green* dan *blue*. Pada bagian bawah dari halaman edit terdapat satu buah tombol *next*.



Gambar 3.14 Rancangan Antarmuka Edit RGB

3.8.7 Rancangan Antarmuka Ubah Kontras

Pada rancangan halaman ubah kontras masih sama dengan antarmuka yang terdapat pada halaman *crop*, ubah kecerahan, dan juga halaman edit RGB. Perbedaannya hanya terdapat pada *slider* yang tampil pada halaman edit RGB dan juga kecerahan akan dihilangkan lalu digantikan dengan slider kontras. Pada bagian bawah masih terdapat tiga buah tombol yang nantinya digunakan sebagai tempat untuk membuka fitur lain. Terdapat juga tombol *next* yang menjadi tombol untuk melakukan simpan gambar.



Gambar 3.15 Rancangan Antarmuka Ubah Kontras

3.8.8 Rancangan Antarmuka Save

Pada rancangan antarmuka save background dari *layout* merupakan background dari antarmuka edit RGB yang dikurangi tingkat kecerahannya dan untuk bagian bawah terdapat kotak *dialog* yang menampilkan pertanyaan konfirmasi dan dua buah tombol yaitu *close* dialog dan *save*.



Gambar 3.16 Rancangan Antarmuka Save

BAB IV

IMPLEMENTASI DAN PENGUJIAN PERANGKAT LUNAK

4.1 Spesifikasi Kebutuhan Perangkat Keras

Berikut ini merupakan kebutuhan perangkat keras yang digunakan oleh penulis dalam membuat dan menguji perangkat lunak yang sedang dibangun.

Tabel 4.1 Spesifikasi Perangkat Keras Ponsel

Perangkat Keras	Spesifikasi
<i>Processor</i>	1.4Hz <i>Snapdragon 425 Quad-core</i>
<i>Android Version</i>	10
<i>Memory (RAM)</i>	3 GB
<i>Storage</i>	32 GB
<i>Graphic</i>	<i>Adreno 308</i>
<i>Screen</i>	6 <i>Inch</i>

4.2 Pengujian Perangkat Keras

Pada pengujian perangkat keras aplikasi diinstall pada beberapa device, pada aplikasi ini penulis menggunakan tiga buah device berbeda dengan tipe yang berbeda pula. Adapun perangkat yang digunakan sebagai berikut ini:

Tabel 4.2 Pengujian Perangkat Keras

Jenis <i>device</i>	Spesifikasi	Hasil
1. Ponsel: <i>Galaxy Young 2</i>		
	Processor: 1.0 GHz Cortex-A7	Pada proses instalasi tidak ada permasalahan yang timbul, namun ketika instalasi telah selesai aplikasi tidak dapat berjalan, karena perangkat tidak memenuhi spesifikasi yang dibutuhkan oleh aplikasi. Sehingga menyebabkan aplikasi menjadi <i>force close</i> .
	OS: <i>Android KitKat</i>	
	RAM: 512MB	
	Storage: 4GB	
	Graphic: TFT,256K <i>colors</i>	
	Screen: 3.5 <i>Inch</i>	
2. Ponsel: <i>Galaxy A52</i>		
	Processor: <i>Qualcom Snapdragon 720G</i>	Aplikasi dapat diinstal pada perangkat ini dengan normal dan juga dapat membuka aplikasi dengan normal. Seluruh fitur yang ada pada aplikasi dapat dijalankan pada <i>device</i> ini.
	OS: <i>Android Honeycomb</i>	
	RAM: 8GB	
	Storage: 256GB	
	Graphic: <i>Adreno 618</i>	
	Screen: 6.5 <i>Inch</i>	
3. Tablet: <i>Galaxy Tab A7 Lite</i>		
	Processor: Mediatek MT8768T	Aplikasi dapat terinstal dengan benar dan juga dapat berjalan sesuai yang diharapkan. Aplikasi juga
	OS: <i>Android Honeycomb</i>	
	RAM: 3GB	

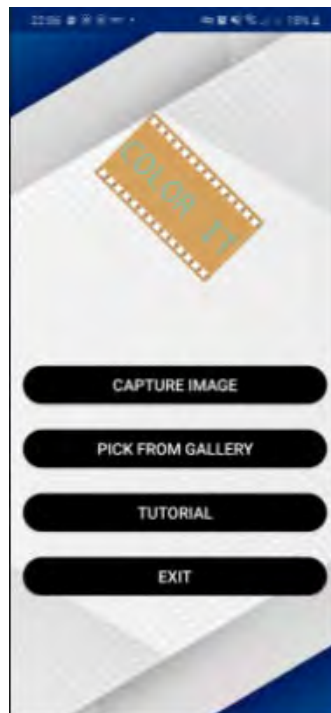
Jenis <i>device</i>	Spesifikasi	Hasil
	Storage: 32GB	mampu menampilkan fitur yang ada dalam aplikasi
	Graphic: PowerVR GE8320	
	Screen: 8.7 Inch	

4.3 Implementasi Antarmuka

4.3.1 Pengujian Antarmuka Home

Pada saat membuka aplikasi *Color It* untuk pertama kali maka akan ditampilkan sebuah menu pilihan . Pada menu pilihan tersebut terdapat beberapa tombol antara lain sebagai berikut:

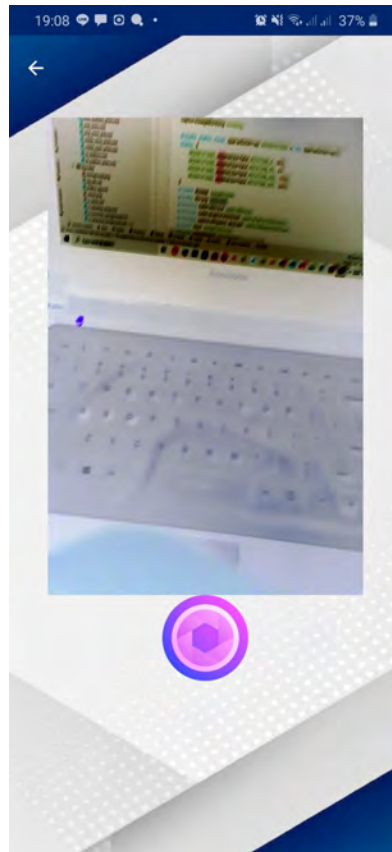
1. Logo *Color It* melambangkan identitas dari aplikasi yang dibuat.
2. Tombol *Convert Image* untuk memulai mengambil gambar negatif dengan kamera ponsel.
3. Tombol *pick from gallery* untuk mengambil gambar dari poenyimpanan perangkat.
4. Tombol Tutorial untuk mengarahkan pengguna ke dalam *YouTube* dari *Color It* dimana terdapat Langkah bagaimana melakukan pengambilan gambar yang benar.
5. Tombol *Exit* untuk keluar dari aplikasi *Color It*.



Gambar 4.1 Pengujian Antarmuka Home

4.3.2 Pengujian Antarmuka Kamera

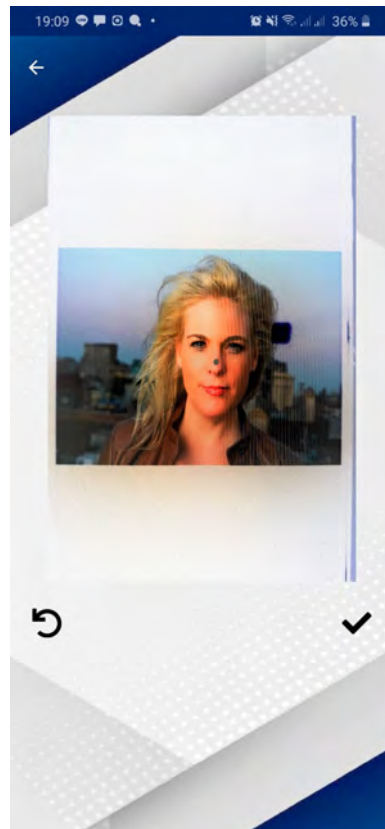
Pada menu *home* aplikasi terdapat sebuah tombol *convert image* yang akan membuka kamera ponsel. Fungsi dari kamera ini adalah untuk melakukan pengambilan gambar yang terdapat pada klise foto untuk dilakukan proses *convert* ke RGB. Pada kamera hanya akan terdapat satu buah tombol untuk melakukan pengambilan gambar.



Gambar 4.2 Pengujian Antarmuka Kamera

4.3.3 Pengujian Antarmuka *View Image*

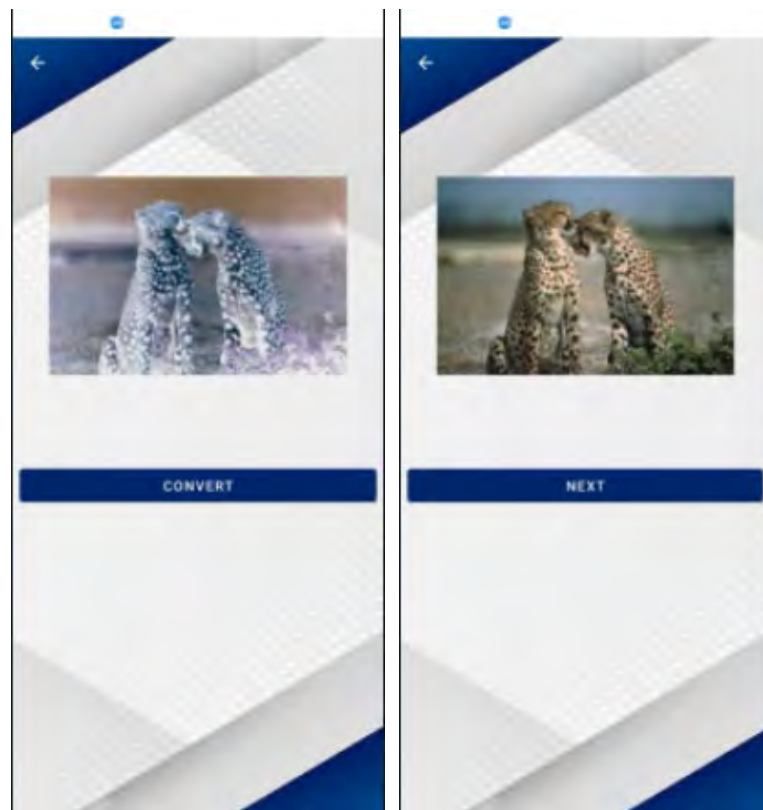
Setelah melakukan pengambilan gambar maka ponsel akan menampilkan gambar dengan *layout* kamera yang sama namun ada perubahan fungsi pada tombol pengambilan gambar, di mana tombol pengambilan gambar akan berubah menjadi ulang pengambilan gambar dan terdapat tombol tambahan yaitu konfirmasi gambar diwakili lambang *checklist*.



Gambar 4.3 Pengujian Antarmuka *View Image*

4.3.4 Pengujian Antarmuka *Pick From Gallery*

Pada halaman *pick from gallery* akan terdapat dua buah tombol. Tombol akan ditampilkan bergantian antara tombol *convert* dan *next*. Setelah pengguna memilih gambar dari *gallery* pengguna diarahkan ke halaman *convert*. Pertama halaman akan menampilkan tombol *convert* kemudian setelah pengguna menekan tombol *convert*, hasil perubahan akan ditampilkan dan tombol akan digantikan dengan tombol *next*.



Gambar 4.4 Pengujian
Antarmuka *Pick From Gallery*

4.3.5 Pengujian Antarmuka *Crop Image*

Pada halaman antarmuka *crop image* terdapat beberapa tombol yang dapat digunakan diantaranya, yaitu:

1. Tombol silang digunakan untuk kembali ke halaman sebelumnya.
2. Tombol *checklist* yang dipakai untuk menerapkan hasil crop.
3. Tombol *crop* untuk memangkas gambar sesuai dengan ukuran yang sediakan.
4. Tombol *rotate* untuk menyesuaikan rotasi gambar..
5. Tombol *scale* untuk memangkas foto sesuai ukuran yang diinginkan.

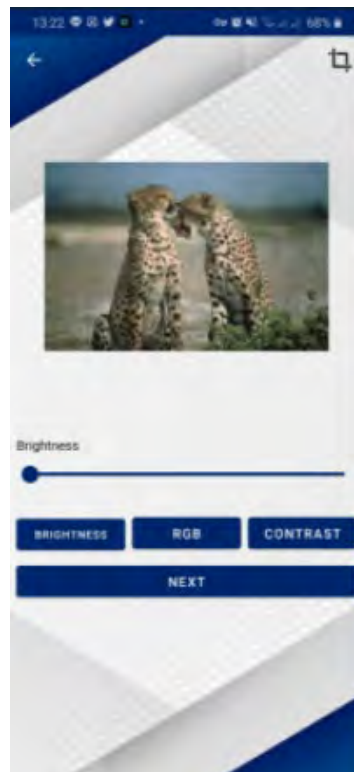


Gambar 4.5 Pengujian Antarmuka *Crop Image*

4.3.6 Pengujian Antarmuka Ubah Kecerahan

Pada halaman edit terdapat pilihan *brightness* yang ketika dipilih akan menampilkan beberapa tombol, antara lain:

1. Tombol *back* digunakan untuk kembali ke halaman sebelumnya.
2. *Slider brightness* yang dipakai untuk mengatur kecerahan gambar.
3. Tombol RGB untuk masuk ke dalam fitur RGB
4. Tombol kontras untuk masuk kedalam fitur ubah kontras.
5. Tombol *crop* untuk masuk kedalam fitur *crop*.
6. Tombol *next* untuk melanjutkan penyimpanan gambar yang sudah selesai.



Gambar 4.6 Pengujian Antarmuka Ubah Kecerahan

4.3.7 Pengujian Antarmuka Edit RGB

Pada halaman edit terdapat pilihan RGB yang akan memunculkan beberapa tombol yang dapat digunakan di antaranya, yaitu:

1. Tombol *back* dipakai untuk kembali ke halaman sebelumnya.
2. *Slider red, green, dan blue* merupakan penentu warna yang akan dinaikan.
3. Tombol *brightness* untuk masuk ke dalam fitur *brightness*.
4. Tombol kontras untuk masuk kedalam fitur ubah kontras.
5. Tombol *crop* untuk masuk kedalam fitur *crop*.
6. Tombol *next* untuk melanjutkan penyimpanan gambar yang sudah selesai.

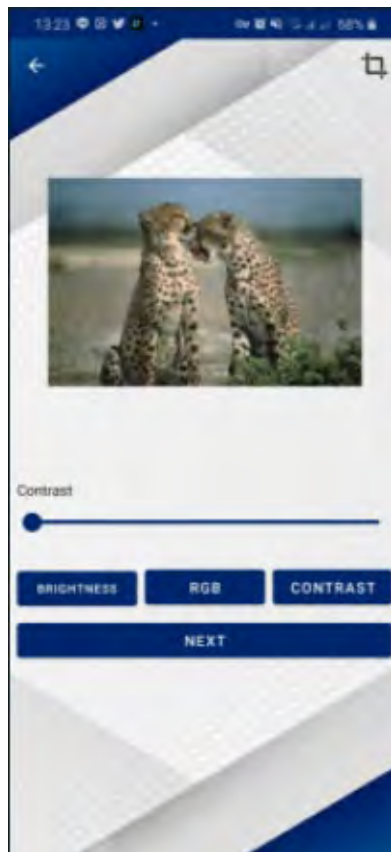


Gambar 4.7 Pengujian Antarmuka Edit RGB

4.3.8 Pengujian Antarmuka Ubah Kontras

Pada halaman ubah kontras terdapat beberapa tombol yang memiliki fungsinya masing masing, antara lain:

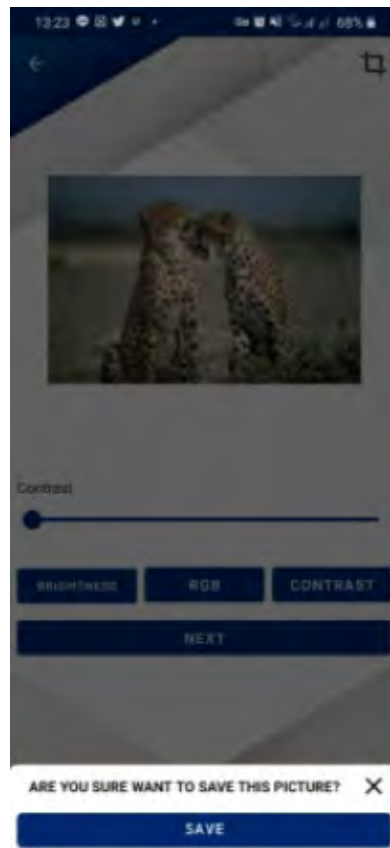
1. Tombol *back* dipakai untuk kembali ke halaman sebelumnya.
2. *Slider kontras* merupakan penentu warna yang akan dinaikan.
3. Tombol *brightness* untuk masuk ke dalam fitur *brightness*.
4. Tombol RGB untuk masuk kedalam fitur RGB.
5. Tombol *crop* untuk masuk kedalam fitur *crop*.
6. Tombol *next* untuk melanjutkan penyimpanan gambar yang sudah selesai.



Gambar 4.8 Pengujian Antarmuka Ubah Kontras

4.3.9 Pengujian Antarmuka Save

Pada antarmuka simpan ini terdapat dua buah tombol yaitu tombol simpan dan *close dialog*. Jika kita melakukan simpan maka secara otomatis gambar akan tersimpan dan aplikasi akan langsung menuju *home* aplikasi dengan menampilkan *toast* bahwa gambar telah disimpan. Namun jika pengguna merasa tidak perlu menyimpannya maka pengguna bisa langsung keluar dengan menekan tombol *back* yang terdapat pada ponsel.



Gambar 4.9 Pengujian Antarmuka Save

4.4 Pengujian Fungsi

Pengujian fungsi perangkat lunak meliputi pengujian *convert image* untuk membuka layout kamera, kemudian *tutorial* yang terdapat pada halaman *home* untuk membuka tutorial pada *YouTube*, dan *exit* untuk keluar dari aplikasi.

4.4.1 Pengujian *Button Convert Image*

Button convert image yang terdapat pada halaman *home* digunakan oleh user untuk memulai mengambil gambar yang akan *diconvert* menjadi warna RGB. Tujuan dari pengujian ini adalah untuk melihat apakah *button* yang dibuat dapat menampilkan kamera pada pengguna yang mengizinkan akses kamera saja atau tidak. Pada tabel di bawah ditampilkan beberapa skenario pengujian yang dilakukan, antara lain:

Tabel 4.3 Skenario Pengujian
Button Convert Image

Skenario pengujian	Hasil yang diharapkan
Pengguna menekan tombol <i>convert image</i> namun tidak memberi akses kamera.	Menampilkan pesan <i>error</i> .
Pengguna menekan tombol <i>convert image</i> dan memilih selalu memberi akses kamera.	Menampilkan <i>layout</i> dari kamera.
Pengguna menekan tombol <i>convert image</i> dan memberi akses kamera sekali saja.	Menampilkan <i>layout</i> dari kamera dan meminta akses kembali ketika pengguna menutup aplikasi dan membuka kembali.

Setelah dilakukan pengujian berdasarkan skenario yang terdapat pada tabel di atas maka dapat diambil data sebagai berikut:

Tabel 4.4 Hasil Uji Skenario
Button Convert Image

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna menekan tombol <i>convert image</i> namun tidak memberi akses kamera.	Menampilkan pesan <i>error</i> .	Menampilkan pesan <i>error</i> "Sorry!!!, you can't use this app without granting permission."	Valid
Pengguna menekan tombol <i>convert image</i> dan memilih selalu memberi akses kamera.	Menampilkan <i>layout</i> dari kamera.	Menampilkan <i>live view</i> dari kamera.	Valid

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna menekan tombol <i>convert image</i> dan memberi akses kamera sekali saja.	Menampilkan <i>layout</i> dari kamera dan meminta akses kembali ketika pengguna menutup aplikasi dan membuka kembali.	Menampilkan <i>liveview</i> dari kamera dan meminta akses kamera ketika pengguna menutup aplikasi dan membuka kembali.	Valid

4.4.2 Pengujian *Button Pick From Gallery*

Button pick from gallery dipakai oleh pengguna sebagai alternatif apabila citra yang dimiliki berada pada penyimpanan perangkat yang dimiliki, sehingga pengguna dapat dengan mudah mengambil citra yang akan dikonversi dari penyimpanan perangkat. Pada tabel pengujian akan dijelaskan apa saja scenario pengujian dan hasil yang diharapkan beserta hasil pengujian.

Tabel 4.5 Skenario Pengujian
Button Pick From Gallery

Skenario pengujian	Hasil yang diharapkan
Pengguna menekan <i>button pick from gallery</i>	Membuka <i>gallery</i> ponsel pengguna
Pengguna memilih gambar yang akan dikonversi	Membuka halaman <i>convert</i>
Pengguna menekan tombol <i>convert</i> pada halaman <i>convert</i>	Mengubah menampilkan hasil konversi dan mengubah fungsi tombol
Pengguna menekan tombol <i>next</i> setelah menekan tombol <i>convert</i>	Masuk ke halaman edit

Hasil pengujian yang dilakukan berdasarkan tabel scenario pengujian di atas, menunjukkan hasil sebagai berikut:

Tabel 4.6 Hasil Uji Skenario
Button Pick From Gallery

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna menekan <i>button pick from gallery</i>	Membuka <i>gallery</i> ponsel pengguna	Membuka <i>gallery</i> ponsel pengguna	Valid
Pengguna memilih gambar yang akan dikonversi	Membuka halaman <i>convert</i>	Membuka halaman <i>convert</i>	Valid
Pengguna menekan tombol <i>convert</i> pada halaman <i>convert</i>	Mengubah menampilkan hasil konversi dan mengubah fungsi tombol	Mengubah menampilkan hasil konversi dan mengubah fungsi tombol	Valid

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna menekan tombol <i>next</i> setelah menekan tombol <i>convert</i>	Masuk ke halaman edit	Masuk ke halaman edit	Valid

4.4.3 Pengujian *Button Tutorial*

Button tutorial dapat digunakan oleh pengguna untuk melihat video bagaimana caranya menggunakan aplikasi ini. *Button tutorial* akan langsung masuk ke halaman *YouTube* dan langsung memutar video *tutorial* dari *ColorIt*. Pada tabel di bawah akan dijelaskan skenario pengujian yang akan dilakukan pada *button tutorial*, antara lain:

Tabel 4.7 Skenario Pengujian
Button Tutorial

Skenario pengujian	Hasil yang diharapkan
Pengguna menekan <i>button tutorial</i> pada halaman <i>home</i> .	Membuka aplikasi <i>YouTube</i> atau <i>browser</i> untuk memulai video tutorial.

Dari scenario diatas pengguna melakukan skenario pengujian tersebut pada dua perangkat yang berbeda. Pada perangkat pertama ponsel memiliki aplikasi *YouTube* dan pada ponsel kedua tidak memiliki aplikasi *YouTube*, kemudian didapatkanlah hasil seperti di bawah ini:

Tabel 4.8 Hasil Uji
Skenario *Button Tutorial*

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna menekan <i>button tutorial</i> pada halaman <i>home</i> .	Membuka aplikasi <i>YouTube</i> atau <i>browser</i> untuk memulai video tutorial.	Membuka aplikasi <i>YouTube</i> .	Valid
Pengguna menekan <i>button tutorial</i> pada halaman <i>home</i> .	Membuka aplikasi <i>YouTube</i> atau <i>browser</i> untuk memulai video tutorial.	Membuka <i>browser</i> sebagai media pengganti <i>YouTube</i> .	Valid

4.4.4 Pengujian *Button Capture* Pada Kamera

Setelah pengguna masuk ke dalam *layout live view* kamera, pengguna dapat mulai mengambil gambar dengan menekan tombol yang terdapat pada bagian bawah *live view*

kamera. Tombol tersebut dipakai untuk mengambil gambar negatif yang akan *diconvert* menjadi gambar RGB. Pada tabel di bawah akan di jelaskan scenario pengujian apa saja yang akan dilakukan, antara lain:

Tabel 4.9 Skenario Pengujian
Button Capture Pada Kamera

Skenario pengujian	Hasil yang diharapkan
Pengguna melakukan klik pada tombol <i>capture</i> yang disediakan.	Mengambil gambar dan menampilkan hasil.
Pengguna menekan tombol dalam waktu lama lalu melepaskan.	Mengambil gambar yang ada di depan kamera pada saat pengguna melepas tombol.

Pengujian yang telah dilakukan berdasarkan skenario di atas dapat dijelaskan seperti tabel di bawah ini:

Tabel 4.10 Hasil Uji Skenario
Button Capture Pada Kamera

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna melakukan klik pada tombol <i>capture</i> yang disediakan.	Mengambil gambar dan menampilkan hasil.	Mengambil gambar dan ditampilkan.	Valid
Pengguna menekan tombol dalam waktu lama lalu melepaskan.	Mengambil gambar yang ada di depan kamera pada saat pengguna melepas tombol dan kemudian ditampilkan.	Mengambil gambar saat tombol dilepas dan menampilkan gambar.	Valid

4.4.5 Pengujian Fitur *Crop* Gambar

Pada fitur *crop* gambar pengguna dapat menggunakan fitur ini untuk memangkas bagian yang tidak dibutuhkan pada gambar yang sudah diambil. Fitur ini juga dapat memfokuskan gambar yang telah diambil hanya kepada objeknya saja. Pengguna juga dapat melakukan rotasi pada gambar dengan fitur ini.

Tabel 4.11 Skenario Pengujian Fitur
Crop Gambar

Skenario pengujian	Hasil yang diharapkan
Pengguna memangkas gambar hanya pada bagian objek.	Gambar menjadi lebih kecil sesuai apa yang akan diambil.

Skenario pengujian	Hasil yang diharapkan
Pengguna melakukan rotasi 90° ke arah kanan dengan posisi awal vertikal.	Gambar berputar menjadi horizontal.

Setelah dilakukan pengujian pada fitur *crop* gambar berikut hasil yang dapat diberikan seperti pada tabel di bawah ini:

Tabel 4.12 Hasil Uji
Skenario *Crop* Gambar

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna memangkas gambar hanya pada bagian objek.	Gambar menjadi lebih kecil sesuai apa yang akan diambil.	Gambar dapat diperkecil dengan fitur <i>crop</i> .	Valid
Pengguna melakukan rotasi 90° ke arah kanan dengan posisi awal vertikal.	Gambar berputar menjadi horizontal.	Gambar berputar 90°.	Valid

4.4.6 Pengujian Fitur *Brightness*

Pada fitur *brightness* pengguna dapat menaikkan dan menurunkan kecerahan yang ada pada gambar sehingga pengguna dapat dengan mudah menaikkan kecerahan foto yang terlihat kurang terang. Adapun skenario pengujian yang dilakukan seperti di bawah ini:

Tabel 4.13 Skenario Pengujian
Fitur *Brightness*

Skenario pengujian	Hasil yang diharapkan
Pengguna menaikkan <i>brightness</i> pada gambar dengan fitur <i>brightness</i> .	Gambar menjadi lebih terang daripada sebelumnya.
Pengguna menurunkan <i>brightness</i> pada gambar.	Gambar menjadi lebih gelap.
Pengguna berpindah fitur setelah menaikkan <i>brightness</i> .	Gambar menyimpan <i>brightness</i> yang sudah diatur.

Setelah dilakukan pengujian pada aplikasi didapatkanlah hasil uji seperti di bawah ini:

Tabel 4.14 Hasil Uji
Skenario Fitur *Brightness*

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna menaikkan kecerahan pada	Gambar menjadi lebih terang	Gambar menjadi lebih terang.	Valid

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
gambar dengan fitur <i>brightness</i> .	daripada sebelumnya.		
Pengguna menurunkan kecerahan pada gambar.	Gambar menjadi lebih gelap.	Gambar menjadi lebih gelap.	Valid
Pengguna berpindah fitur setelah menaikkan kecerahan.	Gambar menyimpan kecerahan yang sudah diatur.	Gambar berubah menjadi seperti semula.	Valid

4.4.7 Pengujian *Slider RGB*

Pada *slider RGB* terdapat empat buah *slider*. *Slider* pertama merupakan warna *alpha* di mana warna *alpha* ini merupakan *opacity* yang dapat diatur, semakin kecil *alpha* yang dipakai maka gambar akan semakin transparan. Ketiga *slider* di bawah *alpha* adalah *red*, *green*, dan *blue* di mana warna tersebut jika semakin besar akan semakin terang. Pada tabel di bawah akan di tampilkan beberapa skenario yang akan dilakukan untuk pengujian, antara lain:

Tabel 4.15 Skenario Pengujian
Slider RGB

Skenario pengujian	Hasil yang diharapkan
Pengguna tidak melakukan perubahan pada <i>slider</i> .	Menampilkan gambar asli setelah <i>convert</i> .
Pengguna menggeser slider RGB namun tidak menaikkan <i>alpha</i> .	Gambar tidak tampil karena <i>alpha</i> lebih kecil dari RGB.
Pengguna menggeser <i>slider</i> dan menaikkan <i>alpha</i> .	Gambar di tampilkan dengan warna yang sudah diatur.
Pengguna menaikkan <i>alpha</i> tanpa mengubah RGB.	Menampilkan gambar asli setelah <i>convert</i> .

Setelah dilakukan pengujian dengan skenario yang ada pada tabel di atas, didapatkan beberapa data sebagai berikut:

Tabel 4.16 Hasil Uji
Skenario Slider RGB

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna tidak melakukan perubahan pada <i>slider</i> .	Menampilkan gambar asli setelah <i>convert</i> .	Menampilkan gambar hasil <i>convert</i> .	Valid
Pengguna menggeser slider	Gambar tidak tampil karena <i>alpha</i>	Gambar tidak ditampilkan.	Valid

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
RGB namun tidak menaikkan <i>alpha</i> .	lebih kecil dari RGB.		
Pengguna menggeser <i>slider</i> dan menaikkan <i>alpha</i> .	Gambar di tampilkan dengan warna yang sudah diatur.	Gambar ditampilkan dengan warna yang berbeda dengan hasil <i>convert</i> asli.	Valid
Pengguna menaikkan <i>alpha</i> tanpa mengubah RGB.	Menampilkan gambar asli setelah <i>convert</i> .	Menampilkan gambar hasil <i>convert</i> .	Valid

4.4.8 Pengujian Fitur Kontras

Pengguna dapat menaikkan kontras yang terdapat pada gambar dengan mengubah slider yang terdapat pada halaman kontras. Skenario pengujian yang akan dilakukan adalah sebagai berikut ini:

Tabel 4.17 Skenario Pengujian
Fitur Kontras

Skenario pengujian	Hasil yang diharapkan
Pengguna menaikkan kontras pada gambar dengan fitur kontras.	Gambar menjadi lebih pekat baik dari segi warna dan outline.
Pengguna menurunkan kontras pada gambar.	Gambar menjadi lebih gelap.
Pengguna berpindah fitur setelah menaikkan <i>brightness</i> .	Gambar menyimpan kontras yang sudah diatur.

Setelah dilakukan pengujian fitur kontras didapatkan hasil seperti yang tertera pada tabel di bawah ini:

Tabel 4.18 Hasil Uji
Skenario Fitur Kontras

Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna menaikkan kontras pada gambar dengan fitur kontras.	Gambar menjadi lebih pekat baik dari segi warna dan outline.	Gambar menjadi lebih pekat warnanya.	Valid
Pengguna menurunkan kontras pada gambar.	Gambar menjadi lebih gelap.	Gambar menjadi lebih gelap.	Valid
Pengguna berpindah fitur setelah menaikkan kontras.	Gambar menyimpan kontras yang sudah diatur.	Gambar berubah menjadi seperti semula.	Valid

4.4.9 Pengujian *Button Save*

Pada halaman edit RGB terdapat tombol *next* yang dapat digunakan oleh pengguna untuk melakukan *save*. Ketika pengguna melakukan klik pada tombol *next* maka aplikasi akan menampilkan kotak dialog yang berisikan pertanyaan konfirmasi. Skenario pengujian yang akan dilakukan adalah seperti pada tabel berikut ini:

Tabel 4.19 Skenario Pengujian
Button Save

Skenario pengujian	Hasil yang diharapkan
Pengguna melakukan klik pada tombol <i>next</i> dan melakukan <i>close</i> pada kotak dialog.	Pengguna dikembalikan ke halaman edit RGB.
Pengguna melakukan klik pada tombol <i>next</i> dan melakukan klik kembali pada tombol <i>save</i> .	System menyimpan gambar dan pengguna diarahkan kepada halaman <i>home</i> . System juga menampilkan <i>toast</i> bahwa gambar telah disimpan.

Setelah dilakukan pengujian terhadap tombol *next* dan *save* diperoleh data sebagai berikut:

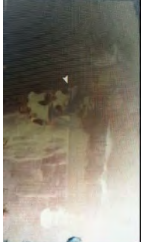
Tabel 4.20 Hasil Uji Skenario
Button Save

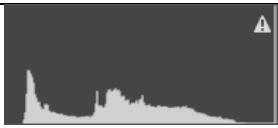
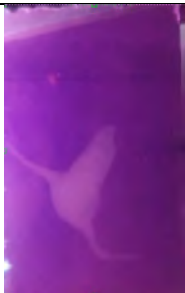
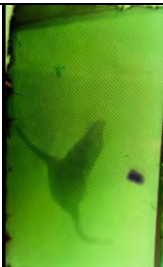
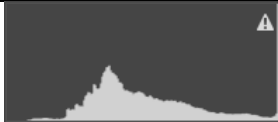
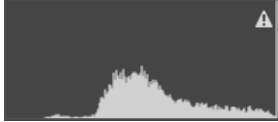
Skenario yang diuji	Hasil yang diharapkan	Hasil yang diperoleh	Kesimpulan
Pengguna melakukan klik pada tombol <i>next</i> dan melakukan <i>close</i> pada kotak dialog.	Pengguna dikembalikan ke halaman edit RGB.	Kembali ke <i>layout</i> edit RGB.	Valid
Pengguna melakukan klik pada tombol <i>next</i> dan melakukan klik kembali pada tombol <i>save</i> .	System menyimpan gambar dan pengguna diarahkan kepada halaman <i>home</i> . System juga menampilkan <i>toast</i> bahwa gambar telah disimpan.	Menyimpan gambar dengan disertakan pesan <i>toast</i> tersimpan dan kembali ke halaman <i>home</i> .	Valid

4.5 Pengujian Berdasarkan Kamera dan Histogram

Pada pengujian berdasarkan histogram terdapat beberapa jenis citra negatif dengan histogram yang berbeda-beda, pengujian ini bertujuan untuk menentukan apakah histogram pada citra negatif berpengaruh besar pada hasil konversi citra yang dilakukan atau tidak. Pada tabel di bawah merupakan hasil pengujian dengan beberapa histogram berbeda, berikut tabel hasilnya:

Tabel 4.21 Pengujian Berdasarkan Kamera dan Histogram

Jenis Kamera	Spesifikasi	Histogram	Citra Negatif	Citra Invert
Canon Mate	Lensa 35mm, F4			
				
				
Konika Pop	Lensa Pop 35mm, F4			
				

Jenis Kamera	Spesifikasi	Histogram	Citra Negatif	Citra Invert
				
Fuji Flezz	Lensa Fujinon 35mm, F4			
				
				

Kesimpulan yang dapat diambil dari pengujian diatas adalah kamera yang digunakan pada saat pengambilan gambar analog sangat berpengaruh dengan hasil yang diberikan ketika gambar akan dijadikan foto digital.

BAB V

Kesimpulan dan Saran

5.1 Kesimpulan

Setelah melakukan pengembangan terhadap aplikasi *Color It*, maka kesimpulan yang dapat diambil adalah sebagai berikut:

1. Aplikasi *Color It* terbukti dapat berjalan pada perangkat android dengan minimal menggunakan android 9. Selain itu aplikasi *Color It* juga terbukti dapat melakukan konversi dengan menggunakan algoritma *invert* citra negatif dari warna citra negatif menjadi citra warna, kemudian aplikasi juga mampu melakukan perubahan warna dengan menggunakan *slider* RGB yang telah disediakan.
2. Aplikasi *Color It* juga terbukti dapat melakukan simpan gambar sehingga pengguna dapat melihat citra dalam bentuk digital. Gambar yang tersimpan memudahkan pengguna dalam melihat hasil dan melakukan perubahan citra digital dengan aplikasi editing.

5.2 Saran

Berikut adalah beberapa saran yang penulis usulkan untuk pengembangan aplikasi selanjutnya:

1. Pengembangan fitur yang terdapat pada aplikasi dapat menggabungkan beberapa fitur editing lainnya.
2. Pengembangan aplikasi serupa yang tidak terbatas oleh perangkat yang digunakan.
3. Pengembangan aplikasi lain yang dapat memanfaatkan warna citra negatif untuk kepentingan lainnya.
4. Melakukan perbandingan terhadap biaya yang diperlukan untuk mengembangkan aplikasi serupa dibandingkan dengan mesin fisik biasa.
5. Melakukan perbandingan warna yang dihasilkan oleh aplikasi dan warna yang dihasilkan oleh alat cetak fisik.

DAFTAR PUSTAKA

- Android Developer 2021, *Temui Android Studio*, Android Developer, dilihat 12 November 2021, <<https://developer.android.com/studio/intro>>.
- Beal, V 2021, *Object Oriented Programing*, Webopedia, dilihat 21 Oktober 2021, <<https://www.webopedia.com/definitions/object-oriented-programming-oop/>>.
- Darmawan, Y. S., & Wikayanto, A 2018, *Tren Kamera Analog Instan Di Kalangan Remaja Indonesia*, Fotografi Analog, Volume 14. dilihat 11 November 2021.
- Gani, R & Kusumalestari, R, R, 2013, *Jurnalistik Foto Suatu Pengantar*, Simbiosis Rekatama Media, Bandung.
- Herpendi 2016, *Aplikasi Pengolaan Nilai Akademik Mahasiswa dan DPNA (Daftar Peserta dan Nilai Akhir)*, Jurnal Sains dan Informatika, vol.2, No.7, dilihat 26 Oktober 2021.
- Ilford Photo 2019, *Ilford Photo Global Film Users Survei*, Ilford Photo, dilihat 21 September 2021, <from <https://www.ilmordphoto.com/ilmord-photo-global-film-users-survey-the-results-are-in/>>.
- Javapoint 2021, *Sejarah Android*, Javapoint, dilihat 12 November 2021, <<https://www.javatpoint.com/android-history-and-versions>>.
- Mcgregor, L 2017, *Dasar-Dasar Grading Warna Dengan Kurva*. Premiumbeat, dilihat 8 November 2021, <<https://www.premiumbeat.com/blog/color-grading-with-curves/>>
- Muslihudin, M, & Oktafiano, 2016, *Menggunakan Model Terstruktur dan UML*, Penerbit Andi, Yogyakarta.
- Nofriadi, 2015, *Java Fundamental Dengan Netbeans 8.0.2*, DeePublish, Yogyakarta.
- Pressman, R, 2011, *Software Engineering A Practitioner's Approach, Seventh Edition*, McGraw-Hill, New York.
- Raharjo, Budi, Heryanto, I, & Haryono, A, 2012, *Mudah Belajar Java revisi kedua*, Penerbit Informatika, Bandung.
- Saputra, D, I., Pranata, T, B, & Handani, S, W 2016, *Prototype Aplikasi Pengolah Citra Invert Sebagai Media Pengolah Klise Foto*. DocPlayer, dilihat 10 November 2021, <<https://docplayer.info/34860673-Prototype-aplikasi-pengolah-citra-invert-sebagai-media-pengolah-klise-foto.html>>.
- Setiawan, R 2021, *Metode SDLC Dalam Pengembangan Software*, Dicoding, dilihat 10 November 2021, <from <https://www.dicoding.com/blog/metode-sdlc/>>.
- Sudarma, K, 2014, *Fotografi*, Graha Ilmu, Yogyakarta.
- Tsui, F, F, et.al, 2016, *Essentials of Software Engineerring*, Jones & Bartlett Learning, Boston.
- Urva, G, & Siregar, H, F 2015, *Pemodelan UML E-Marketing Minyak Goreng*, DocPlayer, dilihat 9 November 2021, <<https://docplayer.info/31270135-Pemodelan-uml-e-marketing-minyak-goreng.html>>.
- Weisfeld, M, 2013, *The Object Oriented Thought Process (4th ed.)*. Pearson Education, New York.

Ramadoni, F 2017, Apa Itu Bahasa Pemrograman Kotlin?, Teknojurnal, dilihat 15 Mei 2022, <<https://teknojurnal.com/apa-itu-bahasa-pemrograman-kotlin/>>.

LAMPIRAN

Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
package="com.basys.colorit">

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus"
/>
    <uses-feature android:name="android.hardware.camera2.full" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/logo"
        android:label="@string/app_name"
        android:roundIcon="@drawable/logo"
        android:supportsRtl="true"
        android:theme="@style/Theme.ColorIt"
        tools:targetApi="31">
        <activity
            android:name=".PreviewImage"
            android:exported="false" />
        <activity
            android:name="com.yalantis.ucrop.UCropActivity"
            android:screenOrientation="portrait"

android:theme="@style/Theme.AppCompat.Light.NoActionBar" />
        <activity
            android:name=".EditImage"
            android:exported="false" />
        <activity
            android:name=".CaptureImage"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"

/>
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            </activity>
        </application>
</manifest>
```

MainActivity.java

```
package com.basys.colorit;

import androidx.activity.result.ActivityResult;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import android.content.ActivityNotFoundException;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.ImageDecoder;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import com.basys.colorit.databinding.ActivityMainBinding;
import java.io.ByteArrayOutputStream;
import java.nio.charset.StandardCharsets;
import java.util.Random;

public class MainActivity extends AppCompatActivity {

    private ActivityMainBinding binding;
    private Bitmap imageBitmap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding =
ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        initView();
    }

    @Override
    public void onBackPressed(){

    }

    private final View.OnClickListener convert = v ->
convertImage();
    private final View.OnClickListener pick = v -> pickImage();
    private final View.OnClickListener tutorial = v ->
watchYoutubeVideo();
    private final View.OnClickListener exit = v -> finish();

    private void pickImage(){
        Intent pickIntent = new Intent();
        pickIntent.setAction(Intent.ACTION_PICK);

pickIntent.setDataAndType(MediaStore.Images.Media.EXTERNAL_CONTENT
```

```

_URI, "image/*");
    pickResult.launch(pickIntent);
}

private void watchYoutubeVideo() {
    //insert youtube video id
    Intent appIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse("vnd.youtube:" + "v4PSN726YKU"));
    Intent webIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.youtube.com/watch?v=" +
"v4PSN726YKU"));
    try {
        startActivity(appIntent);
    } catch (ActivityNotFoundException ex) {
        startActivity(webIntent);
    }
}

private void previewImage(Uri uri){
    Intent intent = new Intent(this, PreviewImage.class);
    intent.putExtra("uri", uri.toString());
    startActivity(intent);
}

private void convertImage(){
    Intent intent = new Intent(this, CaptureImage.class);
    startActivity(intent);
}

private void initView(){
    Window window = this.getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_B
ACKGROUNDS);

window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STAT
US);

window.setStatusBarColor(this.getResources().getColor(R.color.seco
ndary, this.getTheme()));
    binding.btnCapture.setOnClickListener(convert);
    binding.btnPickImage.setOnClickListener(pick);
    binding.btnExit.setOnClickListener(exit);
    binding.btnTutorial.setOnClickListener(tutorial);
}

    ActivityResultLauncher<Intent> pickResult =
registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if (result.getResultCode() == RESULT_OK){
            if (result.getData() != null) {
                previewImage(result.getData().getData());
            }
        }
    }
);
}

```

AppExt.kt

```
package com.basys.colorit

import android.app.Activity
import android.content.ContentValues
import android.content.Context
import android.content.Intent
import android.graphics.Bitmap
import android.net.Uri
import android.os.Build
import android.os.Environment
import android.provider.MediaStore
import androidx.core.content.res.ResourcesCompat
import www.sanju.motiontoast.MotionToast
import www.sanju.motiontoast.MotionToastStyle
import java.io.File
import java.io.FileOutputStream
import java.io.OutputStream

@Suppress("DEPRECATION")
fun Context.saveImageToStorage(
    bitmap: Bitmap?,
    filename: String = "screenshot.jpg",
) {
    val imageOutputStream: OutputStream
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
        val values = ContentValues().apply {
            put(MediaStore.DownloadColumns.DISPLAY_NAME, filename)
            put(MediaStore.DownloadColumns.MIME_TYPE,
                "image/jpeg")
            put(MediaStore.DownloadColumns.RELATIVE_PATH,
                "Download/COLORIT")
        }

        contentResolver.run {
            val uri =

contentResolver.insert(MediaStore.Downloads.EXTERNAL_CONTENT_URI,
values)

            ?: return
            imageOutputStream = openOutputStream(uri) ?: return
            imageOutputStream.use {
                bitmap?.compress(Bitmap.CompressFormat.JPEG, 100, it) }
            sendBroadcast(
                Intent(
                    Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, uri
                )
            )
        }
    } else {
        val fileGallery = getFileGallery()
        val image = File(fileGallery.path, filename)
        imageOutputStream = FileOutputStream(image)
        imageOutputStream.use {
            bitmap?.compress(Bitmap.CompressFormat.JPEG, 100, it) }
        sendBroadcast(
            Intent(
```

```

        Intent.ACTION_MEDIA_SCANNER_SCAN_FILE,
        Uri.fromFile(image)
    )
    }
}
fun Context.getFileGallery(): File {
    val folder: File = if (Environment.getExternalStorageState()
== Environment.MEDIA_MOUNTED) {
        File(Environment.getExternalStorageDirectory(),
"Download/COLORIT/")
    } else {
        cacheDir
    }
    if (!folder.exists()) {
        folder.mkdirs()
    }
    return folder
}
fun Activity.toastSuccess(title: String = "SUCCESS", message:
String) {
    MotionToast.createColorToast(
        this,
        title,
        message,
        MotionToastStyle.INFO,
        MotionToast.GRAVITY_BOTTOM,
        MotionToast.LONG_DURATION,
        ResourcesCompat.getFont(
            this,
            R.font.bmono_regular
        )
    )
}

```

CaptureImage.java

```
package com.basys.colorit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.SurfaceTexture;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraCaptureSession;
import android.hardware.camera2.CameraCharacteristics;
import android.hardware.camera2.CameraDevice;
import android.hardware.camera2.CameraManager;
import android.hardware.camera2.CameraMetadata;
import android.hardware.camera2.CaptureRequest;
import android.hardware.camera2.params.StreamConfigurationMap;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerThread;
import android.provider.MediaStore;
import android.util.Size;
import android.util.SparseIntArray;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Toast;
import com.basys.colorit.databinding.CaptureImageBinding;
import java.io.ByteArrayOutputStream;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.Objects;
import java.util.Random;

public class CaptureImage extends AppCompatActivity {

    CaptureImageBinding binding;

    private static final SparseIntArray ORIENTATIONS = new
    SparseIntArray();
    static {
        ORIENTATIONS.append(Surface.ROTATION_0, 90);
        ORIENTATIONS.append(Surface.ROTATION_90, 0);
        ORIENTATIONS.append(Surface.ROTATION_180, 270);
        ORIENTATIONS.append(Surface.ROTATION_270, 180);
    }
    private Bitmap imageBitmap;
    protected CameraDevice cameraDevice;
    protected CameraCaptureSession cameraCaptureSessions;
    protected CaptureRequest.Builder captureRequestBuilder;
    private Size imageDimension;
    private static final int REQUEST_CAMERA_PERMISSION = 200;
```

```

private Handler mBackgroundHandler;
private HandlerThread mBackgroundThread;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding =
CaptureImageBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());
    setSupportActionBar(binding.toolbar);

Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(false);
    initView();
}

private final View.OnClickListener back = v ->
onBackPressed();

private void initView(){
    Window window = this.getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);

window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS);

window.setStatusBarColor(this.getResources().getColor(R.color.secondary, this.getTheme()));
    binding.toolbar.setNavigationOnClickListener(back);

binding.textureView.setSurfaceTextureListener(textureListener);
    binding.btnCapture.setOnClickListener(capture);
    binding.btnRecapture.setOnClickListener(recapture);
    binding.btnNext.setOnClickListener(next);
}

private final View.OnClickListener capture = v ->
captureImage();
private final View.OnClickListener recapture = v ->
recaptureImage();
private final View.OnClickListener next = v -> sentImage();

private Uri getImageUri(Context inContext, Bitmap inImage) {
    byte[] array = new byte[7]; // length is bounded by 7
    new Random().nextBytes(array);
    String generatedString = new String(array,
StandardCharsets.UTF_8);
    ByteArrayOutputStream bytes = new ByteArrayOutputStream();
    inImage.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
    String path =
MediaStore.Images.Media.insertImage(inContext.getContentResolver(),
, inImage, generatedString, null);
    return Uri.parse(path);
}

private void sentImage(){
    Uri uri = getImageUri(this, imageBitmap);

```

```

        Intent intent = new Intent(this, EditImage.class);
        intent.putExtra("uri", uri.toString());
        startActivity(intent);
    }

    private void recaptureImage(){
        imageBitmap = null;
        binding.imageResult.setVisibility(View.GONE);
        binding.textureView.setVisibility(View.VISIBLE);
        binding.btnCapture.setVisibility(View.VISIBLE);
        binding.btnNext.setVisibility(View.INVISIBLE);
        binding.btnRecapture.setVisibility(View.INVISIBLE);
    }

    private void captureImage(){
        imageBitmap = binding.textureView.getBitmap();
        binding.imageResult.setImageBitmap(imageBitmap);
        binding.imageResult.setVisibility(View.VISIBLE);
        binding.textureView.setVisibility(View.GONE);
        binding.btnCapture.setVisibility(View.INVISIBLE);
        binding.btnNext.setVisibility(View.VISIBLE);
        binding.btnRecapture.setVisibility(View.VISIBLE);
    }

    private final CameraDevice.StateCallback stateCallback = new
    CameraDevice.StateCallback() {
        @Override
        public void onOpened(CameraDevice camera) {
            //This is called when the camera is open
            cameraDevice = camera;
            createCameraPreview();
        }
        @Override
        public void onDisconnected(CameraDevice camera) {
            cameraDevice.close();
        }
        @Override
        public void onError(CameraDevice camera, int error) {
            cameraDevice.close();
            cameraDevice = null;
        }
    };

    protected void startBackgroundThread() {
        mBackgroundThread = new HandlerThread("Camera
Background");
        mBackgroundThread.start();
        mBackgroundHandler = new
Handler(mBackgroundThread.getLooper());
    }

    protected void stopBackgroundThread() {
        mBackgroundThread.quitSafely();
        try {
            mBackgroundThread.join();
            mBackgroundThread = null;
            mBackgroundHandler = null;
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

```

```

    }
}

TextureView.SurfaceTextureListener textureListener = new
TextureView.SurfaceTextureListener() {

    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture
surface, int width, int height) {
        //open your camera here
        openCamera();
    }
    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture
surface, int width, int height) {
        // Transform you image captured size according to the
surface width and height
    }
    @Override
    public boolean onSurfaceTextureDestroyed(SurfaceTexture
surface) {
        return false;
    }
    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture
surface) {
    }
};

protected void createCameraPreview() {
    try {
        SurfaceTexture texture =
binding.textureView.getSurfaceTexture();
        assert texture != null;

        texture.setDefaultBufferSize(imageDimension.getWidth(),
imageDimension.getHeight());
        //Log.e(TAG, "Width: " +
String.valueOf(imageDimension.getWidth()) + "| Height: " +
String.valueOf(imageDimension.getHeight()));
        Surface surface = new Surface(texture);
        captureRequestBuilder =
cameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
        captureRequestBuilder.addTarget(surface);
        cameraDevice.createCaptureSession(List.of(surface),
new CameraCaptureSession.StateCallback(){
            @Override
            public void onConfigured(@NonNull
CameraCaptureSession cameraCaptureSession) {
                //The camera is already closed
                if (null == cameraDevice) {
                    return;
                }
                // When the session is ready, we start
displaying the preview.
                cameraCaptureSessions = cameraCaptureSession;
                updatePreview();
            }
        }, null);
    } catch (CameraAccessException e) {
        // Camera is not found among the other components
    }
}
}

```

```

        public void onConfigureFailed(@NonNull
CameraCaptureSession cameraCaptureSession) {
            Toast.makeText(CaptureImage.this,
"Configuration change", Toast.LENGTH_SHORT).show();
        }
    }, null);
} catch (CameraAccessException e) {
    e.printStackTrace();
}
}

private void openCamera() {
    CameraManager manager = (CameraManager)
getSystemService(Context.CAMERA_SERVICE);
    try {
        String cameraId = manager.getCameraIdList()[0];
        CameraCharacteristics characteristics =
manager.getCameraCharacteristics(cameraId);
        StreamConfigurationMap map =
characteristics.get(CameraCharacteristics.SCALER_STREAM_CONFIGURAT
ION_MAP);
        assert map != null;
        imageDimension =
map.getOutputSizes(SurfaceTexture.class)[0];
        // Add permission for camera and let user grant the
permission
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(CaptureImage.this, new
String[]{Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE},
REQUEST_CAMERA_PERMISSION);
            return;
        }
        manager.openCamera(cameraId, stateCallback, null);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}

protected void updatePreview() {
    captureRequestBuilder.set(CaptureRequest.CONTROL_MODE,
CameraMetadata.CONTROL_MODE_AUTO);

    captureRequestBuilder.set(CaptureRequest.CONTROL_EFFECT_MODE,
CaptureRequest.COLOR_CORRECTION_MODE_HIGH_QUALITY);
    try {

cameraCaptureSessions.setRepeatingRequest(captureRequestBuilder.bu
ild(), null, mBackgroundHandler);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}
}

```

```

        @Override
        public void onRequestPermissionsResult(int requestCode,
        @NonNull String[] permissions, @NonNull int[] grantResults) {
            if (requestCode == REQUEST_CAMERA_PERMISSION) {
                if (grantResults[0] ==
PackageManager.PERMISSION_DENIED) {
                    // close the app
                    Toast.makeText(CaptureImage.this, "Sorry!!!, you
can't use this app without granting permission",
Toast.LENGTH_LONG).show();
                    finish();
                }
            }
            super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        }

        @Override
        protected void onResume() {
            super.onResume();
            startBackgroundThread();
            if (binding.textureView.isAvailable()) {
                openCamera();
            } else {

binding.textureView.setSurfaceTextureListener(textureListener);
            }
        }

        @Override
        protected void onPause() {
            stopBackgroundThread();
            super.onPause();
        }
    }
}

```

ConfirmDialog.java

```
package com.basys.colorit;

import android.app.Dialog;
import android.content.Context;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.view.Gravity;
import android.view.ViewGroup;
import android.view.Window;
import com.basys.colorit.databinding.ConfirmDialogBinding;

public class ConfirmDialog extends Dialog {

    public ConfirmDialog(Context context, final Callback<Boolean>
callback){
        super(context);
        getWindow().setWindowAnimations(R.style.DialogFromBottom);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        com.basys.colorit.databinding.ConfirmDialogBinding binding
= ConfirmDialogBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        getWindow().setLayout(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
        getWindow().setGravity(Gravity.BOTTOM);
        setCancelable(true);
        setCanceledOnTouchOutside(true);
        binding.btnOk.setOnClickListener(view -> {
            dismiss();
            callback.onFinish(true);
        });
    }
}
```

EditImage.java

```
package com.basys.colorit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.res.ResourcesCompat;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.ColorFilter;
import android.graphics.ImageDecoder;
import android.graphics.LightingColorFilter;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.SeekBar;
import android.widget.Toast;
import com.basys.colorit.databinding.EditImageBinding;
import com.google.android.material.slider.Slider;
import com.yalantis.ucrop.UCrop;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.nio.charset.StandardCharsets;
import java.util.Objects;
import java.util.Random;

public class EditImage extends AppCompatActivity {

    EditImageBinding binding;
    Bitmap imageBitmap;
    BitmapDrawable original;
    Bitmap cropped;
    Uri imageUri;
    private Double alpha = 0.0;
    private Double red = 0.0;
    private Double green = 0.0;
    private Double blue = 0.0;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        binding = EditImageBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        setSupportActionBar(binding.toolbar);

        Objects.requireNonNull(getSupportActionBar()).setDisplayShowTitleEnabled(false);
```

```

        setImageUri();
        setImageBitmap(imageUri);
        initView();
        initSlider();
        initSeekBar();
        original = (BitmapDrawable)
binding.imageResult.getDrawable();
    }

    private final View.OnClickListener back = v ->
onBackPressed();
    private final View.OnClickListener showBright = v ->
setBrightness();
    private final View.OnClickListener showContrast = v ->
setContrast();
    private final View.OnClickListener showRGB = v -> setRGB();
    private final View.OnClickListener save = v ->
showDialogPrompt();
    private final View.OnClickListener crop = v -> cropImage();

    private void setImageUri(){
        imageUri = Uri.parse(getIntent().getStringExtra("uri"));
    }

    private void setImageBitmap(Uri imageUri){
        ContentResolver contentResolver = getContentResolver();
        try {
            if(Build.VERSION.SDK_INT < 28) {
                imageBitmap =
MediaStore.Images.Media.getBitmap(contentResolver, imageUri);
            } else {
                ImageDecoder.Source source =
ImageDecoder.createSource(contentResolver, imageUri);
                imageBitmap = ImageDecoder.decodeBitmap(source);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private Bitmap getImageBitmap(Uri imageUri){
        ContentResolver contentResolver = getContentResolver();
        Bitmap bitmap = null;
        try {
            if(Build.VERSION.SDK_INT < 28) {
                bitmap =
MediaStore.Images.Media.getBitmap(contentResolver, imageUri);
            } else {
                ImageDecoder.Source source =
ImageDecoder.createSource(contentResolver, imageUri);
                bitmap = ImageDecoder.decodeBitmap(source);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bitmap;
    }
}

```

```

        private void cropImage(){
            Drawable drawable = binding.imageResult.getDrawable();
            Bitmap toCrop = ((BitmapDrawable) drawable).getBitmap();
            Uri uri = getImageUri(this, toCrop);
            String destinationUri = "SAMPLE_CROPPED_IMAGE_NAME.jpg";
            UCrop.of(uri, Uri.fromFile(new File(getCacheDir(),
destinationUri)))
                .start(this);
        }

        private void initView(){
            Window window = this.getWindow();

            window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_B
ACKGROUNDS);

            window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STAT
US);

            window.setStatusBarColor(this.getResources().getColor(R.color.seco
ndary, this.getTheme()));

            binding.imageResult.setImageURI(imageUri);
            binding.toolbar.setNavigationOnClickListener(back);
            binding.btnSave.setOnClickListener(save);
            binding.btnBrightness.setOnClickListener(showBright);
            binding.btnRGB.setOnClickListener(showRGB);
            binding.btnContrast.setOnClickListener(showContrast);
            binding.btnCrop.setOnClickListener(crop);
        }

        private void setBrightness(){
            binding.contrastContainer.setVisibility(View.GONE);
            binding.brightnessContainer.setVisibility(View.VISIBLE);
            binding.RGBContainer.setVisibility(View.GONE);
        }

        private void setContrast(){
            binding.contrastContainer.setVisibility(View.VISIBLE);
            binding.brightnessContainer.setVisibility(View.GONE);
            binding.RGBContainer.setVisibility(View.GONE);
        }

        private void setRGB(){
            binding.contrastContainer.setVisibility(View.GONE);
            binding.brightnessContainer.setVisibility(View.GONE);
            binding.RGBContainer.setVisibility(View.VISIBLE);
        }

        private Uri getImageUri(Context inContext, Bitmap inImage) {
            byte[] array = new byte[7]; // length is bounded by 7
            new Random().nextBytes(array);
            String generatedString = new String(array,
StandardCharsets.UTF_8);
            ByteArrayOutputStream bytes = new ByteArrayOutputStream();
            inImage.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
            String path =
MediaStore.Images.Media.insertImage(inContext.getContentResolver()
, inImage, generatedString, null);

```

```

        return Uri.parse(path);
    }

    private void showDialogPrompt() {
        byte[] array = new byte[7]; // length is bounded by 7
        new Random().nextBytes(array);
        String generatedString = new String(array,
StandardCharsets.UTF_8);
        ConfirmDialog confirmDialog = new ConfirmDialog(this,
object -> {
            if (object) {

saveImage(getScreenShotFromView(binding.imageResult),
generatedString);
            }
        });
        confirmDialog.show();
    }

    private Bitmap getScreenShotFromView(View v){
        Bitmap screenshot = null;
        try{
            screenshot = Bitmap.createBitmap(v.getMeasuredWidth(),
v.getMeasuredHeight(), Bitmap.Config.ARGB_8888);
            Canvas canvas = new Canvas(screenshot);
            v.draw(canvas);
        } catch (Exception e){
            Log.e("error", e.toString());
        }
        return screenshot;
    }

    private void saveImage(Bitmap bitmap, String name){
        runOnUiThread(() ->
AppExtKt.saveImageToStorage(getApplicationContext(), bitmap,
name));
        AppExtKt.toastSuccess(this, "SUCCESS", "Saved On
Download/COLORIT");
        Intent intent = new Intent(this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(intent);
        finish();
    }

    private void initSeekBar(){
        binding.sliderBrightness.addOnSliderTouchListener(new
Slider.OnSliderTouchListener() {
            @Override
            public void onStartTrackingTouch(@NonNull Slider
slider) {
            }

            @Override
            public void onStopTrackingTouch(@NonNull Slider
slider) {
                adjustBrightness((int) slider.getValue());
            }
        });
        binding.sliderContrast.addOnSliderTouchListener(new

```

```

Slider.OnSliderTouchListener() {
    @Override
    public void onStartTrackingTouch(@NonNull Slider
slider) {

        }

    @Override
    public void onStopTrackingTouch(@NonNull Slider
slider) {
        adjustContrast((int) slider.getValue());
    }
});

private void adjustContrast(int i){
    Bitmap bmp = original.getBitmap();
    if (cropped != null){
        bmp = cropped;
    }

    String initialHex = Tool.hexScale()[i];
    String initialMul = "0X" + initialHex + initialHex +
initialHex;
    int mul = Integer.decode(initialMul);
    int add = 0X000000;

    Bitmap output = Bitmap.createScaledBitmap(bmp,
bmp.getWidth(), bmp.getHeight(),
false).copy(Bitmap.Config.ARGB_8888, true);
    Paint paint = new Paint();
    ColorFilter filter = new LightingColorFilter(mul, add);
    paint.setColorFilter(filter);

    Canvas canvas = new Canvas(output);
    canvas.drawBitmap(output, 0, 0, paint);

    binding.imageResult.setImageBitmap(output);
}

private void adjustBrightness(int i){
    Bitmap bmp = original.getBitmap();
    if (cropped != null){
        bmp = cropped;
    }

    final int mul = 0XFFFFFF;

    String initialHex = Tool.hexScale()[i];
    String initialAdd = "0X"+initialHex + initialHex +
initialHex;
    int add = Integer.decode(initialAdd);

    Bitmap output = Bitmap.createScaledBitmap(bmp,
bmp.getWidth(), bmp.getHeight(),
false).copy(Bitmap.Config.ARGB_8888, true);
    Paint paint = new Paint();
    ColorFilter filter = new LightingColorFilter(mul, add);
    paint.setColorFilter(filter);

```

```

        Canvas canvas = new Canvas(output);
        canvas.drawBitmap(output, 0, 0, paint);

        binding.imageResult.setImageBitmap(output);
    }

    private void initSlider(){
        binding.sliderBlue.addOnSliderTouchListener(new
Slider.OnSliderTouchListener() {
            @Override
            public void onStartTrackingTouch(@NonNull Slider
slider) {

            }

            @Override
            public void onStopTrackingTouch(@NonNull Slider
slider) {
                blue = (double) slider.getValue();
                if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.O) {

binding.imageResult.setColorFilter(Color.argb(alpha.intValue(),
red.intValue(), green.intValue(), blue.intValue()),
PorterDuff.Mode.MULTIPLY);
                }
            }
        });
        binding.sliderAlpha.addOnSliderTouchListener(new
Slider.OnSliderTouchListener() {
            @Override
            public void onStartTrackingTouch(@NonNull Slider
slider) {

            }

            @Override
            public void onStopTrackingTouch(@NonNull Slider
slider) {
                alpha = (double) slider.getValue();
                if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.O) {

binding.imageResult.setColorFilter(Color.argb(alpha.intValue(),
red.intValue(), green.intValue(), blue.intValue()),
PorterDuff.Mode.MULTIPLY);
                }
            }
        });
        binding.sliderGreen.addOnSliderTouchListener(new
Slider.OnSliderTouchListener() {
            @Override
            public void onStartTrackingTouch(@NonNull Slider
slider) {

            }
        }
    }

```

```

        @Override
        public void onStopTrackingTouch(@NonNull Slider
slider) {
            green = (double) slider.getValue();
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.O) {

binding.imageResult.setColorFilter(Color.argb(alpha.intValue(),
red.intValue(), green.intValue(), blue.intValue()),
PorterDuff.Mode.MULTIPLY);
            }
        }
    });
    binding.sliderRed.addOnSliderTouchListener(new
Slider.OnSliderTouchListener() {
        @Override
        public void onStartTrackingTouch(@NonNull Slider
slider) {

        }

        @Override
        public void onStopTrackingTouch(@NonNull Slider
slider) {
            red = (double) slider.getValue();
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.O) {

binding.imageResult.setColorFilter(Color.argb(alpha.intValue(),
red.intValue(), green.intValue(), blue.intValue()),
PorterDuff.Mode.MULTIPLY);
            }
        }
    });
}

    @Override
    public void onActivityResult(int requestCode, int resultCode,
Intent data) {
        if (resultCode == RESULT_OK && requestCode ==
UCrop.REQUEST_CROP){
//            imageUri = UCrop.getOutput(data);
            cropped = getImageBitmap(UCrop.getOutput(data));
            binding.imageResult.setImageBitmap(cropped);
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
}

```

PathUtil.java

```
package com.basys.colorit;
import android.content.ContentResolver;
import android.content.ContentUris;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.os.Build;
import android.os.Environment;
import android.provider.DocumentsContract;
import android.provider.MediaStore;
import android.webkit.MimeTypeMap;

public class PathUtil {

    public Context mContext;

    public PathUtil(Context context)
    {
        this.mContext = context;
    }

    public String getRealPathFromUri(final Uri uri) {
        // DocumentProvider
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(mContext, uri)) {
            // ExternalStorageProvider
            if (isExternalStorageDocument(uri)) {
                final String docId =
DocumentsContract.getDocumentId(uri);
                final String[] split = docId.split(":");
                final String type = split[0];

                if ("primary".equalsIgnoreCase(type)) {
                    return
Environment.getExternalStorageDirectory() + "/" + split[1];
                }
            }
            // DownloadsProvider
            else if (isDownloadsDocument(uri)) {

                final String id =
DocumentsContract.getDocumentId(uri);
                final Uri contentUri = ContentUris.withAppendedId(
Uri.parse("content://downloads/public_downloads"),
Long.valueOf(id));

                return getDataColumn(mContext, contentUri, null,
null);
            }
            // MediaProvider
            else if (isMediaDocument(uri)) {
                final String docId =
DocumentsContract.getDocumentId(uri);
                final String[] split = docId.split(":");
                final String type = split[0];
```

```

        Uri contentUri = null;
        if ("image".equals(type)) {
            contentUri =
MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
        } else if ("video".equals(type)) {
            contentUri =
MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
        } else if ("audio".equals(type)) {
            contentUri =
MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        }

        final String selection = "_id=?";
        final String[] selectionArgs = new String[]{
            split[1]
        };

        return getDataColumn(mContext, contentUri,
selection, selectionArgs);
    }
    // MediaStore (and general)
    else if ("content".equalsIgnoreCase(uri.getScheme())) {

        // Return the remote address
        if (isGooglePhotosUri(uri))
            return uri.getLastPathSegment();

        return getDataColumn(mContext, uri, null, null);
    }
    // File
    else if ("file".equalsIgnoreCase(uri.getScheme())) {
        return uri.getPath();
    }

    return null;
}

private String getDataColumn(Context context, Uri uri, String
selection,
                                String[] selectionArgs) {

    Cursor cursor = null;
    final String column = "_data";
    final String[] projection = {
        column
    };

    try {
        cursor = context.getContentResolver().query(uri,
projection, selection, selectionArgs,
            null);
        if (cursor != null && cursor.moveToFirst()) {
            final int index =
cursor.getColumnIndexOrThrow(column);
            return cursor.getString(index);
        }
    } finally {
        if (cursor != null)

```

```

        cursor.close();
    }
    return null;
}

public String getMimeType(Uri uri) {
    String mimeType = null;
    if
(ContentResolver.SCHEME_CONTENT.equals(uri.getScheme())) {
        ContentResolver cr = mContext.getContentResolver();
        mimeType = cr.getType(uri);
    } else {
        String fileExtension =
MimeTypeMap.getFileExtensionFromUrl(uri
        .toString());
        mimeType =
MimeTypeMap.getSingleton().getMimeTypeFromExtension(
        fileExtension.toLowerCase());
    }
    return mimeType;
}

private boolean isExternalStorageDocument(Uri uri) {
    return
"com.android.externalstorage.documents".equals(uri.getAuthority())
;
}

private boolean isDownloadsDocument(Uri uri) {
    return
"com.android.providers.downloads.documents".equals(uri.getAuthorit
y());
}

private boolean isMediaDocument(Uri uri) {
    return
"com.android.providers.media.documents".equals(uri.getAuthority())
;
}

private boolean isGooglePhotosUri(Uri uri) {
    return
"com.google.android.apps.photos.content".equals(uri.getAuthority()
);
}
}

```

PreviewImage.java

```
package com.basys.colorit;

import androidx.appcompat.app.AppCompatActivity;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.ImageDecoder;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import com.basys.colorit.databinding.PreviewImageBinding;
import java.io.ByteArrayOutputStream;
import java.nio.charset.StandardCharsets;
import java.util.Random;

public class PreviewImage extends AppCompatActivity {

    PreviewImageBinding binding;
    Uri imageUri;
    Bitmap imageBitmap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding =
PreviewImageBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        setImageUri();
        initView();
    }

    private final View.OnClickListener invert = v -> getBitmap();
    private final View.OnClickListener next = v -> editImage();

    private void editImage(){
        Drawable image = binding.imageResult.getDrawable();
        Bitmap bitmap = ((BitmapDrawable) image).getBitmap();
        Uri imageUri = getImageUri(this, bitmap);
        Intent intent = new Intent(this, EditImage.class);
        intent.putExtra("uri", imageUri.toString());
        startActivity(intent);
    }

    private Uri getImageUri(Context inContext, Bitmap inImage) {
        byte[] array = new byte[7]; // length is bounded by 7
        new Random().nextBytes(array);
        String generatedString = new String(array,
StandardCharsets.UTF_8);
        ByteArrayOutputStream bytes = new ByteArrayOutputStream();
        inImage.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
        String path =
```

```

MediaStore.Images.Media.insertImage(inContext.getContentResolver()
, inImage, generatedString, null);
    return Uri.parse(path);
}

private void getBitmap(){
    Drawable image = binding.imageResult.getDrawable();
    Bitmap bitmap = ((BitmapDrawable) image).getBitmap();
    Bitmap invertedImage = invertImage(bitmap);
    binding.imageResult.setImageBitmap(invertedImage);
    binding.btnNext.setVisibility(View.VISIBLE);
    binding.btnConvert.setVisibility(View.GONE);
}

public static Bitmap invertImage(Bitmap original){
    Bitmap finalImage =
Bitmap.createBitmap(original.getWidth(), original.getHeight(),
Bitmap.Config.ARGB_8888);
    int A, R, G, B;
    int pixelColor;
    int height = original.getHeight();
    int width = original.getWidth();

    for (int y = 0; y<height; y++){
        for (int x = 0; x<width; x++){
            pixelColor = original.getPixel(x, y);
            A = Color.alpha(pixelColor);
            R = 255-Color.red(pixelColor);
            G = 255-Color.green(pixelColor);
            B = 255-Color.blue(pixelColor);

            finalImage.setPixel(x,y, Color.argb(A,R,G,B));
        }
    }
    return finalImage;
}

private void initView(){
    binding.imageResult.setImageURI(imageUri);
    binding.btnConvert.setOnClickListener(invert);
    binding.btnNext.setOnClickListener(next);
}

private void setImageUri(){
    imageUri = Uri.parse(getIntent().getStringExtra("uri"));
}

}

```

HexScale.java

```
package com.basys.colorit;

public class Tool {
    public static String[] hexScale(){
        return new String[]{
            "00", "01", "02", "03", "04", "05", "06", "07",
            "08", "09", "0A", "0B", "0C", "0D", "0E", "0F",
            "10", "11", "12", "13", "14", "15", "16", "17",
            "18", "19", "1A", "1B", "1C", "1D", "1E", "1F",
            "20", "21", "22", "23", "24", "25", "26", "27",
            "28", "29", "2A", "2B", "2C", "2D", "2E", "2F",
            "30", "31", "32", "33", "34", "35", "36", "37",
            "38", "39", "3A", "3B", "3C", "3D", "3E", "3F",
            "40", "41", "42", "43", "44", "45", "46", "47",
            "48", "49", "4A", "4B", "4C", "4D", "4E", "4F",
            "50", "51", "52", "53", "54", "55", "56", "57",
            "58", "59", "5A", "5B", "5C", "5D", "5E", "5F",
            "60", "61", "62", "63", "64", "65", "66", "67",
            "68", "69", "6A", "6B", "6C", "6D", "6E", "6F",
            "70", "71", "72", "73", "74", "75", "76", "77",
            "78", "79", "7A", "7B", "7C", "7D", "7E", "7F",
            "80", "81", "82", "83", "84", "85", "86", "87",
            "88", "89", "8A", "8B", "8C", "8D", "8E", "8F",
            "90", "91", "92", "93", "94", "95", "96", "97",
            "98", "99", "9A", "9B", "9C", "9D", "9E", "9F",
            "A0", "A1", "A2", "A3", "A4", "A5", "A6", "A7",
            "A8", "A9", "AA", "AB", "AC", "AD", "AE", "AF",
            "B0", "B1", "B2", "B3", "B4", "B5", "B6", "B7",
            "B8", "B9", "BA", "BB", "BC", "BD", "BE", "BF",
            "C0", "C1", "C2", "C3", "C4", "C5", "C6", "C7",
            "C8", "C9", "CA", "CB", "CC", "CD", "CE", "CF",
            "D0", "D1", "D2", "D3", "D4", "D5", "D6", "D7",
            "D8", "D9", "DA", "DB", "DC", "DD", "DE", "DF",
            "E0", "E1", "E2", "E3", "E4", "E5", "E6", "E7",
            "E8", "E9", "EA", "EB", "EC", "ED", "EE", "EF",
            "F0", "F1", "F2", "F3", "F4", "F5", "F6", "F7",
            "F8", "F9", "FA", "FB", "FC", "FD", "FE", "FF",
        };
    }
}
```