FaPa FlipBook
Extender

Aplikasi
FaPa FlipBook Extender

Dikembangkan Oleh:

Dr. Andriyani, M.Si.

Dr. Edhy Susatya

Dr. Achadi Budi Santosa, M.Pd.

Mochammad Hamsyi, S.Kom

UNIVERSITAS AHMAD DAHLAN

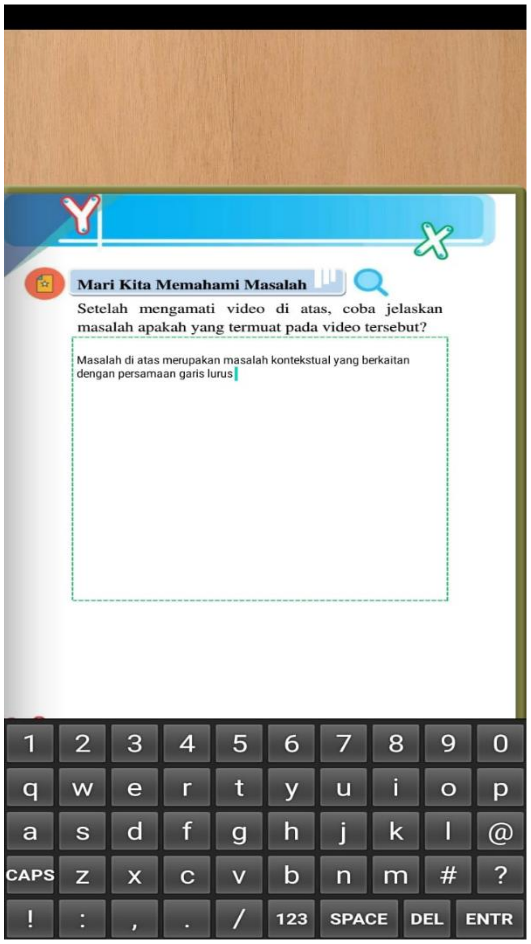
Aplikasi FaPa FlipBook Extender

Aplikasi FaPa FlipBook Extender merupakan aplikasi berbasis android yang menguraikan serta menerjemahkan objek-objek dan instruksi yang disusun oleh pengguna dengan menggunakan skrip JSON (*JavaScript Object Notation*) dengan tujuan untuk menambahkan fitur-fitur dari keluaran (*output*) suatu *Flipbook Creator*. Fitur yang dimaksud berupa fitur menulis teks, simbol matematis sederhana, *screenshoot* foto, menampilkan dan menyembunyikan informasi tertentu, menggambar kurva lurus, memutar video dan mengirimkan pesan. Aplikasi ini berguna untuk penyusunan ataupun pengembangan *flipbook*, khususnya sebagai media pembelajaran berbasis android yang memuat materi dan video pembelajaran, lembar kerja atau penugasan, penilaian, umpan balik, serta pengiriman hasil belajar.

Mekanisme Penggunaan Aplikasi FaPa FlipBook Extender

Sementara ini, aplikasi **FaPa FlipBook Extender** tidak dapat di download oleh calon pengguna secara bebas di *Google Play Store*, tapi dibagikan di link khusus *google drive* oleh pengembang. Setelah mendownload, maka pengguna dapat melakukan perubahan atau penyesuaian fitur/fungsi tertentu sesuai dengan kebutuhannya (meng-custom), baik itu fitur menulis teks, simbol matematis sederhana, *screenshoot* foto, menampilkan dan menyembunyikan informasi tertentu, menggambar kurva lurus, memutar video maupun fitur mengirimkan pesan. Fitur-fitur yang dapat digunakan oleh pengguna untuk menyusun ataupun mengembangan *flipbook* sebagai media pembelajaran memuat komponen-komponen: menulis teks, simbol matematis sederhana, *screenshoot* foto, menampilkan dan menyembunyikan informasi tertentu, menggambar kurva lurus, memutar video serta komponen mengirimkan pesan. Contoh implementasi masing-masing fitur pada media pembelajaran tersaji pada gambar-gambar berikut.

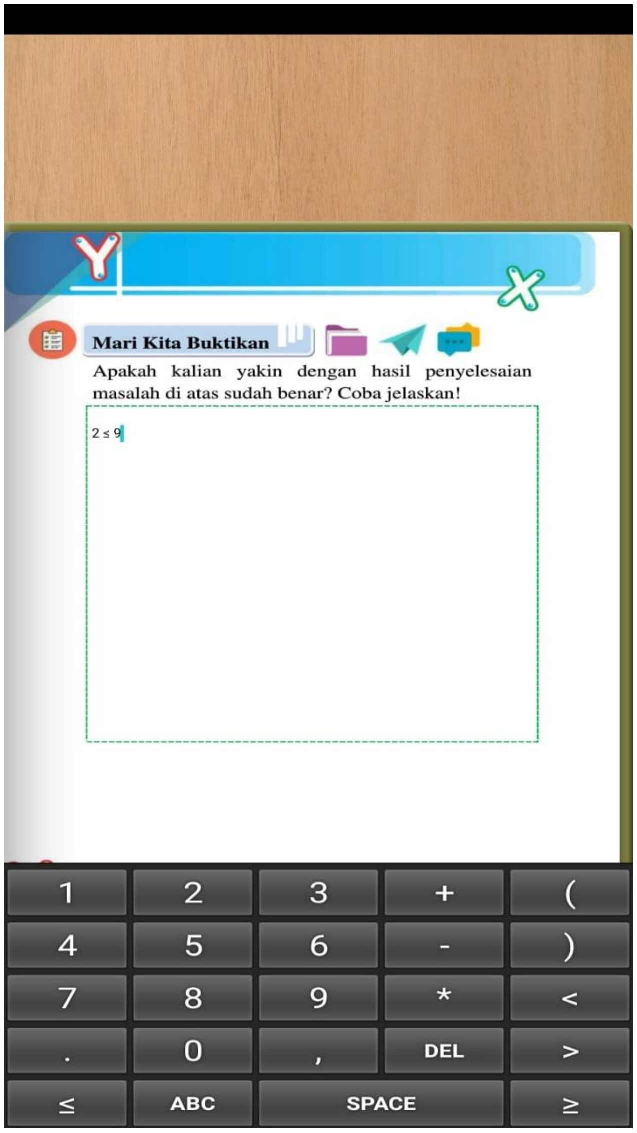
Menulis Teks


Mari Kita Memahami Masalah

Setelah mengamati video di atas, coba jelaskan masalah apakah yang termuat pada video tersebut?

Masalah di atas merupakan masalah kontekstual yang berkaitan dengan persamaan garis lurus

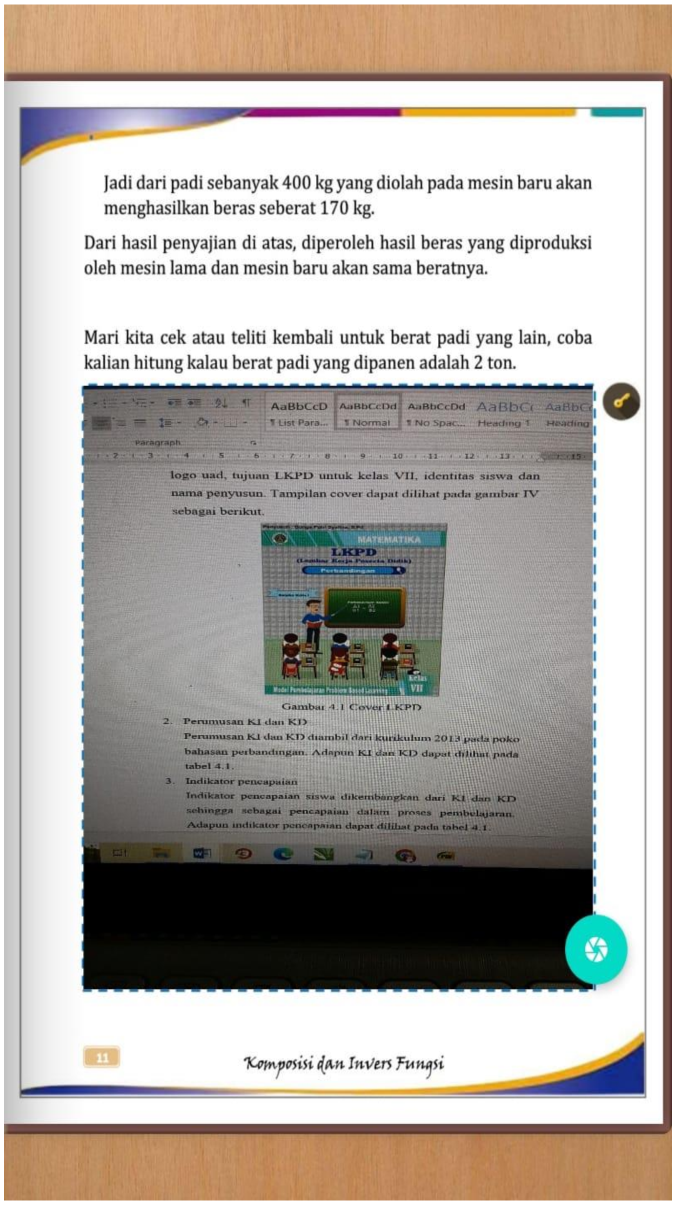
Menulis Simbol Matematis Sederhana


Mari Kita Buktikan

Apakah kalian yakin dengan hasil penyelesaian masalah di atas sudah benar? Coba jelaskan!

$2 \leq 9$

Screenshoot Foto



Jadi dari padi sebanyak 400 kg yang diolah pada mesin baru akan menghasilkan beras seberat 170 kg.

Dari hasil penyajian di atas, diperoleh hasil beras yang diproduksi oleh mesin lama dan mesin baru akan sama beratnya.

Mari kita cek atau teliti kembali untuk berat padi yang lain, coba kalian hitung kalau berat padi yang dipanen adalah 2 ton.

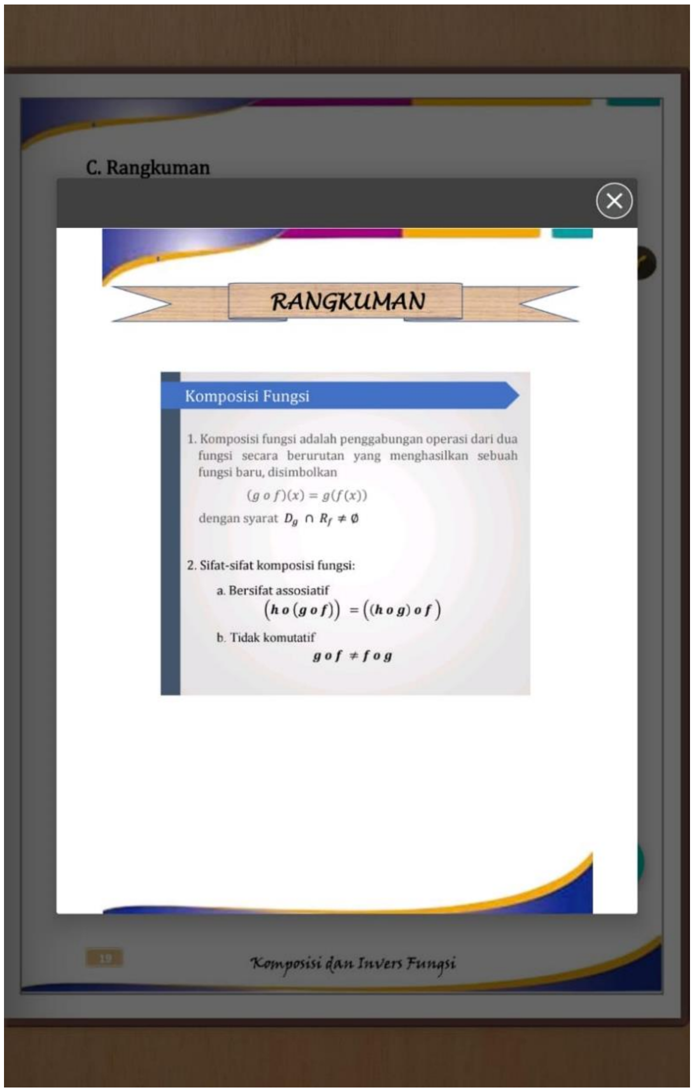
logo uad, tujuan LKPD untuk kelas VII, identitas siswa dan nama penyusun. Tampilan cover dapat dilihat pada gambar IV sebagai berikut.

Gambar 4.1 Cover LKPD

2. Perumusan KI dan KD
   Perumusan KI dan KD diambil dari kurikulum 2013 pada poko bahasan perbandingan. Adapun KI dan KD dapat dilihat pada tabel 4.1.
3. Indikator pencapaian
   Indikator pencapaian siswa dikembangkan dari KI dan KD sehingga sebagai pencapaian dalam proses pembelajaran. Adapun indikator pencapaian dapat dilihat pada tabel 4.1.

11 Komposisi dan Invers Fungsi

Menampilkan Dan Menyembunyikan Informasi Tertentu



C. Rangkuman

RANGKUMAN

Komposisi Fungsi

1. Komposisi fungsi adalah penggabungan operasi dari dua fungsi secara berurutan yang menghasilkan sebuah fungsi baru, disimbolkan

$$(g \circ f)(x) = g(f(x))$$

dengan syarat $D_g \cap R_f \neq \emptyset$

2. Sifat-sifat komposisi fungsi:

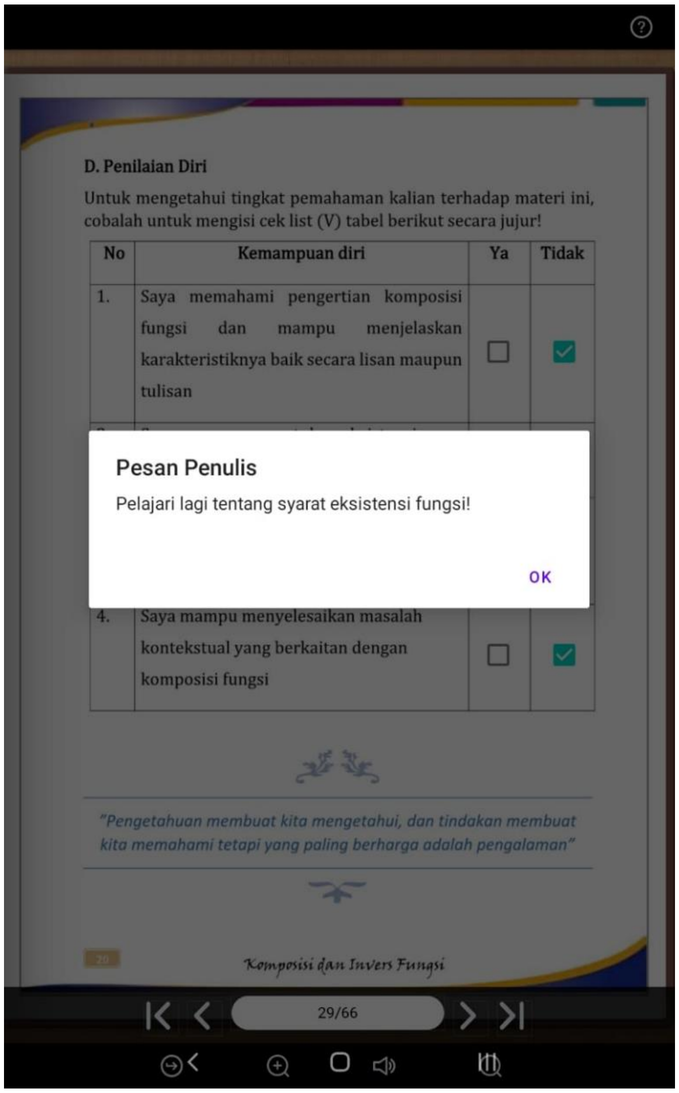
a. Bersifat assosiatif

$$\left(h \circ (g \circ f)\right) = \left((h \circ g) \circ f\right)$$

b. Tidak komutatif

$$g \circ f \neq f \circ g$$

19 Komposisi dan Invers Fungsi



D. Penilaian Diri

Untuk mengetahui tingkat pemahaman kalian terhadap materi ini, cobalah untuk mengisi cek list (V) tabel berikut secara jujur!

No	Kemampuan diri	Ya	Tidak
1.	Saya memahami pengertian komposisi fungsi dan mampu menjelaskan karakteristiknya baik secara lisan maupun tulisan	☐	✓
4.	Saya mampu menyelesaikan masalah kontekstual yang berkaitan dengan komposisi fungsi	☐	✓

Pesan Penulis

Pelajari lagi tentang syarat eksistensi fungsi!

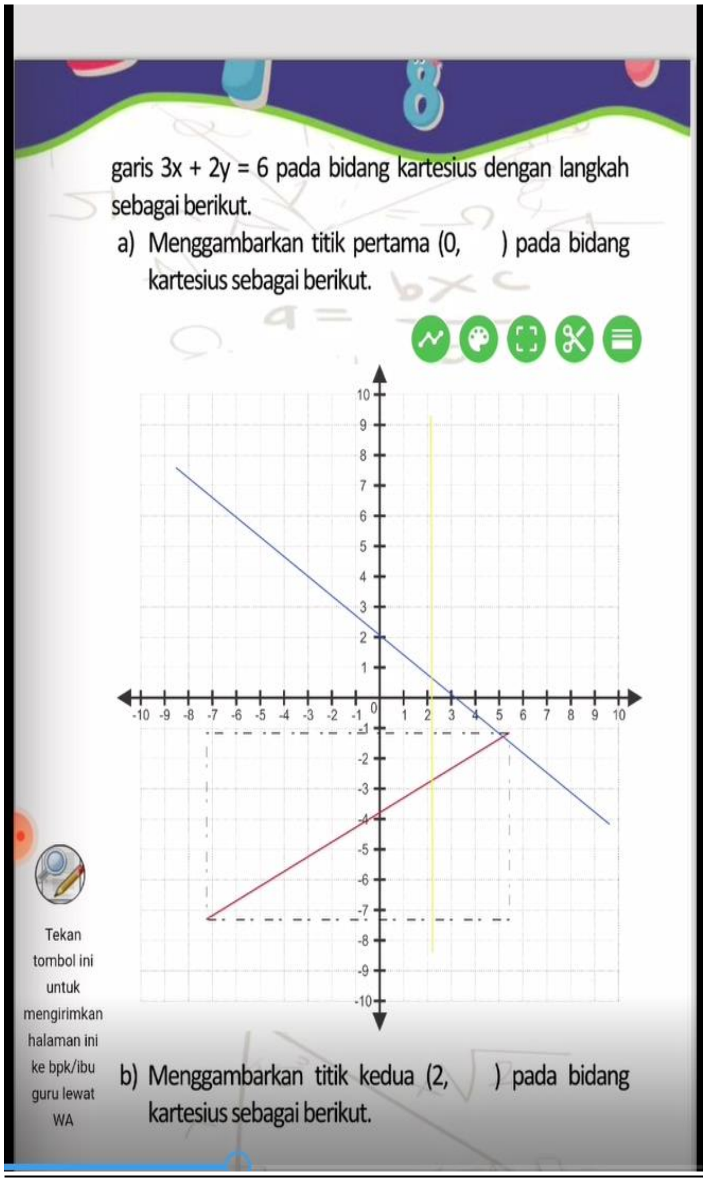
OK

"Pengetahuan membuat kita mengetahui, dan tindakan membuat kita memahami tetapi yang paling berharga adalah pengalaman"

20 Komposisi dan Invers Fungsi

29/66

Menggambar Kurva Lurus



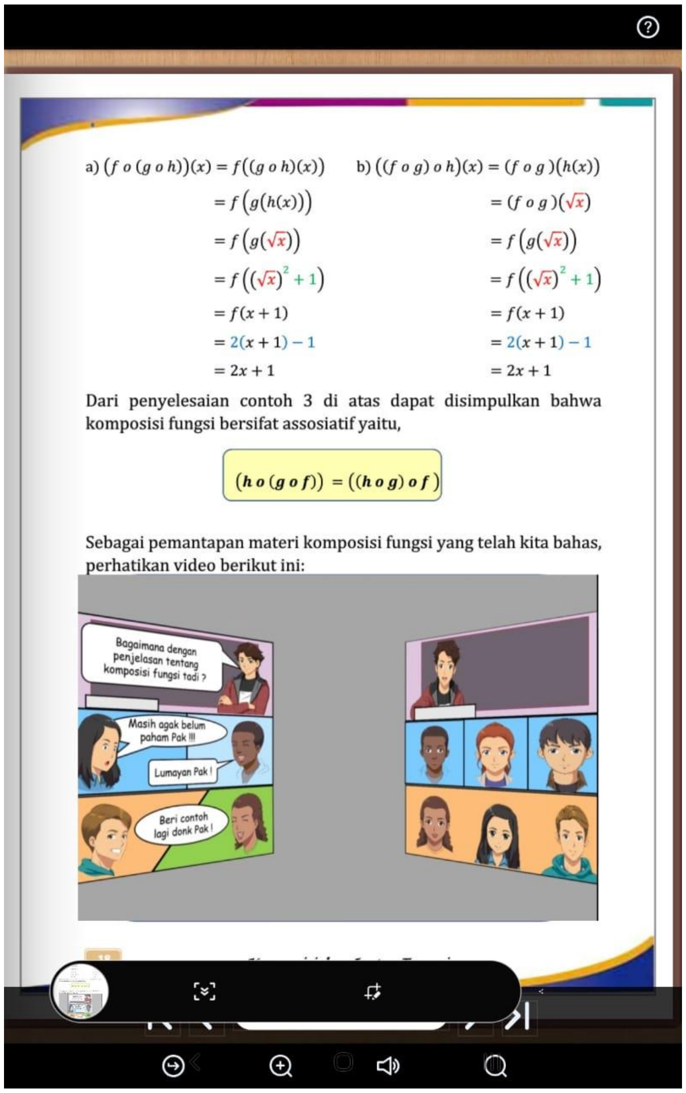
garis 3x + 2y = 6 pada bidang kartesius dengan langkah sebagai berikut.

a) Menggambarkan titik pertama (0,   ) pada bidang kartesius sebagai berikut.

Tekan tombol ini untuk mengirimkan halaman ini ke bpk/ibu guru lewat WA

b) Menggambarkan titik kedua (2,   ) pada bidang kartesius sebagai berikut.

Memutar Video



a) $(f \circ (g \circ h))(x) = f((g \circ h)(x))$
$= f(g(h(x)))$
$= f(g(\sqrt{x}))$
$= f((\sqrt{x})^2 + 1)$
$= f(x+1)$
$= 2(x+1) - 1$
$= 2x + 1$

b) $((f \circ g) \circ h)(x) = (f \circ g)(h(x))$
$= (f \circ g)(\sqrt{x})$
$= f(g(\sqrt{x}))$
$= f((\sqrt{x})^2 + 1)$
$= f(x+1)$
$= 2(x+1) - 1$
$= 2x + 1$

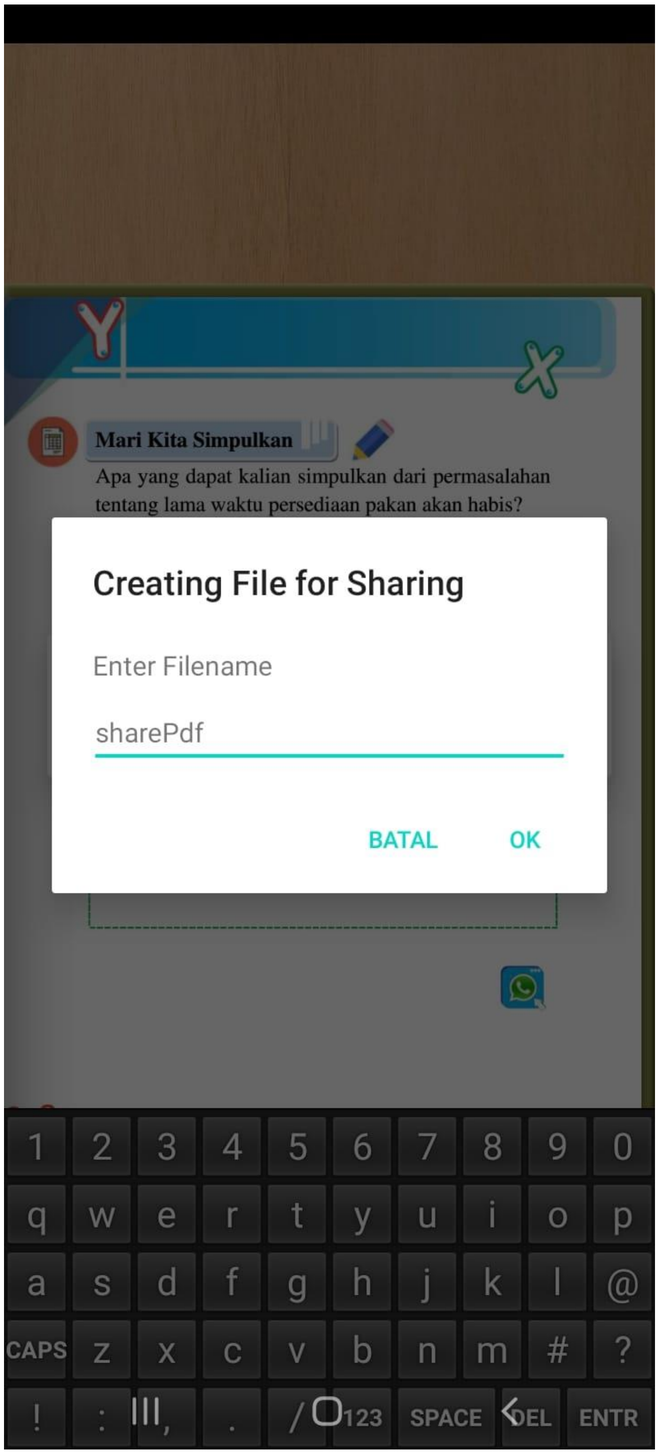
Dari penyelesaian contoh 3 di atas dapat disimpulkan bahwa komposisi fungsi bersifat assosiatif yaitu,

$$\boxed{(h \circ (g \circ f)) = ((h \circ g) \circ f)}$$

Sebagai pemantapan materi komposisi fungsi yang telah kita bahas, perhatikan video berikut ini:

Bagaimana dengan penjelasan tentang komposisi fungsi tadi ?
Masih agak belum paham Pak !!!
Lumayan Pak !
Beri contoh lagi donk Pak !

Mengirimkan Pesan



Mari Kita Simpulkan

Apa yang dapat kalian simpulkan dari permasalahan tentang lama waktu persediaan pakan akan habis?

Creating File for Sharing

Enter Filename

sharePdf

Adapun *source code* pada masing-masing komponen pada fitur-fitur aplikasi **FaPa FlipBook Extender** disajikan seperti berikut.

COMPONENT MODULES

>> BASE COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Rect;
import android.graphics.Typeface;
import android.icu.util.ULocale;
import android.os.Build;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RelativeLayout;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.ndayax.eflipmodule.db.FlipBookDb;
import org.ndayax.eflipmodule.db.entities.BaseEntity;
import org.ndayax.eflipmodule.db.entities.CheckBoxEntity;
import org.ndayax.eflipmodule.db.entities.EditBoxEntity;
import org.ndayax.eflipmodule.db.entities.FlipBookEntity;
import org.ndayax.eflipmodule.db.entities.GraphBoxEntity;
import org.ndayax.eflipmodule.db.entities.RadioBoxEntity;
import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.models.FlipBookPage;
import org.ndayax.eflipmodule.utils.BitmapUtils;
import org.ndayax.eflipmodule.utils.JSONUtils;

import java.util.List;

import androidx.annotation.RequiresApi;

public class BaseComponent {

    public enum ComponentType {
        FLIP_BOX,
        LENS_BOX,
        VIDEO_BOX,
        TEXT_BOX,
        EDIT_BOX,
        RADIO_BOX,
        CHECK_BOX,
        GRAPH_BOX,
        CAMERA_BOX,
        DESCRIPTION_BOX,
        SCORING_BOX,
        TOGGLE_BOX,
        POPUP_BOX,
        PICTURE_BOX,
```

```java
        SHARE_BUTTON,
        POPUP_BUTTON,
        SHOW_HIDDEN_BUTTON,
        PLAY_VIDEO_BUTTON,
        TOGGLE_BUTTON,
        GEOGEBRA_BOX,
        PDFREADER_BOX
    };
    public enum TextAlignment {
        LEFT,
        CENTER,
        RIGHT
    };

    private Context mContext;

    private String mId;
    private int mGroupId;

    private float mLeft;
    private float mTop;
    private float mWidth;
    private float mHeight;
    private boolean mHide;
    private float mAspectRatio;

    private int mExamId;
    private String mExamCategory;

    private String mText;
    private String mHiddenText;
    private TextAlignment mTextAlign = TextAlignment.LEFT;
    private float mTextSize;
    private String mTextColor;
    private ComponentType mType;
    private View mView;
    private BaseEntity mEntity;
    private FlipBookPage mPage;
    private BaseComponent mHiddenComponent;

    private List<BaseComponent> mGroupMembers;
    private List<BaseComponent> mGroupExam;

    protected BaseComponent(Context context, ComponentType type) {
        mContext = context;
        mType = type;

        mAspectRatio = 0.0f;
        mHide = false;
        mGroupId = -1;

        mExamId = 0;
        mExamCategory = "";
        mEntity = null;
        mPage = null;
        mHiddenComponent = null;
        mHiddenText = "";
    }
    public Context getContext() {
        return mContext;
    }
    public ComponentType getType() {
        return mType;
    }
    public void setEntity(BaseEntity entity) {
        mEntity = entity;
    }
    public BaseEntity getEntity() {
        return mEntity;
    }
    public void setPage(FlipBookPage page){
        mPage = page;
    }
    public FlipBookPage getPage() {
        return mPage;
    }
    public void setId(String id) {
        mId = id;
    }
    public String getId() {
        return mId;
    }
    public void setGroupId(int groupId) {
        mGroupId = groupId;
    }
    public int getGroupId() {
        return mGroupId;
    }
    public void setExamId(int examId) {
        mExamId = examId;
    }
    public int getExamId() {
        return mExamId;
    }
    public boolean isExamComponent() {
        return mExamId > 0;
    }
    public void setExamCategory(String examCategory) {
        mExamCategory = examCategory;
    }
    public String getExamCategory() {
        return mExamCategory;
    }
    public void setLeft(float left) {
        mLeft = left;
    }
    public float getLeft() {
        return mLeft;
    }
    public void setTop(float top) {
        mTop = top;
    }
    public float getTop() {
        return mTop;
```

```java
    }
    public void setWidth(float width) {
        mWidth = width;
    }
    public float getWidth() {
        return mWidth;
    }
    public void setHeight(float height) {
        mHeight = height;
    }
    public float getHeight() {
        return mHeight;
    }
    public void setHidden(boolean hide){
        mHide = hide;
    }
    public boolean isHidden() {
        return mHide;
    }

    /**
     * default initialize for EditBox, CheckBoxEx, RadioBox and GraphBox
     *
     * @param flipBookEntity
     */
    protected void initComponent(FlipBookEntity flipBookEntity) {
        if (this instanceof EditBox) {
            EditBoxEntity editBoxEntity =
flipBookEntity.getEditBoxEntity(getId());
            editBoxEntity.setComponent(this);
            editBoxEntity.setComponentId(getId());
            setEntity(editBoxEntity);
        } else if (this instanceof CheckBoxEx) {
            CheckBoxEntity checkBoxEntity =
flipBookEntity.getCheckBoxEntity(getId());
            checkBoxEntity.setComponent(this);
            checkBoxEntity.setComponentId(getId());
            setEntity(checkBoxEntity);
        } else if (this instanceof RadioBox) {
            RadioBoxEntity radioBoxEntity =
flipBookEntity.getRadioBoxEntity(getId());
            radioBoxEntity.setComponent(this);
            radioBoxEntity.setComponentId(getId());
            setEntity(radioBoxEntity);
        } else if (this instanceof GraphBox) {
            GraphBoxEntity graphBoxEntity =
flipBookEntity.getGraphBoxEntity(getId());
            graphBoxEntity.setComponent(this);
            graphBoxEntity.setComponentId(getId());
            setEntity(graphBoxEntity);
        }
    }
    public float getAspectRatio() {
        return mAspectRatio;
    }

    /**
```

```
     * Register the view
     *
     * @param view
     */
    protected void registerView(View view) {
        mView = view;
        mView.setTag(this);
    }

    /**
     * Unregister registered view
     */
    protected void unRegisterView() {
        mView = null;
        mView.setTag(null);
    }

    /**
     *
     * @param oldView The view you want to replace
     * @param newView The replacement view
     */
    protected void replaceView(View oldView, View newView) {
        mView = newView;
        oldView.setTag(null);
        newView.setTag(this);
    }
    public View getView() {
        return mView;
    }

    /**
     * Setting the text alignment on the view
     *
     * @param view
     */
    protected void setAlign(View view) {
        switch (getTextAlignment()) {
            case CENTER:
                view.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
                break;
            case RIGHT:
                view.setTextAlignment(View.TEXT_ALIGNMENT_VIEW_END);
                break;
            default:
                view.setTextAlignment(View.TEXT_ALIGNMENT_VIEW_START);
                break;
        }
    }
    public Typeface getTypeface() {
        return Typeface.create("sans-serif-condensed", Typeface.NORMAL);
    }
    public void setTextAlignment(TextAlignment textAlign) {
        mTextAlign = textAlign;
    }
    public TextAlignment getTextAlignment() {
        return mTextAlign;
    }
    public void setText(String text){
        mText = text;
    }
    public String getText() {
        return mText;
    }
    public void setHiddenText(String hiddenText) {
        mHiddenText = hiddenText;
    }
    public String getHiddenText(){
        return mHiddenText;
    }
    public void setHiddenComponent(BaseComponent hiddenComponent) {
        mHiddenComponent = hiddenComponent;
    }
    public BaseComponent getHiddenComponent(){
        return mHiddenComponent;
    }
    public void setTextSize(float textSize){
        mTextSize = BitmapUtils.convertPtToPx(getContext(), (int)textSize) ;
    }
    public float getTextSize() {
        return mTextSize;
    }
    public void setTextColor(String textColor){
        mTextColor = textColor;
    }
    public String getTextColor() {
        return mTextColor;
    }
    public void setGravity(View view) {
    }
    public void stop() {
        if (getView() != null) {
            ViewGroup viewGroup = (ViewGroup) getView().getParent();
            if (viewGroup != null) {
                viewGroup.removeAllViews();
            }
        }

        removeChilds();
        save();
        hide();
    }
    public void start() {
        show();
    }
    public void save() {
        if (mEntity != null &&
                isValueChanged()) {
            SQLiteDatabase db =
FlipBookDb.getInstance().getWritableDatabase();
            if(mEntity.getParent() != null) {
                mEntity.getParent().save(db);
            }
            mEntity.save(db);
        }
    }
    protected boolean isValueChanged() {
        return false;
    }
    public void hide() {
        if (!mHide) {
            ActivityMgr.getInstance().hideView(mView);
            mHide = true;
        }
    }
    public void show() {
        if (mHide) {
            ActivityMgr.getInstance().showView(mView);
            mHide = false;
        }
    }
    private volatile boolean delayed = false;
    public void showHiddenTextOrComponent(long delay) {
        if (delayed) {
            return;
        }
        delayed = true;
        doShowHiddenTextOrComponent();

        mView.postDelayed(new Runnable() {
            @Override
            public void run() {
                doHideHiddenTextOrComponent();
                delayed = false;
            }
        }, delay);
    }
    protected void doShowHiddenTextOrComponent() {
        if (getHiddenComponent() != null && getPage() != null) {
            getPage().replaceView(this, getView(),
getHiddenComponent().getView());
        }
    }
    protected void doHideHiddenTextOrComponent() {
        if (getHiddenComponent() != null && getPage() != null) {
            getPage().replaceView(this, getHiddenComponent().getView(),
getView());
        }
    }
    protected void doShowDefault(){
    }
    protected void doHideDefault(){
    }
    public void removeChilds() {
    }
    public boolean isCorrectValue() {
        return false;
    }
    public int getScore() {
        return 0;
    }
```

```java
    public boolean isSelected() { return false; }
    public boolean isEmpty() { return true; }
    public boolean isInputForm() { return false; }
    public View getFirstView() {
        return null;
    }
    public View getViewAt(int index){
        return null;
    }
    public View getNextView() {
        return null;
    }
    public void movePrevView() {
    }
    public void moveNextView() {
    }
    public Rect getRect() {
        Rect rect = new Rect();
        rect.left = (int)(getLeft() * getAspectRatio());
        rect.top = (int)(getTop() * getAspectRatio());
        rect.right = rect.left + (int)(getWidth() * getAspectRatio());
        rect.bottom = rect.top + (int)(getHeight() * getAspectRatio());

        return rect;
    }

    /**
     * check if the x,y coordinates are inside the view
     *
     * @param x, Coordinate x on screen
     * @param y, Coordinate y on screen
     * @return true if inside other false
     */
    public boolean contains(int x, int y) {
        Rect rect = getRect();
        if (rect != null &&
                rect.contains(x, y)) {
            return true;
        }
        return false;
    }
    public void showButtons() {
    }
    public void hideButtons() {
    }
    public void buttonClicked() {
    }
    public void setGroupMembers(List<BaseComponent> groupMembers) {
        mGroupMembers = groupMembers;
    }
    public List<BaseComponent> getGroupMembers() {
        return mGroupMembers;
    }
    public void setExamGroup(List<BaseComponent> groupExam) {
        mGroupExam = groupExam;
    }
    public List<BaseComponent> getExamGroup() {
        return mGroupExam;
    }
    public void resizeLayout(float aspectRatio, boolean force) {
        if (mAspectRatio == aspectRatio
                && !force) {
            return ;
        }

        mAspectRatio = aspectRatio;

        if (mView != null) {
            resizeView(mView, aspectRatio);
        }
    }

    /**
     * Resize view based on aspect ratio
     *
     * @param view, The view you want to resize
     * @param aspectRatio, The aspect ratio
     */
    protected void resizeView(View view, float aspectRatio) {
        RelativeLayout.LayoutParams params = (RelativeLayout.LayoutParams)
view.getLayoutParams();

        if (getType() == ComponentType.RADIO_BOX ||
                getType() == ComponentType.CHECK_BOX) {
            if (params == null) {
                params = new
RelativeLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
                        ViewGroup.LayoutParams.WRAP_CONTENT);
            } else {
                params.width = ViewGroup.LayoutParams.WRAP_CONTENT;
                params.height = ViewGroup.LayoutParams.WRAP_CONTENT;
            }
        } else {
            if (params == null) {
                params = new RelativeLayout.LayoutParams((int) (getWidth() *
aspectRatio + 0.5f),
                        (int) (getHeight() * aspectRatio + 0.5f));
            } else {
                params.width = (int) (getWidth() * aspectRatio *
getProportion() + 0.5f);
                params.height = (int) (getHeight() * aspectRatio *
getProportion() + 0.5f);
            }
        }

        params.leftMargin = (int) (getLeft() * aspectRatio  + 0.5f);
        params.topMargin = (int) (getTop() * aspectRatio + 0.5f);

        view.setPadding(0, 0, 0, 0);
        view.setLayoutParams(params);
    }
    protected float getProportion() {
        return 1.0f;
    }
    /**
     * Parse json script and generate component instance.
     *
     * @param context
     * @param flipBookEntity
     * @param page
     * @param jsonObject
     * @param root
     * @return Component instance
     * @throws JSONException
     * @throws RuntimeException
     */
    @RequiresApi(api = Build.VERSION_CODES.M)
    public static BaseComponent createComponentFromJson(Context context,
                                                        FlipBookEntity
flipBookEntity,
                                                        FlipBookPage page,
                                                        JSONObject jsonObject,
                                                        String root) throws
JSONException, RuntimeException {
        String componentType = JSONUtils.getString(jsonObject, "type", "");
        if (componentType.length() == 0) {
            throw new RuntimeException("Component name must defined...");
        }
        String id = JSONUtils.getString(jsonObject, "id", "");
        if(id.length() == 0) {
            throw new RuntimeException("Component id must be defined...");
        }

        int groupId = JSONUtils.getInt(jsonObject, "group", -1);
        int l = JSONUtils.getInt(jsonObject, "left", -1);
        if (l == -1 &&
                !(componentType.equals("popupbox"))) {
            throw new RuntimeException("Component left position must be
defined...");
        }

        int t = JSONUtils.getInt(jsonObject, "top", -1);
        if (t == -1 &&
                !(componentType.equals("popupbox"))) {
            throw new RuntimeException("Component top position must be
defined...");
        }

        int w = JSONUtils.getInt(jsonObject, "width", -1);
        if (w == -1 &&
                !(componentType.equals("popupbox"))) {
            throw new RuntimeException("Component width must be defined...");
        }

        int h = JSONUtils.getInt(jsonObject, "height", -1);
        if (h == -1 &&
                !(componentType.equals("popupbox"))) {
            throw new RuntimeException("Component height must be defined...");
        }
        String hiddenText = JSONUtils.getString(jsonObject, "hidden_text",
"");
        JSONObject hiddenObj = JSONUtils.getJSONObject(jsonObject,
"hidden_component");
        BaseComponent hiddenComponent = null;
        if (hiddenObj != null) {
            hiddenComponent = BaseComponent.createComponentFromJson(context,
flipBookEntity, page,
                    hiddenObj, root);
        }

        switch (componentType) {
            case "flipbox": {
                String folder = JSONUtils.getString(jsonObject, "folder", "");
                boolean shadow = JSONUtils.getBoolean(jsonObject, "shadow",
false);
                if (folder.length() == 0) {
                    throw new RuntimeException("Book folder must be
defined...");
                }
                return FlipBookBox.create(context, page, id, groupId,
(float)l, (float)t,
                        (float)w, (float)h, root + "/" + folder, shadow,
hiddenComponent);
            }
            case "videobox": {
                String filename = JSONUtils.getString(jsonObject,
"video_file", "");
                if (filename.length() == 0)
                    throw new RuntimeException("File video must be
defined...");
                boolean repeat = JSONUtils.getBoolean(jsonObject, "repeat",
false);
                boolean autoplay = JSONUtils.getBoolean(jsonObject,
"autoplay", false);
                return VideoBox.create(context, id, (float)l, (float)t,
(float)w, (float)h,
                        repeat, autoplay, root + "/" + filename);
            }
            case "lensbox": {
                LensBox lensBox = LensBox.create(context, page, id, (float)l,
(float)t, (float)w, (float)h,
                        root, hiddenComponent);
                return lensBox;
            }
            case "editbox": {
                String text = JSONUtils.getString(jsonObject, "text", "");
                float textSize = JSONUtils.getFloat(jsonObject, "text_size",
22.0f);
                String textColor = JSONUtils.getString(jsonObject,
"text_color", "#FF000000");
                String align = JSONUtils.getString(jsonObject, "text_align",
"left");
                int examId = JSONUtils.getInt(jsonObject, "exam_id", -1);
                String examCategory = JSONUtils.getString(jsonObject,
"exam_category", "");
                boolean isNumeric = JSONUtils.getBoolean(jsonObject,
"numeric", false);
                int maxLines = JSONUtils.getInt(jsonObject, "max_lines", 1);
                String hint = JSONUtils.getString(jsonObject, "hint", "");
                return EditBox.create(context, flipBookEntity, page, id,
groupId, l, t, w, h,
                        text, hint, hiddenText, textSize, textColor, align,
examId, examCategory,
                        isNumeric, maxLines, hiddenComponent);
            }
            case "textbox": {
                String text = JSONUtils.getString(jsonObject, "text", "");
                float textSize = JSONUtils.getFloat(jsonObject, "text_size",
22.0f);
                String textColor = JSONUtils.getString(jsonObject,
"text_color", "#FF000000");
                String align = JSONUtils.getString(jsonObject, "text_align",
"left");
                return TextBox.create(context, page, id, groupId, l, t, w, h,
text, hiddenText,
                        textSize, textColor, align, hiddenComponent);
            }
            case "checkbox": {
                boolean checked = JSONUtils.getBoolean(jsonObject, "checked",
false);
                int examId = JSONUtils.getInt(jsonObject, "exam_id", -1);
                String examCategory = JSONUtils.getString(jsonObject,
"exam_category", "");
                int score = JSONUtils.getInt(jsonObject, "exam_score", 0);
                String msg = JSONUtils.getString(jsonObject,
"on_checked_message", null);
                String title = JSONUtils.getString(jsonObject,
"on_checked_message_title", "Message");
                return CheckBoxEx.create(context, flipBookEntity, id, groupId,
l, t, w, h,
                        checked, examId, examCategory, score, msg, title);
            }
            case "radiobox": {
                boolean correctValue = JSONUtils.getBoolean(jsonObject,
"correct_value", false);
                int examId = JSONUtils.getInt(jsonObject, "exam_id", -1);
                String examCategory = JSONUtils.getString(jsonObject,
"exam_category", "");
                int score = JSONUtils.getInt(jsonObject, "exam_score", 0);
                String onCheckedMsg = JSONUtils.getString(jsonObject,
"on_checked_message", null);
                String onCheckedMsgTitle = JSONUtils.getString(jsonObject,
"on_checked_message_title", "Message");
                return RadioBox.create(context, flipBookEntity, id, groupId,
l, t, w, h,
                    correctValue, examId, examCategory, score, onCheckedMsg,
onCheckedMsgTitle);
            }
            case "graphbox": {
                int examId = JSONUtils.getInt(jsonObject, "exam_id", -1);
                String examCategory = JSONUtils.getString(jsonObject,
"exam_category", "");
                String bg_filename = JSONUtils.getString(jsonObject,
"background", "");
                int maxPoints = JSONUtils.getInt(jsonObject, "max_points", 2);
                String buttons = JSONUtils.getString(jsonObject, "buttons",
"bottom");
                return GraphBox.create(context, flipBookEntity, page, id,
groupId, l, t, w, h,
                        maxPoints, examId, examCategory, root + "/" +
bg_filename, buttons,
                        hiddenComponent);
            }
            case "descriptionbox": {
                float textSize = JSONUtils.getFloat(jsonObject, "text_size",
22.0f);
                String textColor = JSONUtils.getString(jsonObject,
"text_color", "#FF000000");
                int examId = JSONUtils.getInt(jsonObject, "exam_id", -1);
                String examCategory = JSONUtils.getString(jsonObject,
"exam_category", "");
                JSONArray description = JSONUtils.getJSONArray(jsonObject,
"description");
                if (description == null) {
                    throw new RuntimeException("Description must be defined
...");
                }
                return DescriptionBox.create(context, id, groupId, l, t, w, h,
textSize, textColor,
                        examId, examCategory, description);
            }
            case "scoringbox":{
                float textSize = JSONUtils.getFloat(jsonObject, "text_size",
22.0f);
                String textColor = JSONUtils.getString(jsonObject,
"text_color", "#FF000000");
                int examId = JSONUtils.getInt(jsonObject, "exam_id", -1);
                String examCategory = JSONUtils.getString(jsonObject,
"exam_category", "");
                boolean isPercentage = JSONUtils.getBoolean(jsonObject,
"percentage", false);
                return ScoringBox.create(context, id, groupId, l, t, w, h,
textSize,
                        textColor, examId, examCategory, isPercentage);
            }
            case "popupbox": {
                JSONObject obj = jsonObject.getJSONObject("component");
                return PopupBox.create(context, flipBookEntity, page, obj,
root, id);
            }
            case "togglebox": {
                JSONArray array = jsonObject.getJSONArray("components");
                return ToggleBox.create(context, flipBookEntity, page, array,
root, id, groupId, l, t, w, h);
            }
            case "picturebox": {
                String filePath = JSONUtils.getString(jsonObject, "file_path",
"");
                boolean hide = JSONUtils.getBoolean(jsonObject, "hide",
false);
```

```
                return PictureBox.create(context, page, id, groupId, l, t, w,
h, root, filePath,
                        hide, hiddenComponent);
            }
            case "camerabox" : {
                int examId = JSONUtils.getInt(jsonObject, "exam_id", -1);
                String examCategory = JSONUtils.getString(jsonObject,
"exam_category", "");
                return CameraBox.create(context, flipBookEntity, page, id,
groupId, l, t, w, h,
                        examId, examCategory, root);
            }
            case "share_button": {
                String pageTypeStr = JSONUtils.getString(jsonObject,
"page_type", "none");
                boolean whatsAppOnly = JSONUtils.getBoolean(jsonObject,
"whats_app_only", false);
                String pageNo = JSONUtils.getString(jsonObject, "page_no",
"");
                return ShareButton.create(context, page, id, groupId, l, t, w,
h, pageTypeStr, pageNo,
                        whatsAppOnly);
            }
            case "popup_button": {
                String popupComponent = JSONUtils.getString(jsonObject,
"popup_component", "");
                String errMsg = JSONUtils.getString(jsonObject, "err_msg",
null);
                return PopupButton.create(context, page, id, groupId, l, t, w,
h, popupComponent, errMsg);
            }
            case "show_hidden_button": {
                String component_id = JSONUtils.getString(jsonObject,
"component_id", "");
                int component_group = JSONUtils.getInt(jsonObject,
"component_group", -1);
                long delay = JSONUtils.getLong(jsonObject, "delay", 1000);
                return ShowHiddenButton.create(context, page, l, t, w, h,
component_id,
                        component_group, delay);
            }
            case "play_video_button": {
                String localFile = JSONUtils.getString(jsonObject,
"local_file", "");
                String videoUrl = JSONUtils.getString(jsonObject, "video_url",
"");
                return PlayVideoButton.create(context, page, id, l, t, w, h,
root, localFile,
                        videoUrl);
            }
            case "pdf_reader_box":{
                String pdfFile = JSONUtils.getString(jsonObject, "filename",
"");
                Boolean shadow = JSONUtils.getBoolean(jsonObject, "shadow",
false);
                return PdfReaderBox.create(context, page, id, groupId, l, t,
w, h, root + "/" + pdfFile,
                        shadow);
            }
            default: return null;
        }
    }
    public String toString() {
        switch (getType()) {
            case EDIT_BOX: return "EDIT BOX (" + getId() + ")";
            case TEXT_BOX: return "TEXT BOX (" + getId() + ")";
            default: return "OTHER COMPONENTS (" + getId() + ")";
        }
    }
}
```

>> FLIPBOOKBOX COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.os.Build;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.RelativeLayout;

import org.json.JSONException;
import org.json.JSONObject;
import org.ndayax.eflipmodule.R;
import org.ndayax.eflipmodule.models.FlipBookPage;
import org.ndayax.eflipmodule.utils.BitmapUtils;
import org.ndayax.eflipmodule.utils.FileUtils;
import org.ndayax.eflipmodule.views.FlipBookView;

import androidx.annotation.RequiresApi;

public class FlipBookBox extends BaseComponent {

    public FlipBookBox(Context context) {
        super(context, ComponentType.FLIP_BOX);
    }

    /**
     * initialize FlipBookBox
     *      - get the flipbookview to view the flipbook
     *
     * @param folder, The flipbook folder
     * @param shadow, The flipbook shadow
     */
    @RequiresApi(api = Build.VERSION_CODES.M)
    public void init(String folder, boolean shadow) {
        LayoutInflater inflater = LayoutInflater.from(getContext());
        View view = inflater.inflate(R.layout.web_activity_main, null);
        FlipBookView flipbook =
(FlipBookView)view.findViewById(R.id.flipbox_view);
        RelativeLayout pageLayout = (RelativeLayout)
view.findViewById(R.id.page_layout);
```

```
        flipbook.setTag(this);
        pageLayout.setTag(this);

        flipbook.initBook(pageLayout, folder);
        if (shadow) {
            Drawable drawable =
getContext().getResources().getDrawable(R.drawable.myrect2);
            view.setBackground(drawable);
            view.setElevation(BitmapUtils.convertDpToPx(getContext(), 5.0f));
            view.setTranslationZ(BitmapUtils.convertDpToPx(getContext(),
5.0f));
        }

        registerView(view);
    }

    /**
     * Load the flipbook page.
     *
     * @param page, The page to load
     */
    public void load(int page) {
        FlipBookView flipBookView =
(FlipBookView)getView().findViewById(R.id.flipbox_view);
        flipBookView.loadView(page);
    }

    @Override
    public void start() {
        super.start();
        load(1);
    }

    @Override
    public void removeChilds() {
        RelativeLayout pageLayout =
(RelativeLayout)getView().findViewById(R.id.page_layout);
        for (int i = 0; i < pageLayout.getChildCount(); i++) {
            View view = pageLayout.getChildAt(i);
            BaseComponent baseComponent = (BaseComponent)view.getTag();
            baseComponent.save();
            pageLayout.removeView(view);
        }
        super.removeChilds();
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    public static FlipBookBox create(Context context, FlipBookPage page,
String id, int groupId,
                                     float l, float t, float w, float h,
                                     String folder, boolean shadow,
                                     BaseComponent hiddenComponent) throws
JSONException {
        FlipBookBox flipBox = new FlipBookBox(context);
        flipBox.setId(id);
        flipBox.setGroupId(groupId);
        flipBox.setLeft(l);
        flipBox.setTop(t);
        flipBox.setWidth(w);
        flipBox.setHeight(h);
        flipBox.setHiddenComponent(hiddenComponent);
        flipBox.setPage(page);
        flipBox.init(folder, shadow);

        return flipBox;
    }
}
```

>> TEXTBOX COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;
import android.graphics.Color;
import android.util.TypedValue;
import android.view.Gravity;
import android.view.View;
import android.widget.TextView;

import org.ndayax.eflipmodule.models.FlipBookPage;

public class TextBox extends BaseComponent{

    private TextAlignment mAlignment;

    protected TextBox(Context context) {
        super(context, ComponentType.TEXT_BOX);
    }

    protected TextBox(Context context, ComponentType type) {
        super(context, type);

        mAlignment = TextAlignment.LEFT;
    }

    public void init() {
        TextView textView = new TextView(getContext());
        textView.setText(getText());
        textView.setTypeface(getTypeface());
        textView.setTextColor(Color.parseColor(getTextColor()));
        setAlign(textView);
        setGravity(textView);
        if (isHidden()) {
            textView.setVisibility(View.INVISIBLE);
        }
        registerView(textView);
    }

    @Override
    public void resizeLayout(float aspectRatio, boolean force) {
        super.resizeLayout(aspectRatio, force);

        TextView textView = (TextView)getView();
        textView.setTextSize(TypedValue.COMPLEX_UNIT_PX, getTextSize() *
aspectRatio);
    }

    @Override
    public void setGravity(View view) {
        TextView textView = (TextView)view;
        switch (getTextAlignment()) {
            case CENTER:
                textView.setGravity(Gravity.CENTER);
                break;
            case RIGHT:
                textView.setGravity(Gravity.END);
                break;
            default:
                textView.setGravity(Gravity.START);
                break;
        }
    }

    @Override
    protected void doShowHiddenTextOrComponent() {
        if (getHiddenText() != null && getHiddenText().length() > 0) {
            TextView textView = (TextView) getView();
            setText(textView.getText().toString());
            textView.setText(getHiddenText());
        } else {
            super.doShowHiddenTextOrComponent();
        }
    }

    @Override
    protected void doHideHiddenTextOrComponent() {
        if (getHiddenText() != null && getHiddenText().length() > 0) {
            TextView textView = (TextView) getView();
            textView.setText(getText());
        } else {
            super.doHideHiddenTextOrComponent();
        }
    }

    public static TextBox create(Context context, FlipBookPage page,
                                 String id, int groupId, float l, float t,
float w, float h,
                                 String text, String hiddenText, float
textSize, String textColor,
                                 String textAlign, BaseComponent
hiddenComponent) {
        TextBox textBox = new TextBox(context);
        textBox.setId(id);
        textBox.setGroupId(groupId);
        textBox.setLeft(l);
        textBox.setTop(t);
        textBox.setWidth(w);
        textBox.setHeight(h);
        textBox.setText(text);
        textBox.setTextSize(textSize);
        textBox.setTextColor(textColor);
        textBox.setHiddenText(hiddenText);
        textBox.setHiddenComponent(hiddenComponent);
        textBox.setPage(page);
        switch (textAlign) {
            case "center" : textBox.setTextAlignment(TextAlignment.CENTER);
break;
            case "right" : textBox.setTextAlignment(TextAlignment.RIGHT);
break;
            default: textBox.setTextAlignment(TextAlignment.LEFT); break;
        }

        textBox.init();
        return textBox;
    }

}
```

>> EDITBOX COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.drawable.GradientDrawable;
import android.os.Build;
import android.text.Editable;
import android.text.InputType;
import android.text.TextWatcher;
import android.util.DisplayMetrics;
import android.util.TypedValue;
import android.view.Gravity;
import android.view.MotionEvent;
import android.view.View;
import android.widget.EditText;
import android.widget.RelativeLayout;

import org.ndayax.eflipmodule.R;
import org.ndayax.eflipmodule.db.entities.EditBoxEntity;
import org.ndayax.eflipmodule.db.entities.FlipBookEntity;
import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.helpers.FlipBookKeyboard;
import org.ndayax.eflipmodule.models.FlipBookPage;

public class EditBox extends BaseComponent implements
View.OnFocusChangeListener, TextWatcher, View.OnTouchListener {
    private String mDefaultText;
    private boolean mMultiline;
    private int mMaxLines;
    private String mHint;

    protected EditBox(Context context) {
        super(context, ComponentType.EDIT_BOX);

        mMultiline = false;
        mMaxLines = 1;
        mHint = "";
    }

    public void init(FlipBookEntity flipBookEntity,
                     boolean isNumeric, int maxLines) {
        initComponent(flipBookEntity);

        mMaxLines = maxLines;
        if (mMaxLines > 1) {
            mMultiline = true;
        }
        EditText editText = new EditText(getContext());
        editText.setText(((EditBoxEntity)getEntity()).getText() == "" ?
getText() :
                ((EditBoxEntity)getEntity()).getText());
        editText.setTypeface(getTypeface());
        editText.setTextColor(Color.parseColor(getTextColor()));
        editText.setBackgroundResource(R.drawable.editbox);

        if (isNumeric) {
            editText.setInputType(InputType.TYPE_CLASS_NUMBER |
InputType.TYPE_NUMBER_VARIATION_NORMAL);
        }
        setAlign(editText);
        setGravity(editText);
        setNoBorder(editText);
        if (mHint.length() > 0) {
            editText.setHint(mHint);
        }
        // create bound rect  ...
        editText.addTextChangedListener(this);
        editText.setOnTouchListener(this);
        editText.setOnFocusChangeListener(this);

        registerView(editText);
        if (isHidden()) {
            hide();
        }
    }

    public boolean isMultiline() {
        return mMultiline;
    }

    @Override
    public void resizeLayout(float aspectRatio, boolean force) {
        super.resizeLayout(aspectRatio, force);

        EditText editText = (EditText)getView();
        editText.setTextSize(TypedValue.COMPLEX_UNIT_PX, getTextSize() *
aspectRatio);
        setPadding(editText);
    }

    private void setNoBorder(EditText editText) {
        if(Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN) {
            editText.setBackgroundDrawable(null);
        } else {
```

```java
            editText.setBackground(null);
        }
    }

    private void setMargin() {
        RelativeLayout.LayoutParams params = (RelativeLayout.LayoutParams)
getView().getLayoutParams();
    }

    private void setPadding(EditText editText) {
        float scale = ActivityMgr.getInstance().getDensity();
        int dpAsPixels = (int) (2*scale + 0.5f);
        editText.setPadding(dpAsPixels, 0, dpAsPixels,0);
    }

    @Override
    public void setGravity(View view) {
        EditText editText = (EditText) view;
        switch (getTextAlignment()) {
            case CENTER:
                editText.setGravity(Gravity.CENTER | Gravity.TOP);
                break;
            case RIGHT:
                editText.setGravity(Gravity.RIGHT | Gravity.TOP);
                break;
            default:
                editText.setGravity(Gravity.RIGHT | Gravity.TOP);
                break;
        }
    }

    @Override
    public String getText() {
        EditText editText = (EditText)getView();
        if (editText == null || (editText != null &&
                editText.getText().length() == 0)) {
            return super.getText();
        }
        return editText.getText().toString();
    }

    @Override
    protected boolean isValueChanged() {
        EditText editText = (EditText)getView();
        String edt = editText.getText().toString();
        EditBoxEntity editBoxEntity = (EditBoxEntity)getEntity();
        if (!edt.equals(editBoxEntity.getText())) {
           editBoxEntity.setText(edt);
           return true;
        }
        return false;
    }

    public void setDefaultText(String defaultText) {
        mDefaultText = defaultText;
    }
    public String getDefaultText() {
        return mDefaultText;
    }

    public void setHint(String hint) {
        mHint = hint;
    }

    public String getHint() {
        return mHint;
    }

    @Override
    protected void doShowDefault() {
        EditText editText = (EditText)getView();
        setText(editText.getText().toString());
        editText.setText("");
        editText.setText(getDefaultText());
    }

    @Override
    protected void doHideDefault() {
        EditText editText = (EditText)getView();
        editText.setText("");
        editText.setText(getText());
    }

    @Override
    protected void doShowHiddenTextOrComponent() {
        if (getHiddenText() != null || getHiddenText().length() > 0) {
            EditText editText = (EditText)getView();
            setText(editText.getText().toString());
            editText.setText("");
            editText.setText(getHiddenText());
        } else {
            super.doShowHiddenTextOrComponent();
        }
    }

    @Override
    protected void doHideHiddenTextOrComponent() {
        if (getHiddenText() != null || getHiddenText().length() > 0) {
            EditText editText = (EditText)getView();
            editText.setText("");
            editText.setText(getText());
        } else {
            super.doHideHiddenTextOrComponent();
        }
    }

    public static EditBox create(Context context, FlipBookEntity
flipBookEntity, FlipBookPage page,
                                 String id, int groupId, float l, float t,
float w, float h,
                                 String text, String hint, String hiddenText,
                                 float textSize, String textColor,
                                 String textAlign, int examId, String
examCategory, boolean isNumeric,
                                 int maxLines,
                                 BaseComponent hiddenComponent) {
        EditBox editBox = new EditBox(context);
        editBox.setId(id);
        editBox.setGroupId(groupId);
        editBox.setLeft(l);
        editBox.setTop(t);
        editBox.setWidth(w);
        editBox.setHeight(h);
        editBox.setText(text);
        editBox.setHint(hint);
        editBox.setTextSize(textSize);
        editBox.setTextColor(textColor);
        editBox.setExamId(examId);
        editBox.setExamCategory(examCategory);
        editBox.setHiddenText(hiddenText);
        editBox.setHiddenComponent(hiddenComponent);
        editBox.setPage(page);

        switch (textAlign) {
            case "center": editBox.setTextAlignment(TextAlignment.CENTER);
break;
            case "right": editBox.setTextAlignment(TextAlignment.RIGHT);
break;
            default: editBox.setTextAlignment(TextAlignment.LEFT); break;
        }
        editBox.init(flipBookEntity, isNumeric, maxLines);
        return editBox;
    }

    @Override
    public void onFocusChange(View view, boolean b) {
        if (!b) {
            FlipBookKeyboard.getInstance().closeKeyboard();
        } else {
            EditText editText = (EditText)view;
            editText.setSelection(editText.length());
        }
    }

    public boolean isFocusable() {
        EditText editText = (EditText)getView();
        if (editText != null) {
            return editText.isFocusable();
        } else {
            return false;
        }
    }

    public void releaseFocus() {
        EditText editText = (EditText)getView();
        if (editText != null) {
            editText.setFocusable(false);
        }
    }
    private CharSequence mOldCharSequence;
    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1,
int i2) {
        mOldCharSequence = charSequence;
    }

    @Override
    public void onTextChanged(CharSequence charSequence, int i, int i1, int
i2) {
    }

    @Override
    public void afterTextChanged(Editable editable) {
        EditText editText = (EditText)getView();
        if (mMaxLines > 1)  {
            if (editText.getLayout().getLineCount() > mMaxLines)
                editText.getText().delete(editText.getText().length() - 1,
editText.getText().length());
        } else {
            Paint paint = editText.getPaint();
            try {
                float textWidth = paint.measureText(editable.toString());
                int width = editText.getWidth() - (editText.getPaddingLeft() -
editText.getPaddingRight());
                if (textWidth >= (width - 4)) {
                    if (mOldCharSequence.length() > 0) {
                        editable.replace(0, mOldCharSequence.length(),
                                mOldCharSequence.subSequence(0,
mOldCharSequence.length() - 1));
                    }
                }
            } catch (Exception e) {
                e.getStackTrace();
            }
        }
    }

    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        FlipBookKeyboard.getInstance().openKeyboard(view);
        view.setFocusable(true);
        view.setFocusableInTouchMode(true);
        view.requestFocus();
        return true;
    }
}
```

>> PICTUREBOX COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;
import android.renderscript.BaseObj;
```

```java
import org.ndayax.eflipmodule.models.FlipBookPage;
import org.ndayax.eflipmodule.views.PictureView;

public class PictureBox extends BaseComponent{

    public PictureBox(Context context) {
        super(context, ComponentType.PICTURE_BOX);
    }

    /**
     *
     * @param root
     * @param filePath
     */
    public void init(String root, String filePath) {
        PictureView pictureView = new PictureView(getContext());
        pictureView.setPictureFolder(root);
        pictureView.setFilePath(filePath);
        registerView(pictureView);
    }

    public static PictureBox create(Context context, FlipBookPage page,
                                    String id, int group, float left, float
top,
                                    float width, float height, String root,
                                    String filePath, boolean hide,
BaseComponent hiddenComponent) {
        PictureBox pictureBox = new PictureBox(context);
        pictureBox.setPage(page);
        pictureBox.setId(id);
        pictureBox.setGroupId(group);
        pictureBox.setLeft(left);
        pictureBox.setTop(top);
        pictureBox.setWidth(width);
        pictureBox.setHeight(height);
        pictureBox.setHidden(hide);
        pictureBox.setHiddenComponent(hiddenComponent);
        pictureBox.init(root, filePath);
        return pictureBox;
    }

}
```

>> CAMERABOX COMPONENT

```java
package org.ndayax.eflipmodule.components;

import android.content.Context;
import android.graphics.Color;
import android.graphics.Matrix;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import org.ndayax.eflipmodule.R;
import org.ndayax.eflipmodule.db.entities.FlipBookEntity;
import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.models.FlipBookPage;
import org.ndayax.eflipmodule.utils.FileUtils;
import org.ndayax.eflipmodule.views.CameraView;
import org.ndayax.eflipmodule.views.PictureView;

import java.io.File;

import androidx.core.view.ViewCompat;

public class CameraBox extends BaseComponent implements View.OnClickListener {

    int mCameraViewId ;
    int mPictureViewId ;
    volatile boolean mPictureTaken;

    @Override
    public void onClick(View view) {
        if (mViewMode == VIEW_MODE.CAMERA_MODE && !mPictureTaken) {
            mPictureTaken = true;
            mCameraView.takePicture();
        } else {
            switchViewMode();
        }
    }

    public enum VIEW_MODE {
        CAMERA_MODE,
        IMAGE_MODE
    }

    private String mPictureFolder;
    private PictureView mPictureView;
    private CameraView mCameraView;
    private VIEW_MODE mViewMode;
    private FlipBookPage mPage;
    private TextView mTextView;
    private FloatingActionButton mFab;

    public CameraBox(Context context) {
        super(context, ComponentType.CAMERA_BOX);
        mViewMode = VIEW_MODE.CAMERA_MODE;
        mPictureTaken = false;
    }

    /**
     * CameraBox initialization
     *
     * @param entity The entity of FlipBook
     * @param page The page of FlipBook
     * @param pictureFolder The external storage folder for picture.
     */
    public void init(FlipBookEntity entity, FlipBookPage page, String
pictureFolder) {
        initComponent(entity);

        mPictureFolder = pictureFolder;
        mPage = page;

        mCameraViewId = ViewCompat.generateViewId();
        mPictureViewId = ViewCompat.generateViewId();

        ViewGroup.LayoutParams params = new
RelativeLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT);

        mPictureView = new PictureView(getContext());
        mPictureView.setPictureFolder(pictureFolder);
        mPictureView.setId(mPictureViewId);
        mPictureView.setLayoutParams(params);

        mCameraView = new CameraView(getContext());
        mCameraView.setSaveFolder(pictureFolder);
        mCameraView.setId(mCameraViewId);
        mCameraView.setLayoutParams(params);

        mFab = new FloatingActionButton(getContext());
        RelativeLayout.LayoutParams fabParams = new
RelativeLayout.LayoutParams(
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);
        fabParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT,
RelativeLayout.TRUE);
        mFab.setLayoutParams(fabParams);

        mFab.setElevation(6.f);
        mFab.setOnClickListener(this);

        FileUtils.createDirectory(getContext(), pictureFolder);
        File file = FileUtils.createFile(getContext(), pictureFolder,
                FileUtils.DIRECTORY_SEPARATOR + getId() + ".jpg");

        if (!file.exists()) {
            // cek for default pic ...
            file = FileUtils.createFile(getContext(), pictureFolder,
                    FileUtils.DIRECTORY_SEPARATOR + getId() + "-default.jpg");
            if (file.exists()) {
                mPictureView.setFilePath(getId() + "-default.jpg");
            }
        }

        if (!file.exists()) {
            registerView(mCameraView);
            mViewMode = VIEW_MODE.CAMERA_MODE;
        } else {
            registerView(mPictureView);
            mViewMode = VIEW_MODE.IMAGE_MODE;
        }

        setFabBelow(getView());
    }

    /**
     * check if the picture with this object id is exist or not
     * @return true if the picture with this object id is not found else
false.
     */
    @Override
    public boolean isEmpty() {
        File file = FileUtils.createFile(getContext(), mPictureFolder,
                FileUtils.DIRECTORY_SEPARATOR + getId() + ".jpg");
        return !file.exists();
    }

    @Override
    public boolean isInputForm() {
        return super.isInputForm();
    }

    /**
     * Set the floating action button (FAB) below the camera box view
     * @param view The camera box view
     */
    private void setFabBelow(View view) {
        RelativeLayout.LayoutParams fabLayoutParams =
(RelativeLayout.LayoutParams)mFab.getLayoutParams();

        fabLayoutParams.removeRule(RelativeLayout.BELOW);
        fabLayoutParams.addRule(RelativeLayout.BELOW, view.getId());

        ActivityMgr.getInstance().getCurrentActivity().runOnUiThread(new
Runnable() {
            @Override
            public void run() {
                mFab.setLayoutParams(fabLayoutParams);
            }
        });
    }

    public PictureView getPictureView() {
        return mPictureView;
    }

    /**
     * Toggle the floating action button icon.
     *
     * if VIEW_MODE is CAMERA_MODE then set the photo camera icon
     * else set the camera icon.
     */
    private void toggleFabIcon() {
        if (mViewMode == VIEW_MODE.CAMERA_MODE) {
```

```
mFab.setImageDrawable(getContext().getDrawable(R.drawable.ic_baseline_photo_ca
mera_24));
        } else {

mFab.setImageDrawable(getContext().getDrawable(R.drawable.ic_baseline_camera_2
4));
        }

        setFabBelow(getView());
    }

    public FloatingActionButton getFab() {
        toggleFabIcon();

        return mFab;
    }

    boolean mFabFirstView = false;

    /**
     * Resize view of camera box view based on the aspect ratio of the
flipbook page.
     *
     * @param view The camera box view
     * @param aspectRatio The aspect ratio of the flipbook page
     */
    @Override
    protected void resizeView(View view, float aspectRatio) {
        super.resizeView(view, aspectRatio);

        if (!mFabFirstView) {
            mFab.getViewTreeObserver().addOnGlobalLayoutListener(new
ViewTreeObserver.OnGlobalLayoutListener() {
                @Override
                public void onGlobalLayout() {

mFab.getViewTreeObserver().removeOnGlobalLayoutListener(this);

                    int w = mFab.getWidth();
                    int h = mFab.getHeight();

                    RelativeLayout.LayoutParams params =
(RelativeLayout.LayoutParams) mFab.getLayoutParams();
                    params.rightMargin = (int)(w / 2.0) + 8;
                    params.topMargin -= (h + 8);
                    mFab.setLayoutParams(params);
                }
            });
            mFabFirstView = true;
        }
    }

    @Override
    public void start() {
        if (mViewMode == VIEW_MODE.CAMERA_MODE) {
            mCameraView.start();
        }
        super.start();
    }

    @Override
    public void stop() {
        if (mViewMode == VIEW_MODE.CAMERA_MODE) {
            mPictureTaken = false;
            mCameraView.stop();
        }

        super.stop();
    }

    /**
     *
     */
    public void switchViewMode() {
        if (mViewMode == VIEW_MODE.CAMERA_MODE) {
            mPictureTaken = false;
            mCameraView.stop();
            replaceView(mCameraView, mPictureView);
            mPage.replaceView(this, mCameraView, mPictureView);
            mViewMode = VIEW_MODE.IMAGE_MODE;
        } else {
            mViewMode = VIEW_MODE.CAMERA_MODE;
            replaceView(mPictureView, mCameraView);
            mPage.replaceView(this, mPictureView, mCameraView);
            mCameraView.start();
        }
        toggleFabIcon();
    }

    /**
     *
     * @param context
     * @param entity
     * @param page
     * @param id
     * @param groupId
     * @param left
     * @param top
     * @param width
     * @param height
     * @param examId
     * @param examCategory
     * @param rootFolder
     *
     * @return CameraBox instance
     */
    public static CameraBox create(Context context, FlipBookEntity entity,
FlipBookPage page,
                                   String id, int groupId,
                                   float left, float top, float width,
                                   float height, int examId, String
examCategory,
                                   String rootFolder) {
```

```java
        CameraBox cameraBox = new CameraBox(context);
        cameraBox.setId(id);
        cameraBox.setGroupId(groupId);
        cameraBox.setLeft(left);
        cameraBox.setTop(top);
        cameraBox.setWidth(width);
        cameraBox.setHeight(height);
        cameraBox.setExamId(examId);
        cameraBox.setExamCategory(examCategory);
        cameraBox.init(entity, page, rootFolder + "/pictures");
        return cameraBox;
    }
}
```

>> CHECKBOX COMPONENT

```java
package org.ndayax.eflipmodule.components;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.util.Log;
import android.view.View;
import android.view.ViewTreeObserver;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.RelativeLayout;

import org.ndayax.eflipmodule.db.entities.CheckBoxEntity;
import org.ndayax.eflipmodule.db.entities.FlipBookEntity;
import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.views.PopupView;

import java.util.List;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;

public class CheckBoxEx extends BaseComponent implements
CompoundButton.OnCheckedChangeListener {
    private int mScore;
    private PopupView mPopupView;
    private boolean mChecked;
    private String mOnCheckedMsg;
    private String mOnCheckedMsgTitle;

    protected CheckBoxEx(Context context) {
        super(context, ComponentType.CHECK_BOX);
    }

    protected void init(FlipBookEntity flipBookEntity, boolean isChecked,
                        String onCheckedMsg, String onCheckedMsgTitle) {
        initComponent(flipBookEntity);
        CheckBox checkBox = new CheckBox(getContext());
        checkBox.setChecked(((CheckBoxEntity)getEntity()).isChecked() ?
                ((CheckBoxEntity)getEntity()).isChecked() : isChecked);
        mChecked = ((CheckBoxEntity)getEntity()).isChecked() ?
                ((CheckBoxEntity)getEntity()).isChecked() : isChecked;
        mOnCheckedMsg = onCheckedMsg;
        mOnCheckedMsgTitle = onCheckedMsgTitle;
        checkBox.setOnCheckedChangeListener(this);
        registerView(checkBox);
    }

    public void setScore(int score) {
        this.mScore = score;
    }

    @Override
    public void resizeLayout(float aspectRatio, boolean force) {
        super.resizeLayout(aspectRatio, force);
        CheckBox checkBox = (CheckBox) getView();

        checkBox.getViewTreeObserver().addOnGlobalLayoutListener(new
ViewTreeObserver.OnGlobalLayoutListener() {
            @Override
            public void onGlobalLayout() {

checkBox.getViewTreeObserver().removeOnGlobalLayoutListener(this);

                int w = checkBox.getWidth();
                int h = checkBox.getHeight();

                float scaleFactor = Math.min(((getWidth() + 10.f) *
aspectRatio)/(float)w,
                        ((getHeight() + 10.f) * aspectRatio)/(float)h);

                checkBox.setScaleX(scaleFactor);
                checkBox.setScaleY(scaleFactor);

                float x = ((getWidth() * aspectRatio) - w) / 2.f;
                float y = ((getHeight()  * aspectRatio) - h) / 2.f;

                RelativeLayout.LayoutParams params =
(RelativeLayout.LayoutParams)checkBox.getLayoutParams();
                params.leftMargin += (int)x;
                params.topMargin += (int)y;
                checkBox.setLayoutParams(params);
            }
        });
    }

    @Override
    public int getScore() {
        return mScore;
    }

    public boolean isChecked() {
        CheckBox checkBox = (CheckBox) getView();
        if (checkBox == null) {
            return false;
        }
        return  checkBox.isChecked();
    }

    public void setChecked(boolean checked) {
        CheckBox checkBox = (CheckBox) getView();
        if (checkBox != null) {
            checkBox.setChecked(checked);
        }
    }

    /**
     *
     * @param context
     * @param flipBookEntity
     * @param id
     * @param groupId
     * @param left
     * @param top
     * @param width
     * @param height
     * @param isChecked
     * @param examId
     * @param examCategory
     * @param score
     * @param onCheckedMsg
     * @param onCheckedMsgTitle
     *
     * @return CheckBox instance
     */

    public static CheckBoxEx create(Context context, FlipBookEntity
flipBookEntity,
                                    String id, int groupId, float left, float
top,
                                    float width, float height, boolean
isChecked,
                                    int examId, String examCategory, int
score, String onCheckedMsg,
                                    String onCheckedMsgTitle) {

        CheckBoxEx checkBoxEx = new CheckBoxEx(context);
        checkBoxEx.setId(id);
        checkBoxEx.setGroupId(groupId);
        checkBoxEx.setLeft(left);
        checkBoxEx.setTop(top);
        checkBoxEx.setWidth(width);
        checkBoxEx.setHeight(height);
        checkBoxEx.setExamId(examId);
        checkBoxEx.setExamCategory(examCategory);
        checkBoxEx.setScore(score);
        checkBoxEx.init(flipBookEntity, isChecked, onCheckedMsg,
onCheckedMsgTitle);
        return checkBoxEx;
    }

    /**
     * if the checkbox component is checked then remove the check from other
checkboxes in the same group
```

```
     * when checked, if the checkbox component has message, show the message.
     *
     * @param compoundButton
     * @param b If checked b is true.
     */
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
        if (b) {
            List<BaseComponent> members = getGroupMembers();
            if (isExamComponent() || members.size() > 0) {
                for (int i = 0; i < members.size(); i++) {
                    BaseComponent member = members.get(i);
                    if (member != this && member.getGroupId() ==
this.getGroupId()) {
                        if (member instanceof CheckBoxEx) {
                            ((CheckBoxEx) member).setChecked(false);
                        } else if (member instanceof ScoringBox) {
                            ((ScoringBox) member).calculateExamScore();
                        } else if (member instanceof DescriptionBox) {
                            ((DescriptionBox) member).calculateScore();
                        }
                    }
                }

                if (mOnCheckedMsg != null) {
                    AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
                    builder.setTitle(mOnCheckedMsgTitle)
                            .setMessage(mOnCheckedMsg)
                            .setPositiveButton("OK", new
DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface
dialogInterface, int i) {
                                    dialogInterface.dismiss();
                                }
                            });

                    AlertDialog dialog = builder.create();
                    dialog.show();
                }
            }
        }
    }

    @Override
    public boolean isValueChanged() {
        CheckBox checkBox = (CheckBox)getView();
        CheckBoxEntity checkBoxEntity = (CheckBoxEntity)getEntity();
        if (checkBox.isChecked() !=
                checkBoxEntity.isChecked()) {
            ((CheckBoxEntity) getEntity()).setChecked(checkBox.isChecked());
            return true;
        }
        return false;
    }
}
```

>> RADIOBOX COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.res.ColorStateList;
import android.graphics.Color;
import android.os.Build;
import android.view.ViewTreeObserver;
import android.widget.CompoundButton;
import android.widget.RadioButton;
import android.widget.RelativeLayout;

import org.ndayax.eflipmodule.db.entities.FlipBookEntity;
import org.ndayax.eflipmodule.db.entities.RadioBoxEntity;

import java.util.List;

public class RadioBox extends BaseComponent implements
CompoundButton.OnCheckedChangeListener {
    private boolean mCorrectValue;
    private int mScore;
    private String mOnCheckedMsg;
    private String mOnCheckedMsgTitle;

    protected RadioBox(Context context) {
        super(context, ComponentType.RADIO_BOX);
    }

    public void init(FlipBookEntity flipBookEntity, boolean isChecked, String
onCheckedMsg, String onCheckedMsgTitle) {
        initComponent(flipBookEntity);
        RadioButton radioButton = new RadioButton(getContext());
        radioButton.setChecked(((RadioBoxEntity)getEntity()).isChecked() ?
                ((RadioBoxEntity)getEntity()).isChecked() : isChecked);
        radioButton.setOnCheckedChangeListener(this);
        mOnCheckedMsg = onCheckedMsg;
        mOnCheckedMsgTitle = onCheckedMsgTitle;

        if(Build.VERSION.SDK_INT >= 21)
        {

            ColorStateList colorStateList = new ColorStateList(
                    new int[][]{
                            new int[]{-android.R.attr.state_enabled},
//disabled
                            new int[]{android.R.attr.state_enabled} //enabled
                    },
                    new int[] {
                            Color.LTGRAY //disabled
                            ,Color.DKGRAY //enabled
                    }
            );
            radioButton.setButtonTintList(colorStateList);//set the color tint
list
            radioButton.invalidate(); //could not be necessary
        }


        registerView(radioButton);
    }

    public void setChecked(boolean checked) {
        RadioButton radioButton = (RadioButton)getView();
        radioButton.setChecked(checked);
    }

    public void setCorrectValue(boolean correctValue) {
        mCorrectValue = correctValue;
    }

    public boolean getCorrectValue() {
        return mCorrectValue;
    }

    public void setScore(int score) {
        mScore = score;
    }

    @Override
    public void resizeLayout(float aspectRatio, boolean force) {
        super.resizeLayout(aspectRatio, force);
        RadioButton radioButton = (RadioButton)getView();

        radioButton.getViewTreeObserver().addOnGlobalLayoutListener(new
ViewTreeObserver.OnGlobalLayoutListener() {
            @Override
            public void onGlobalLayout() {

radioButton.getViewTreeObserver().removeOnGlobalLayoutListener(this);

                int w = radioButton.getWidth();
                int h = radioButton.getHeight();

                float scaleFactor = Math.min(((getWidth() + 10.f) *
aspectRatio)/(float)w,
                            ((getHeight() + 10.f) * aspectRatio)/(float)h);

                radioButton.setScaleX(scaleFactor);
                radioButton.setScaleY(scaleFactor);

                float x = ((getWidth() * aspectRatio) - w) / 2.f;
                float y = ((getHeight()  * aspectRatio) - h) / 2.f;

                RelativeLayout.LayoutParams params =
(RelativeLayout.LayoutParams)radioButton.getLayoutParams();
                params.leftMargin += (int)x;
                params.topMargin += (int)y;
                radioButton.setLayoutParams(params);
            }
        });
    }

    @Override
    protected boolean isValueChanged() {
        RadioButton radioButton = (RadioButton)getView();
        RadioBoxEntity radioBoxEntity = (RadioBoxEntity)getEntity();

        if (radioButton.isChecked() !=
                radioBoxEntity.isChecked()) {
            radioBoxEntity.setChecked(radioButton.isChecked());
            return true;
        }
        return false;
    }

    @Override
    public int getScore() {
        return mScore;
    }

    @Override
    public boolean isCorrectValue() {
        RadioButton radioButton = (RadioButton)getView();
        return radioButton.isChecked() == getCorrectValue();
    }

    @Override
    public boolean isSelected() {
        return isChecked();
    }

    public boolean isChecked() {
        RadioButton radioButton = (RadioButton)getView();
        return radioButton.isChecked();
    }

    /**
     * if the radiobox component is checked then remove the check from other
radio boxes in the same group
     * when checked, if the radiobox component has message, show the message.
     *
     * @param compoundButton
     * @param b If checked b is true.
     */
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
        if (b && isExamComponent()) {
            List<BaseComponent> members = getGroupMembers();
            for (int i = 0; i < members.size(); i++) {
                BaseComponent member = members.get(i);
                if (member != this) {
                    if (member instanceof RadioBox) {
                        ((RadioBox) member).setChecked(false);
                    } else if (member instanceof ScoringBox) {
                        ((ScoringBox) member).calculateExamScore();
                    } else if (member instanceof DescriptionBox) {
                        ((DescriptionBox) member).calculateScore();
                    }
                }
            }

            if (mOnCheckedMsg != null) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
                builder.setTitle("Message")
                        .setMessage(mOnCheckedMsg)
                        .setPositiveButton("OK", new
DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface
dialogInterface, int i) {
                                dialogInterface.dismiss();
                            }
                        });

                AlertDialog dialog = builder.create();
                dialog.show();
            }
        }
    }

    public static RadioBox create(Context context, FlipBookEntity
flipBookEntity,
                                  String id, int groupId, float left, float
top, float width, float height,
                                  boolean correctValue, int examId, String
examCategory, int score,
                                  String onCheckedMsg, String
onCheckedMsgTitle) {
        RadioBox radioBox = new RadioBox(context);
        radioBox.setId(id);
        radioBox.setGroupId(groupId);
        radioBox.setLeft(left);
        radioBox.setTop(top);
        radioBox.setWidth(width);
        radioBox.setHeight(height);
        radioBox.setCorrectValue(correctValue);
        radioBox.setExamId(examId);
        radioBox.setExamCategory(examCategory);
        radioBox.setScore(score);
        radioBox.init(flipBookEntity, false, onCheckedMsg, onCheckedMsgTitle);
        return radioBox;
    }
}
```

>> VIDEOBOX COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Handler;
import android.view.View;
import android.widget.VideoView;

import org.ndayax.eflipmodule.WebViewActivity;
import org.ndayax.eflipmodule.containers.FaPaVideoContainer;
import org.ndayax.eflipmodule.controllers.FaPaMediaController;
import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.utils.FileUtils;
import org.ndayax.eflipmodule.views.FaPaVideoView2;

public class VideoBox extends BaseComponent implements
MediaPlayer.OnCompletionListener {
    private boolean mRepeat;
    private boolean mAutoplay;
    private boolean mFullscreen;
    private FaPaMediaController mMediaController;
    private String mVideoUrl;
    private int mVideoCurrPos;

    public VideoBox(Context context) {
        super(context, ComponentType.VIDEO_BOX);
    }

    public void init(String filename) {
        if (!FileUtils.isFileExists(getContext(), filename)) {
            return;
        }

        FaPaVideoContainer videoContainer = new
FaPaVideoContainer(getContext());
        FaPaVideoView2 videoView = new FaPaVideoView2(getContext());
        videoView.setAnchorView(videoContainer);
        videoView.setDisplayMode(FaPaVideoView2.DisplayMode.FILL);
        mMediaController = new FaPaMediaController(getContext(), true);
        //videoView.setMediaController(mMediaController);

        mVideoUrl = FileUtils.getBooksFileUrl(getContext(), filename);
        videoView.setVideoUrl(mVideoUrl);
        mVideoCurrPos = 0;
        videoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
            @Override
            public void onPrepared(MediaPlayer mediaPlayer) {
                videoView.setMediaController(mMediaController);
            }
        });

        videoView.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mediaPlayer) {
                mVideoCurrPos = 0;
                videoView.seekTo(100);
            }
        });

        videoView.setOnMediaPlayerReleasedListener(new
FaPaVideoView2.OnMediaPlayerReleasedListener() {
```

```
            @Override
            public void onMediaPlayerReleased(View sender) {
                mVideoCurrPos = ((FaPaVideoView2)sender).getCurrentPosition();
            }
        });


videoView.setOnClickFullScreenListener((WebViewActivity)ActivityMgr.getInstanc
e().getCurrentActivity());
        videoContainer.attachVideoView(videoView);
        registerView(videoContainer);
        //view.setOnClickListener(new View.OnClickListener() {
        //    @Override
        //    public void onClick(View view) {
        //        mMediaController.setVisibility(View.VISIBLE);
        //        play();
        //    }
        //});
    }

    public void setRepeat(boolean repeat) {
        mRepeat = repeat;
    }

    public boolean getRepeat() {
        return mRepeat;
    }

    public void setAutoplay(boolean autoplay) {
        mAutoplay = autoplay;
    }

    public boolean getAutoplay() {
        return mAutoplay;
    }

    public void showMediaController() {
        mMediaController.setVisibility(View.VISIBLE);
    }

    public void play() {
        FaPaVideoView2 videoView = (FaPaVideoView2)
((FaPaVideoContainer)getView()).getVideoView();
        if (!videoView.isPlaying()) {
            videoView.seekTo(mVideoCurrPos);
            videoView.start();
        }
    }

    @Override
    public void show() {
        FaPaVideoView2 videoView = (FaPaVideoView2)
((FaPaVideoContainer)getView()).getVideoView();
        if (!videoView.isPlaying()) {
            videoView.seekTo(mVideoCurrPos == 0 ? 100 : mVideoCurrPos);
        }
        //VideoView view = (VideoView) getView();
        //view.seekTo(100);
        //FaPaVideoView2 view = (FaPaVideoView2) getView();
        //view.seekTo(100);
        //mMediaController.hide();

        super.show();
    }

    @Override
    public void stop() {
        FaPaVideoView2 videoView = (FaPaVideoView2)
((FaPaVideoContainer)getView()).getVideoView();
        if (videoView.isPlaying()) {
            videoView.stopPlayback();
        }
        videoView.suspend();
        super.stop();


        //VideoView view = (VideoView) getView();
        //if (view.isPlaying()) {
        //    view.stopPlayback();
        //}
        //FaPaVideoView2 view = (FaPaVideoView2) getView();
        //if (view.isPlaying()) {
        //    view.stopPlayback();
        //}
    }

    @Override
    public void start() {
        super.start();
        //play();
        //if (!mAutoplay) {
            //    VideoView view = (VideoView) getView();
        //    new Handler().postDelayed(new Runnable() {
        //        @Override
        //        public void run() {
        //
ActivityMgr.getInstance().getCurrentActivity().runOnUiThread(new Runnable() {
        //                @Override
        //                public void run() {
        //                    view.pause();
        //                }
        //            });
        //        }
        //    }, 5);
        //}
    }

    @Override
    public void hide() {
        FaPaVideoView2 videoView = (FaPaVideoView2)
((FaPaVideoContainer)getView()).getVideoView();
        if (videoView.isPlaying()) {
            videoView.stopPlayback();
        }
        super.hide();
    }

    public static VideoBox create(Context context, String id, float l, float
t, float w, float h,
                                  boolean repeat, boolean autoplay, String
filename) {
        VideoBox videoBox = new VideoBox(context);
        videoBox.setId(id);
        videoBox.setLeft(l);
        videoBox.setTop(t);
        videoBox.setWidth(w);
        videoBox.setHeight(h);
        videoBox.setRepeat(repeat);
        videoBox.setAutoplay(autoplay);
        videoBox.init(filename);
        return videoBox;
    }

    @Override
    protected float getProportion() {
        return 1.5f;
    }

    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
        if (mRepeat) {
            start();
        }
    }
}
```

>>GRAPHBOX COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;
import android.graphics.Bitmap;
import android.provider.ContactsContract;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.ImageButton;

import org.ndayax.eflipmodule.R;
import org.ndayax.eflipmodule.db.entities.FlipBookEntity;
import org.ndayax.eflipmodule.models.FlipBookPage;
import org.ndayax.eflipmodule.utils.BitmapUtils;
import org.ndayax.eflipmodule.views.GraphView;

public class GraphBox extends BaseComponent implements View.OnClickListener {
    private Bitmap mBackground = null;
    private int mMaxPoints = 2;

    protected GraphBox(Context context) {
        super(context, ComponentType.GRAPH_BOX);
    }

    /**
     * initialize the GraphBox component.
     *
     * @param flipBookEntity
     * @param bg_filename, The background
     * @param buttons, String position of button
     */
    public void init(FlipBookEntity flipBookEntity,
                     String bg_filename, String buttons) {
        initComponent(flipBookEntity);

        if (bg_filename != null &&
                bg_filename.length() > 0) {
            mBackground = BitmapUtils.loadBitmap(getContext(), bg_filename);
        }

        LayoutInflater inflater = LayoutInflater.from(getContext());
        View view ;
        if (buttons.equals("top")) {
            view = inflater.inflate(R.layout.graph_activity_view_top, null);
        } else {
            view = inflater.inflate(R.layout.graph_activity_view_bottom,
null);
        }

        ImageButton drawBtn = view.findViewById(R.id.draw_button);
        ImageButton selectBtn = view.findViewById(R.id.select_button);
        ImageButton pickColorBtn = view.findViewById(R.id.pick_color_button);
        ImageButton deleteBtn = view.findViewById(R.id.delete_button);
        ImageButton penSizeBtn = view.findViewById(R.id.line_size_button);

        GraphView graphView = view.findViewById(R.id.graph_view);
        //graphView.restoreLayers();

        drawBtn.setOnClickListener(this);
        selectBtn.setOnClickListener(this);
        pickColorBtn.setOnClickListener(this);
        deleteBtn.setOnClickListener(this);
        penSizeBtn.setOnClickListener(this);

//        GraphView graphView = new GraphView(getContext());
        graphView.setBackgroundBitmap(mBackground);
        graphView.setTag(this);
        registerView(view);
    }

    public GraphView getGraphView() {
        if (getView() == null) {
            return null;
        }

        return (GraphView)getView().findViewById(R.id.graph_view);
    }
    public void setMaxPoints(int maxPoints) {
        mMaxPoints = maxPoints;
    }

    public int getMaxPoints() {
        return mMaxPoints;
    }

    @Override
    protected boolean isValueChanged() {
        return true;
    }

    public static GraphBox create(Context context, FlipBookEntity
flipBookEntity,
                                  FlipBookPage page, String id, int groupId,
float left, float top,
                                  float width, float height, int maxPoints,
                                  int examId, String examCategory,
                                  String bg_filename, String buttons,
BaseComponent hiddenComponent) {
        GraphBox graphBox = new GraphBox(context);
        graphBox.setId(id);
        graphBox.setGroupId(groupId);
        graphBox.setLeft(left);
        graphBox.setTop(top);
        graphBox.setWidth(width);
        graphBox.setHeight(height);
        graphBox.setMaxPoints(maxPoints);
        graphBox.setExamId(examId);
        graphBox.setExamCategory(examCategory);
        graphBox.setHiddenComponent(hiddenComponent);
        graphBox.setPage(page);
        graphBox.init(flipBookEntity, bg_filename, buttons);
        return graphBox;
    }

    public String getImageFilename() {
        return getPage().getParent().getGraphImagesFolder() + "/" + getId() +
".jpg";
    }
    @Override
    public void stop() {
        GraphView graphView = getGraphView();
        String folder = getPage().getParent().getGraphImagesFolder();
        graphView.saveGraphBitmap(folder + "/" + getId() + ".jpg") ;
        super.stop();
    }

    @Override
    public void onClick(View view) {
        GraphView graphView = getView().findViewById(R.id.graph_view);
        switch (view.getId()) {
            case R.id.draw_button:
graphView.setActionType(GraphView.ActionType.DRAWING); break;
            case R.id.select_button:
graphView.setActionType(GraphView.ActionType.SELECT); break;
```

```
            case R.id.delete_button: graphView.removeLayer(); break;
            case R.id.pick_color_button:
graphView.showPickPenColorDialog();break;
            case R.id.line_size_button: graphView.showPenSizeDialog(); break;
            default: break;
        }
    }
}
```

>> POPUPBOX COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;
import android.util.Log;

import org.json.JSONException;
import org.json.JSONObject;
import org.ndayax.eflipmodule.db.entities.FlipBookEntity;
import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.models.FlipBookPage;
import org.ndayax.eflipmodule.views.PopupView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;

public class PopupBox extends BaseComponent{

    private PopupView mPopupView;
    //private BaseComponent _popupComponent;

    public PopupBox(Context context) {
        super(context, ComponentType.POPUP_BOX);
    }

    public void init(BaseComponent component) {
        mPopupView = new PopupView(component);
        //_popupComponent = component;
    }

    @Override
    public void show() {
        ActivityMgr.getInstance().getCurrentActivity().runOnUiThread(new
Runnable() {
            @Override
            public void run() {
                AppCompatActivity activity =
(AppCompatActivity)ActivityMgr.getInstance().getCurrentActivity();
                FragmentManager fm = activity.getSupportFragmentManager();
                //PopupView popupView = new PopupView(_popupComponent);
                try {
                    Fragment fragment = fm.findFragmentByTag("Popup");
                    if (fragment != null && fragment.isAdded()) {
                        fm.beginTransaction().remove(fragment).commit();
                    }
                    fm.executePendingTransactions();
                    mPopupView.show(fm, "Popup");
                } catch (Exception e) {
                    Log.d("PopupBox", e.getMessage());
                }
            }
        });
    }

    public static PopupBox create(Context context, FlipBookEntity
flipBookEntity, FlipBookPage page,
                                  JSONObject jsonObject, String root, String
id) throws JSONException {
        PopupBox popupBox = new PopupBox(context);
        popupBox.setId(id);
        BaseComponent component =
BaseComponent.createComponentFromJson(context,
                flipBookEntity, page, jsonObject, root);
        popupBox.init(component);
        return popupBox;
    }
}
```

>> SHAREBUTTON COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;

import org.ndayax.eflipmodule.interfaces.OnClickShareButtonListener;
import org.ndayax.eflipmodule.models.FlipBook;
import org.ndayax.eflipmodule.models.FlipBookPage;

public class ShareButton extends BaseComponent {
    private boolean mWhatsAppOnly;
    private FlipBookPage.PageType mPageType;
    private OnClickShareButtonListener mListener;
    private String mPageNo;

    public ShareButton(Context context) {
        super(context, ComponentType.SHARE_BUTTON);

        mWhatsAppOnly = false;
        mPageType = FlipBookPage.PageType.NONE;
        mListener = null;
        mPageNo = "";
    }

    public FlipBookPage.PageType getPageType() {
        return mPageType;
    }

    public void setPageType(FlipBookPage.PageType pageType) {
        mPageType = pageType;
    }

    public void setPageType(String pageTypeStr) {
        mPageType = FlipBookPage.convertString(pageTypeStr);
```

```java
    }

    public boolean isWhatsAppOnly() {
        return mWhatsAppOnly;
    }

    public void setWhatsAppOnly(boolean whatsAppOnly) {
        mWhatsAppOnly = whatsAppOnly;
    }

    public void setPageNo(String pageNo) {
        mPageNo = pageNo;
    }

    public String getPageNo() {
        return mPageNo;
    }

    public void setOnClickShareButtonListener(OnClickShareButtonListener
listener) {
        mListener = listener;
    }

    @Override
    public void buttonClicked() {
        if (mListener != null) {
            if (mPageType == FlipBookPage.PageType.NONE) {
                if (getPageNo() == "") {
                    if (getPage() != null) {
                        mListener.sharePage(getPage(), mWhatsAppOnly);
                    }
                } else {
                    mListener.sharePagesEx(getPageNo(), mWhatsAppOnly);
                }
            } else {
                mListener.sharePages(mPageType, mWhatsAppOnly);
            }
        }
    }

    public static ShareButton create(Context context, FlipBookPage page,
                                     String id, int groupId, float left, float
top, float width,
                                     float height, String pageTypeStr, String
pageNo, boolean whatsAppOnly) {
        ShareButton shareButton = new ShareButton(context);
        shareButton.setId(id);
        shareButton.setGroupId(groupId);
        shareButton.setLeft(left);
        shareButton.setTop(top);
        shareButton.setWidth(width);
        shareButton.setHeight(height);
        shareButton.setPageType(pageTypeStr);
        shareButton.setPage(page);
        shareButton.setWhatsAppOnly(whatsAppOnly);
        shareButton.setPageNo(pageNo);
        return shareButton;
    }
}
```

>> POPUP BUTTON COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;

import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.interfaces.OnClickPopupButtonListener;
import org.ndayax.eflipmodule.models.FlipBookPage;

import java.util.List;

public class PopupButton extends BaseComponent {

    private OnClickPopupButtonListener mListener;
    private String mPopupComponent;
    private String mErrMsg;

    public PopupButton(Context context) {
        super(context, ComponentType.POPUP_BUTTON);
        mListener = null;
    }

    public void setOnClickPopupListener(OnClickPopupButtonListener listener) {
        mListener = listener;
    }

    public void setPopupComponent(String popupComponent) {
        mPopupComponent = popupComponent;
    }

    public String getPopupComponent() {
        return mPopupComponent;
    }

    public void setErrorMessage(String errMsg) {
        mErrMsg = errMsg;
    }

    public String getErrorMessage() {
        return mErrMsg;
    }

    @Override
    public void buttonClicked() {
        if (mListener != null) {
            boolean cont = true;
            if (mErrMsg != null) {
                List<BaseComponent> groups = getGroupMembers();
                int i = 0;
                cont = false;
                for (;i < groups.size(); i++) {
                    BaseComponent component = groups.get(i);
                    if (component == this)
                        continue;

                    if (component.getExamId() > 0 && component.isEmpty())
                        break;
                }
                if (i >= groups.size()) {
                    cont = true;
                } else {
                    ActivityMgr.getInstance().showAlertDialog("Pesan
Kesalahan", mErrMsg);
                }
            }

            if (cont) {
                mListener.showPopup(getPopupComponent());
            }
        }
    }

    public static PopupButton create(Context context, FlipBookPage page,
                                     String id, int groupId, float left, float
top,
                                     float width, float height,
                                     String popupComponent, String errMsg) {
        PopupButton popupButton = new PopupButton(context);
        popupButton.setId(id);
        popupButton.setGroupId(groupId);
        popupButton.setPage(page);
        popupButton.setLeft(left);
        popupButton.setTop(top);
        popupButton.setWidth(width);
        popupButton.setHeight(height);
        popupButton.setPopupComponent(popupComponent);
        popupButton.setErrorMessage(errMsg);
        return popupButton;
    }
}
```

>> SHOWHIDDENBUTTON COMPONENT

```
package org.ndayax.eflipmodule.components;

import android.content.Context;

import org.ndayax.eflipmodule.interfaces.OnClickShowHiddenButtonListener;
import org.ndayax.eflipmodule.models.FlipBookPage;

public class ShowHiddenButton extends BaseComponent {
    private String mComponentId;
    private int mComponentGroup;
    private long mDelay;
    private OnClickShowHiddenButtonListener mListener;

    public ShowHiddenButton(Context context) {
        super(context, ComponentType.SHOW_HIDDEN_BUTTON);
        mListener = null;
        mComponentId = "";
        mComponentGroup = -1;
        mDelay = 1000;
    }

    public void
setOnClickShowHiddenButtonListener(OnClickShowHiddenButtonListener listener) {
        mListener = listener;
    }

    public void setComponentId(String componentId) {
        mComponentId = componentId;
    }

    public String getComponentId() {
        return mComponentId;
    }

    public void setComponentGroup(int componentGroup) {
        mComponentGroup = componentGroup;
    }

    public int getComponentGroup() {
        return mComponentGroup;
    }

    public void setDelay(long delay) {
        mDelay = delay;
    }

    public long getDelay() {
        return mDelay;
    }

    /**
     * Show hidden component when button click
     */
    @Override
    public void buttonClicked() {
        if (mListener != null) {
            if (!mComponentId.equals("") || mComponentId.length() > 0) {
                mListener.showHidden(mComponentId, mDelay);
            } else if (mComponentGroup > 0) {
                mListener.showHidden(mComponentGroup, mDelay);
            }
        }
    }

    public static ShowHiddenButton create(Context context, FlipBookPage page,
float left, float top,
                                          float width, float height, String
componentId, int componentGroup,
                                          long delay) {

        ShowHiddenButton showHiddenButton = new ShowHiddenButton(context);
        showHiddenButton.setPage(page);
        showHiddenButton.setLeft(left);
        showHiddenButton.setTop(top);
        showHiddenButton.setWidth(width);
        showHiddenButton.setHeight(height);
        showHiddenButton.setComponentId(componentId);
        showHiddenButton.setComponentGroup(componentGroup);
        showHiddenButton.setDelay(delay);
        return showHiddenButton;
    }


}
```

VIEW MODULES

>> FLIPBOOKVIEW

```
package org.ndayax.eflipmodule.views;

import android.annotation.TargetApi;
import android.content.Context;
import android.content.res.Configuration;
import android.os.Build;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.RelativeLayout;

import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.interfaces.JavascriptInterfaces;
import org.ndayax.eflipmodule.models.FlipBook;
import org.ndayax.eflipmodule.models.FlipBookException;
import org.ndayax.eflipmodule.utils.FileUtils;

import androidx.annotation.RequiresApi;

public class FlipBookView extends WebView  {
    private static final String TAG = FlipBookView.class.getSimpleName();
    private Context _context;

    private FlipBook _flipBook;
    private String _bookUrl;

    public FlipBookView(Context context) {
        super(getFixedContext(context));
        _context = context;
    }

    public FlipBookView(Context context, AttributeSet attrs) {
        super(getFixedContext(context), attrs);
    }

    public FlipBookView(Context context, AttributeSet attrs, int defStyleAttr)
{
        super(getFixedContext(context), attrs, defStyleAttr);
    }
    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public FlipBookView(Context context, AttributeSet attrs, int defStyleAttr,
int defStyleRes) {
        super(getFixedContext(context), attrs, defStyleAttr, defStyleRes);
    }

    public static Context getFixedContext(Context context) {
        return context.createConfigurationContext(new Configuration());
    }

    @Override
    public void onPause() {
        _flipBook.saveCurrentPage();
        super.onPause();
    }

    @Override
    public void destroy() {
        _flipBook.saveCurrentPage();
        super.destroy();
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    public void initBook(RelativeLayout pageLayout, String folder) {
        _bookUrl = FileUtils.getBooksFileUrl(getContext(), folder) +
"/index.html";
        WebSettings settings = getSettings();
        settings.setJavaScriptEnabled(true);
        settings.setAllowUniversalAccessFromFileURLs(true);
        settings.setAllowFileAccess(true);
        clearCache(true);
        settings.setCacheMode(WebSettings.LOAD_NO_CACHE);
        setWebViewClient(new FlipBookViewClient());

        _flipBook = new
FlipBook(ActivityMgr.getInstance().getCurrentActivity(),
                pageLayout);

        try {
            String bookFolder =
FileUtils.getBooksAbsolutePathFor(getContext(), folder);
            _flipBook.create(this, folder, bookFolder + "/book_configs.json");
        } catch (FlipBookException e) {
            Log.d(TAG, e.getMessage());
        }

        addJavascriptInterface(new JavascriptInterfaces(getContext(),
_flipBook),
                "AndroidJS");
    }

    public void loadView(int page) {
        loadUrl(_bookUrl + "#p=" + Integer.toString(page));
    }

    public boolean isZoomIn() {
        return _flipBook.isPageZoomin();
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_MOVE:
_flipBook.movePage(event.getAction()); break;
            case MotionEvent.ACTION_UP: _flipBook.movePage(event.getAction());
break;
            default: break;
        }

        return super.onTouchEvent(event);
    }

    public  FlipBook getBook() {
        return _flipBook;
    }
}
```

>> GRAPHVIEW

```
package org.ndayax.eflipmodule.views;

import android.annotation.TargetApi;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.res.Configuration;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Point;
import android.os.Build;
import android.text.InputType;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import org.ndayax.eflipmodule.components.GraphBox;
import org.ndayax.eflipmodule.db.FlipBookDb;
import org.ndayax.eflipmodule.db.entities.GraphBoxEntity;
import org.ndayax.eflipmodule.db.entities.LayerEntity;
import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.utils.BitmapUtils;
import org.ndayax.eflipmodule.utils.graph.Layer;
import org.ndayax.eflipmodule.utils.graph.Pen;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import yuku.ambilwarna.AmbilWarnaDialog;

public class GraphView extends View {


    private static final String TAG = GraphView.class.getSimpleName();

    public enum ActionType {
        DRAWING,
        SELECT
    }

    private Paint _canvasPaint = new Paint(Paint.DITHER_FLAG |
Paint.ANTI_ALIAS_FLAG);
    private List<Layer> _layers = new ArrayList<>();
    private int _width;
    private int _height;
    private Layer _activeLayer;
    private Canvas _canvas = new Canvas();
    private ActionType _actionType = ActionType.DRAWING;

    public GraphView(Context context) {
        super(getFixedContext(context));
    }

    public GraphView(Context context, AttributeSet attrs) {
        super(getFixedContext(context), attrs);
    }

    public GraphView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(getFixedContext(context), attrs, defStyleAttr);
    }

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public GraphView(Context context, AttributeSet attrs, int defStyleAttr,
int defStyleRes) {
        super(getFixedContext(context), attrs, defStyleAttr, defStyleRes);
    }

    public static Context getFixedContext(Context context) {
        return context.createConfigurationContext(new Configuration());
    }

    private Layer getActiveLayer(Point pt) {
        if (_layers.size() > 0) {
            for (int i = 0; i < _layers.size(); i++) {
                Layer layer = _layers.get(i);
                if (layer.contains(pt) && !(layer.getLayerType() ==
Layer.LayerType.BACKGROUND)) {
                    return layer;
                }
            }
        }
        return null;
    }

    private Layer getBackgroundLayer(List<Layer> layers) {
```

```java
        for (int i = 0; i < layers.size(); i++) {
            Layer layer = layers.get(i);
            if (layer.getLayerType() == Layer.LayerType.BACKGROUND) {
                return layer;
            }
        }
        return null;
    }

    @Override
    public void draw(Canvas canvas) {
        super.draw(canvas);

        try {
            canvas.drawColor(Color.WHITE);
            for (int i = 0; i < _layers.size(); i++) {
                Bitmap bitmap = _layers.get(i).getBitmap();
                canvas.drawBitmap(bitmap, 0, 0, _canvasPaint);
            }
        } catch (Exception e) {
            e.getStackTrace();
        }
    }

    private GraphBoxEntity getGraphBoxEntity() {
        return (GraphBoxEntity)getGraphBox().getEntity();
    }

    private GraphBox getGraphBox() {
        return (GraphBox)getTag();
    }

    private Layer _selected = null;

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        int xPos = (int) (event.getX() + 0.5f);
        int yPos = (int) (event.getY() + 0.5f);

        Point pt = new Point(xPos, yPos);

        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                Log.d(TAG, "on touch down x: " + Integer.toString(xPos) +
                        ", y: " + Integer.toString(yPos));

                releaseActiveLayer();

                _activeLayer = getActiveLayer(pt);
                if (_activeLayer != null) {
                    _activeLayer.bindCanvas(_canvas);
                    _activeLayer.drawLine(true, 1.f);
                    _canvas.setBitmap(_activeLayer.getBitmap());

                    if (_actionType == ActionType.SELECT) {
                        _selected = _activeLayer;
                        invalidate();
                    }
                } else
                if (_activeLayer == null) {
                    if (_actionType == ActionType.DRAWING) {
                        _activeLayer = new Layer(_width, _height,
getGraphBox().getMaxPoints());
                        _activeLayer.bindCanvas(_canvas);
                        LayerEntity layerEntity = new
LayerEntity(getGraphBoxEntity());
                        layerEntity.setLayer(_activeLayer);
                        _activeLayer.setLayerEntity(layerEntity);
                        getGraphBoxEntity().addLayer(layerEntity);
                        _layers.add(_activeLayer);
                        _activeLayer.addPoint(pt);
                    }
                }
                break;
            case MotionEvent.ACTION_MOVE:
                if (_actionType == ActionType.DRAWING) {
                    _activeLayer.addPoint(pt);
                    _activeLayer.drawLine(true, 1.f);
                    invalidate();
                }
                break;
            case MotionEvent.ACTION_UP:
                if (_actionType == ActionType.DRAWING) {
                    _activeLayer.addPoint(pt);
                    _activeLayer.drawLine(true, 1.f);
                    invalidate();
                }
                break;
            default:
                break;
        }
        return true;
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);
        initGraphLayers(w, h);
    }

    private Bitmap _backgroundBitmap = null;

    private void initGraphLayers(int w, int h) {
        _width = w;
        _height = h;

        for (int i = 0; i < _layers.size(); i++) {
            Layer layer = _layers.get(i);
            layer.resize(w, h);
        }

        doSetBackground(_layers, w, h, _canvas, _backgroundBitmap);
        restoreLayers(_layers, _canvas, 1.f);
    }
    private void doSetBackground(List<Layer> layers, int w, int h, Canvas
canvas, Bitmap bgBitmap) {
        if (bgBitmap != null) {
            Layer backgroundLayer = getBackgroundLayer(layers);
            if (backgroundLayer == null) {
                backgroundLayer = new Layer(w, h, Layer.LayerType.BACKGROUND);
                backgroundLayer.setBitmap(bgBitmap, canvas);
                layers.add(backgroundLayer);
            }
        }
    }

    public void restoreLayers(List<Layer> layers, Canvas canvas, float
aspectRatio) {
        GraphBoxEntity entity = getGraphBoxEntity();
        if (entity != null && (entity.getLayers().size() > 0)) {
            for (int i = 0; i < entity.getLayers().size(); i++) {
                LayerEntity layerEntity = entity.getLayers().get(i);
                Layer layer = layerEntity.getLayer();
                if (layer != null &&
                        !layers.contains(layer)) {
                    layer.getPen().setStrokeWidth(layerEntity.getPenSize());
                    layer.getPen().setColor(layerEntity.getPenColor());
                    layer.bindCanvas(canvas);
                    layer.drawLine(false, aspectRatio);
                    layers.add(layer);


Pen.getInstance().setStrokeWidth(layerEntity.getPenSize());
                    Pen.getInstance().setColor(layerEntity.getPenColor());
                }
            }
        }
    }

    public void setBackgroundBitmap(int rawId) {
        InputStream is = null;
        try {
            is = getContext().getResources().openRawResource(rawId);
            _backgroundBitmap = BitmapFactory.decodeStream(is);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (is != null) {
                try {
                    is.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }

    public void setBackgroundBitmap(Bitmap bitmap) {
        _backgroundBitmap = bitmap;
    }
    public void releaseActiveLayer() {
        if (_activeLayer != null) {
            _activeLayer.drawLine(false, 1.f);
            _activeLayer.releaseCanvas();
        }
        _activeLayer = null;
    }

    public void saveGraphBitmap(String filename) {
        Bitmap bitmap = getGraphBitmap();
        BitmapUtils.saveBitmap(getContext(), bitmap, filename);
    }

    public Bitmap getGraphBitmap() {
        Bitmap bitmap = Bitmap.createBitmap(_width, _height,
Bitmap.Config.ARGB_8888);
        Canvas canvas = new Canvas();
        canvas.setBitmap(bitmap);
        canvas.drawColor(Color.WHITE);
        for (int i = 0; i < _layers.size(); i++) {
            _layers.get(i).drawLine(false, 1.f);
            Bitmap bmp = _layers.get(i).getBitmap();
            canvas.drawBitmap(bmp, 0, 0, _canvasPaint);
        }
        canvas.setBitmap(null);
        return bitmap;
    }

    public Bitmap getGraphBitmap(int w, int h, float aspectRatio) {
        // create layer ..
        List<Layer> layers = new ArrayList<>();
        Canvas canvas = new Canvas();
        Paint paint = new Paint(Paint.DITHER_FLAG | Paint.ANTI_ALIAS_FLAG);
        // add background image to layers ...
        Bitmap bg = BitmapUtils.resizeBitmap(_backgroundBitmap, w, h);
        doSetBackground(layers, w, h, canvas, bg);
        restoreLayers(layers, canvas, aspectRatio);
        Bitmap bitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
        canvas.setBitmap(bitmap);
        for (int i = 0; i < layers.size(); i++) {
            Layer layer = layers.get(i);
            Bitmap scalableBitmap =
BitmapUtils.resizeBitmap(layer.getBitmap(), w, h);
            canvas.drawBitmap(scalableBitmap, 0, 0, paint);
        }
        canvas.setBitmap(null);
        return bitmap;
    }

    public void setActionType(ActionType actionType) {
        _actionType = actionType;
    }

    public ActionType getActionType() {
        return _actionType;
    }
    public void removeLayer() {
        if (_activeLayer != null) {
            _activeLayer.releaseCanvas();
            _layers.remove(_activeLayer);
            if (_activeLayer.getLayerEntity() != null) {

_activeLayer.getLayerEntity().setState(FlipBookDb.dbState.DELETE);
            }
            _activeLayer = null;
            invalidate();
        }
    }

    public void showPenSizeDialog() {
        EditText input = new EditText(getContext());
        input.setInputType(InputType.TYPE_CLASS_NUMBER);
        AlertDialog dialog = new
AlertDialog.Builder(ActivityMgr.getInstance().getCurrentActivity())
                .setTitle("Pen Size Dialog")
                .setMessage("Get Pen Size")
                .setView(input)
                .setPositiveButton("OK", new DialogInterface.OnClickListener()
{
                    @Override
                    public void onClick(DialogInterface dialogInterface, int
i) {
                        try {
                            double penSize =
Double.parseDouble(input.getText().toString());
                            Pen.getInstance().setStrokeWidth((float)penSize);
                        } catch (Exception e) {
                            Log.d(TAG, "error parse double, pen size
dialog!");
                        }
                    }
                })
                .setNegativeButton("Cancel", null)
                .create();
        dialog.show();
    }

    public void showPickPenColorDialog() {
        AmbilWarnaDialog dialog = new
AmbilWarnaDialog(ActivityMgr.getInstance().getCurrentActivity(),
                Pen.getInstance().getColor(),
                new AmbilWarnaDialog.OnAmbilWarnaListener() {
                    @Override
                    public void onOk(AmbilWarnaDialog dialog, int color) {
                        // color is the color selected by the user.
                        Pen.getInstance().setColor(color);
                    }

                    @Override
                    public void onCancel(AmbilWarnaDialog dialog) {
                        // cancel was selected by the user
                    }
                });
        dialog.show();
    }

    public void toggleAction() {
        if (_actionType == ActionType.DRAWING) {
            _actionType = ActionType.SELECT;

            Toast.makeText(getContext(), "Select Mode",
Toast.LENGTH_LONG).show();
        } else {
            _actionType = ActionType.DRAWING;

            Toast.makeText(getContext(), "Drawing Mode",
Toast.LENGTH_LONG).show();
        }
    }
}
```

>> PICTUREVIEW

```
package org.ndayax.eflipmodule.views;

import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Context;
import android.content.res.Configuration;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.os.Build;
import android.util.AttributeSet;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;

import org.ndayax.eflipmodule.components.CameraBox;
import org.ndayax.eflipmodule.helpers.ActivityMgr;
import org.ndayax.eflipmodule.utils.BitmapUtils;

public class PictureView extends View{
    private String mPictureFolder;
    private String mFilePath;
    private int mWidth;
    private int mHeight;

    public PictureView(Context context) {
        super(getFixedContext(context));
    }

    public PictureView(Context context, AttributeSet attrs) {
        super(getFixedContext(context), attrs);
    }

    public PictureView(Context context, AttributeSet attrs, int defStyleAttr)
{
        super(getFixedContext(context), attrs, defStyleAttr);
    }
    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public PictureView(Context context, AttributeSet attrs, int defStyleAttr,
int defStyleRes) {
        super(getFixedContext(context), attrs, defStyleAttr, defStyleRes);
    }

    public static Context getFixedContext(Context context) {
        return context.createConfigurationContext(new Configuration());
    }

    public void setPictureFolder(String pictureFolder) {
        mPictureFolder = pictureFolder;
    }

    public String getPictureFolder() {
        return mPictureFolder;
    }

    public void setFilePath(String filePath) {
        mFilePath = filePath;
    }

    public String getFilePath() {
        return mFilePath;
    }
    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);

        mWidth = w;
        mHeight = h;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        String bmpFilename;
        if (mFilePath != null && mFilePath.length() > 0) {
            bmpFilename = getPictureFolder() + "/" + mFilePath;
        } else {
            CameraBox cameraBox = (CameraBox) getTag();
            bmpFilename = getPictureFolder() + "/" + cameraBox.getId() +
".jpg";
        }
        Bitmap bitmap = BitmapUtils.loadBitmap(getContext(), bmpFilename);
        Bitmap scaled = Bitmap.createScaledBitmap(bitmap, mWidth, mHeight,
true);
        canvas.drawBitmap(scaled, 0, 0, new Paint(Paint.DITHER_FLAG));
        scaled.recycle();
        bitmap.recycle();
    }


    public Bitmap getBitmap() {
        CameraBox cameraBox = (CameraBox) getTag();
        if (cameraBox == null)
            return null;
        return BitmapUtils.loadBitmap(getContext(), getPictureFolder() + "/" +
cameraBox.getId() + ".jpg");
    }
}
```

>> POPUPVIEW

```
package org.ndayax.eflipmodule.views;

import android.app.Dialog;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.FrameLayout;
import android.widget.ImageButton;

import org.ndayax.eflipmodule.R;
import org.ndayax.eflipmodule.components.BaseComponent;
import org.ndayax.eflipmodule.helpers.ActivityMgr;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.DialogFragment;

public class PopupView extends DialogFragment {

    private BaseComponent _component = null;
    private View _contentView = null;

    public PopupView(BaseComponent component) {
        setComponent(component);
    }

    public void setComponent(BaseComponent component) {
        _component = component;
    }

    public BaseComponent getComponent() {
        return _component;
    }

    @Override
    public void onStart() {
        super.onStart();

        Dialog dialog = getDialog();
        if (dialog != null)
        {
            DisplayMetrics displayMetrics = new DisplayMetrics();
            ActivityMgr.getInstance().getDisplayMetrics(displayMetrics);
            int width = (int)  (displayMetrics.widthPixels * 0.9f);
            int height = (int) (displayMetrics.heightPixels * 0.75f);
            dialog.getWindow().setLayout(width, height);
            dialog.setCanceledOnTouchOutside(false);
            getComponent().start();

            dialog.getWindow().getDecorView().setSystemUiVisibility(
                    View.SYSTEM_UI_FLAG_LAYOUT_STABLE
                    | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
                    | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
                    | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
                    | View.SYSTEM_UI_FLAG_FULLSCREEN
                    | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY);


dialog.getWindow().addFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS
);
        }
    }

    @Override
    public void onResume() {
        super.onResume();
    }

    @Override
    public void onStop() {
        Log.d("PopupView", "PopupView onStop()");
        getComponent().stop();
        super.onStop();
    }

    @Override
    public void onPause() {
        Log.d("PopupView", "PopupView onPause()");
        super.onPause();
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
                             @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        _contentView = super.onCreateView(inflater, container,
savedInstanceState);
        if (_contentView == null &&
                getComponent() != null) {
            _contentView = inflater.inflate(R.layout.popup_activity_view,
null);
            FrameLayout contentView =
(FrameLayout)_contentView.findViewById(R.id.content_view);
            contentView.addView(getComponent().getView());
            ImageButton button =
(ImageButton)_contentView.findViewById(R.id.button_close);
            button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    dismiss();
                }
            });
        }
        return _contentView;
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState)
{
        super.onViewCreated(view, savedInstanceState);
    }
}
```

Contoh book_configs.json

```
{
  "book_title": "PROGRAM LINEAR",
  "page_width": 595.28,
  "page_height": 841.89,
  "pages": [
     {
      "page": 1  -→ halaman dari buku anda, dimulai dari 1
      "page_type":
      "reading"/"self_examination"/"examination"/"exercise"/"eval
      uation" → optional atau pilih salah satu, default adalah
      "reading".
      "components": [
        {
          "type": "editbox",
          "id": "edt1",
          "left": 0,
          "top": 0,
          "width": 595.28,
          "height": 841.89,
          "text_size": 22,
          "default_text": "123",
          "text": "",
          …
        },
        {
          "type": "editbox",
          "id": "edt2",
          "left": 0,
          "top": 0,
          "width": 595.28,
          "height": 841.89,
          "text_size": 22,
          “default_text”: “123”,
          “text”: “”,
        }
        ]
      },
      {
      “page”: 2  -→ halaman dari buku anda, dimulai dari 1
      “page_type”:
      “reading”/”self_examination”/”examination”/”exercise”/”eval
      uation” → optional atau pilih salah satu, default adalah
      “reading”.
      “components”: [
      {
          “type”: “editbox”,
          “id”: “edt1”,
          “left”: 0,
          “top”: 0,
          “width”: 595.28,
          “height”: 841.89,
          “text_size”: 22,
          “default_text”: “123”,
          “text”: “”,
          …
      },
      {
          “type”: “editbox”,
          “id”: “edt2”,
          “left”: 0,
          “top”: 0,
          “width”: 595.28,
          “height”: 841.89,
          “text_size”: 22,
          “default_text”: “123”,
          “text”: “”,
      }
      ]
     },
  ]
}
```