```java
package com.evgenii.aescrypto;

import com.evgenii.aescrypto.interfaces.JsEncryptorInterface;
import com.evgenii.aescrypto.interfaces.MainActivityInterface;
import com.evgenii.jsevaluator.interfaces.JsCallback;

public class Decrypt {
   private final MainActivityInterface mActivity;
   private final JsEncryptorInterface mJsEncryptor;

   public Decrypt(MainActivityInterface activity, JsEncryptorInterface jsEncryptor) {
      mActivity = activity;
      mJsEncryptor = jsEncryptor;
   }

   public void decryptAndUpdate() {
      if (!isDecryptable())
         return;

      mActivity.updateBusy(true);
      mJsEncryptor.decrypt(mActivity.trimmedMessage(), mActivity.trimmedPassword(),
            new JsCallback() {
               @Override
               public void onResult(final String decryptedTextFromJs) {
                  mActivity.updateBusy(false);

                  if (decryptedTextFromJs != null &&
!decryptedTextFromJs.trim().isEmpty()) {
                     mActivity.setMessage(decryptedTextFromJs);
                  }
               }

               @Override
               public void onError(String errorMessage) {
                  // Process JavaScript error here.
                  // This method is called in the UI thread.
               }
            });
   }

   public boolean isDecryptable() {
      if (mActivity.isBusy())
         return false;

      if (!mActivity.hasPassword())
         return false;

      if (!mJsEncryptor.isEncrypted(mActivity.trimmedMessage()))
         return false;

      return true;
   }
}
package com.evgenii.aescrypto;

import com.evgenii.aescrypto.interfaces.ClipboardInterface;
import com.evgenii.aescrypto.interfaces.JsEncryptorInterface;
import com.evgenii.aescrypto.interfaces.MainActivityInterface;
import com.evgenii.jsevaluator.interfaces.JsCallback;

public class Encrypt {
   private final MainActivityInterface mActivity;
   private final JsEncryptorInterface mJsEncryptor;
   private final ClipboardInterface mClipboard;
   private boolean mJustCopied;

   public Encrypt(MainActivityInterface activity, JsEncryptorInterface jsEncryptor,
         ClipboardInterface clipboard) {
      mActivity = activity;
      mJsEncryptor = jsEncryptor;
      mClipboard = clipboard;
   }

   public void encryptAndUpdate() {
      if (!isEncryptable())
         return;

      mActivity.updateBusy(true);
      mJsEncryptor.encrypt(mActivity.trimmedMessage(), mActivity.trimmedPassword(),
            new JsCallback() {
               @Override
               public void onResult(final String encryptedMessage) {
                  storeMessageInClipboard(encryptedMessage);
                  mActivity.setMessage(encryptedMessage);
                  mActivity.updateEncryptButtonTitle();
                  mActivity.updateBusy(false);
               }

               @Override
               public void onError(String errorMessage) {
                  // Process JavaScript error here.
                  // This method is called in the UI thread.
               }
            });
   }

   public boolean getJustCopied() {
      return mJustCopied;
   }

   public boolean isEncryptable() {
      return mActivity.hasMessage() && mActivity.hasPassword() && !mActivity.isBusy();
   }

   private void storeMessageInClipboard(String message) {
      mClipboard.set(message);
      updateJustCopied(true);
   }

   public void updateJustCopied(boolean justCopied) {
      mJustCopied = justCopied;
   }

}
package com.evgenii.aescrypto;

import android.annotation.SuppressLint;
import android.app.ActionBar;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.ShareCompat;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;

import com.evgenii.aescrypto.interfaces.MainActivityInterface;

public class MainActivity extends Activity implements MainActivityInterface {
    private JsEncryptor mJsEncryptor;
    private EditText mMessage;
    private EditText mPassword;
    private boolean mIsBusy;
    private Clipboard mClipboard;
    private Encrypt mEncrypt;
    private Decrypt mDecrypt;

    private String getEncryptButtonTitle() {
        if (mEncrypt.getJustCopied())
            return getResources().getString(R.string.menu_encrypt_title_copied);
        else
            return getResources().getString(R.string.menu_encrypt_title);
    }

    public JsEncryptor getEncryptor() {
        return mJsEncryptor;
    }

    @Override
    public boolean hasMessage() {
        return trimmedMessage().length() > 0;
    }

    @Override
    public boolean hasPassword() {
        return trimmedPassword().length() > 0;
    }

    @Override
    public boolean isBusy() {
        return mIsBusy;
    }

    public void onClearTapped(View view) {
        setMessage("");
    }

    public void onCopyTapped(View view) {
        if (!hasMessage())
            return;

        mClipboard.set(trimmedMessage());
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        mJsEncryptor = JsEncryptor.evaluateAllScripts(this);

        mMessage = (EditText) findViewById(R.id.message);
        mPassword = (EditText) findViewById(R.id.password);
        mClipboard = new Clipboard(this);
```

```java
        mDecrypt = new Decrypt(this, mJsEncryptor);
        mEncrypt = new Encrypt(this, mJsEncryptor, mClipboard);

        setupInputChange();
        setupActionBar();
        handleIncomingContent();
    }

    // Receive shared text from other apps
    private void handleIncomingContent() {
        Intent intent = getIntent();
        String action = intent.getAction();
        String type = intent.getType();

        if (type == null) return;
        if (!Intent.ACTION_SEND.equals(action)) return;
        if (!"text/plain".equals(type)) return;
        String sharedText = intent.getStringExtra(Intent.EXTRA_TEXT);
        if (sharedText == null) return;
        setMessage(sharedText);
    }

    public void onDecryptTapped(View view) {
        mDecrypt.decryptAndUpdate();
    }

    public void onEncryptTapped(View view) {
        mEncrypt.encryptAndUpdate();
    }

    public void onShareTapped(View view) {
        if (!hasMessage()) return;
        Share.shareMessage(this, trimmedMessage());
    }

    private void updateShareButtonVisibility() {
        ImageButton button = (ImageButton) findViewById(R.id.shareImageButton);
        button.setAlpha((float)(hasMessage() ? 1.0 : 0.3));
    }

    private void onPasswordOrMessageChanged() {
        if (isBusy())
            return;

        mEncrypt.updateJustCopied(false);
        updateEncryptButtonTitle();
        updateShareButtonVisibility();
    }

    public void onPasteTapped(View view) {
        final String messageFromClipboard = mClipboard.get();
        if (messageFromClipboard == null || messageFromClipboard.trim().isEmpty())
            return;

        setMessage(messageFromClipboard);
    }

    public void onShowHelpClicked(View view) {
        final Intent intent = new Intent(this, HelpActivity.class);
        startActivity(intent);
    }

    @Override
    public void setMessage(String message) {
        mMessage.setText(message);
    }

    @SuppressLint("InflateParams")
    protected void setupActionBar() {
        final ActionBar actionBar = getActionBar();
        actionBar.setDisplayShowHomeEnabled(false);
        actionBar.setDisplayUseLogoEnabled(false);
        actionBar.setDisplayShowTitleEnabled(false);
        actionBar.setDisplayShowCustomEnabled(true);
        final View actionBarView =
getLayoutInflater().inflate(R.layout.main_action_bar, null);
        actionBar.setCustomView(actionBarView);
        actionBar.setDisplayOptions(ActionBar.DISPLAY_SHOW_CUSTOM);
        updateShareButtonVisibility();
    }

    private void setupInputChange() {
        mMessage.addTextChangedListener(new TextWatcher() {

            @Override
            public void afterTextChanged(Editable s) {
                onPasswordOrMessageChanged();
            }

            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int
after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int
count) {
            }
        });

        mPassword.addTextChangedListener(new TextWatcher() {

            @Override
            public void afterTextChanged(Editable s) {
                onPasswordOrMessageChanged();
            }

            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int
after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int
count) {
            }
        });
    }

    @Override
    public String trimmedMessage() {
        return mMessage.getText().toString().trim();
    }

    @Override
    public String trimmedPassword() {
        return mPassword.getText().toString().trim();
    }

    @Override
    public void updateBusy(boolean isBusy) {
        mIsBusy = isBusy;
    }

    @Override
    public void updateEncryptButtonTitle() {
        final Button encryptButton = (Button) findViewById(R.id.encryptButton);
        encryptButton.setText(getEncryptButtonTitle());
    }

}
```