Advances in Social Science, Education and Humanities Research, volume 613
4th International Conference on Innovative Research Across Disciplines (ICIRAD 2021)

Development of Mobile Based Application for Balinese Lontar Image Transliteration

Made Windu Antara Kesiman[1,*] Kadek Teguh Dermawan[1] Gede Aditra Pradnyana[1]

[1] *Educational of Informatic Program Study, Universitas Pendidikan Ganesha, Singaraja, Indonesia*
[*]*Corresponding author. Email: antara.kesiman@undiksha.ac.id*

ABSTRACT
AKSALont is an application that can automatically transliterate the digital image of Lontar Bali into text data format in the Latin alphabet. Seeing the broader needs of the AKSALont application, it was deemed necessary to develop a mobile application version of Akasalont that remains integrated with the main web-based application. This paper presents the detail development process of mobile based application of AKSALont. Three main development phases are presented. Software design phase consists of software architecture design and software interface design. The software implementation modules for AKSALont mobile application consists of three sub-modules namely image cropping module, machine interconnection module, and mobile application interface module. Several black box testing scenarios were performed to test the software. Results show that the mobile-based interface of AKSALont is successfully integrated into interconnection with the main web server and machine of AKSALont.

Keywords: *Mobile based application, Balinese Lontar, Image, Transliteration.*

1. INTRODUCTION

Recently, the attention of the Indonesian people to the existence of ancient manuscript collections has been increasing. Several groups of researchers and practitioners of ancient manuscripts began to propose forums for activities to increase knowledge and awareness of the wider community about ancient manuscripts [1]-[3]. In addition, researchers in the field of digital technology are also trying to propose systems that can assist efforts to save and preserve ancient manuscripts. One application that has been proposed is AKSALont [4].

AKSALont is an application that can automatically transliterate the digital image of Lontar Bali into text data format in the Latin alphabet [5]. AKSALont was proposed and developed by a research group from Virtual, Vision, Image, and Pattern (VVIP) [6], and is under the scheme of Project EpsiLont (Electronic Pattern Analysis for Lontar). Initially, AKSALont was only developed with a web-based main interface. The main machine of AKSALont is built by a Python module that implements the LSTM deep learning model. This LSTM model was previously trained to directly transliterate images of text from Lontar Bali into Latin text [4][7].

Seeing the wider needs of the Akasalont application, it was deemed necessary to develop a mobile application version of Akasalont that remains integrated with the main web-based application. With a mobile-based application, it is hoped that the main goal of the EpsiLont Project which is the development of the Bali Lontar transliteration application, can be more touching and effectively target the community more quickly and widely. Web-based applications also, of course offer major advantages in terms of portability and ease of access.

This paper presents the detail development process of mobile based application of AKSALont. Section 2 will describe the software design phase. Software implementation modules will be presented in Section 3. Section 4 will show the software testing scenarios and results. Finally, conclusions will be presented in Section 5.

2. SOFTWARE DESIGN

AKSALont mobile application is developed under the Android platform. AKSALont mobile application is built based on an interconnection ecosystem with web services.

2.1. Software Architecture Design

The principle of cross-platform interoperability is used for machine integration in the Akasalont mobile application platform. AKSALont machine runs in a python working environment, and AKSALont mobile application is developed with an Android-based mobile application architecture.

In the first stage, the AKSALont machine needs to be configured in such a way that the transliteration result of the AKSALont machine is sent through an API. The API will be executed by the client-side then the API response will be given to the mobile application client. The Flask framework was used in the process of producing and sending the AKSALont API. The task of the Flask framework is creating API access via the HTTP/HTTPS protocol.

In the second stage, the mobile application performs the cropping process on the image. The cropped image will be captured and encoded using base64 image encoding. This aims to prevent the tampering condition of the image for the AKSALont machine. Then the request is sent to the AKSALont machine. The series of image cropping processes are carried out using the android API. This process requires access permission from the user device to access the External Storage with READ and WRITE mode access permission. Figure 1 shows the implementation of the access rights configuration code carried out on the AKSALont mobile application.

The mobile application interface captures the interaction of cropping an image into a file, then encoding it using a base64 image. The encoded image file snippet becomes a request parameter to the AKSALont transliteration API. The POST method is used as the request method to the API. The transliteration result of the AKSALont is read through the JSON response generated from the Flask framework. Figure 2 shows a code snippet of the encoded image data request implementation to the AKSALont machine.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.aksalont_mobile">

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

    <application
      android:name=".MainApplication"
      android:label="@string/app_name"
      android:icon="@mipmap/ic_launcher"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:allowBackup="false"
      android:theme="@style/AppTheme">
      <activity
        android:name=".MainActivity"
        android:label="@string/app_name"
        android:configChanges="keyboard|keyboardHidden|orientation|screenSize|uiMode"
        android:launchMode="singleTask"
        android:windowSoftInputMode="adjustResize">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>
    </application>
</manifest>
```

Figure 1 Access rights configuration code.

```
const proceed = () => {
  setIsLoading(true);
  let url = 'https://aksalont.mudratech.org:5000/engine';
  let formData = new FormData();
  formData.append('data', imgdata.preview);
  formData.append('textFile', 'aksalont');
  fetch(url, {
    method: 'POST',
    body: formData,
  })
    .then((resp) => resp.json())
    .then((payload) => {
      setIsLoading(false);
      setResult(payload);
    })
    .catch((err) => {
      setIsLoading(false);
      console.log(err);
    });
};
```

Figure 2 Encoded image data request.

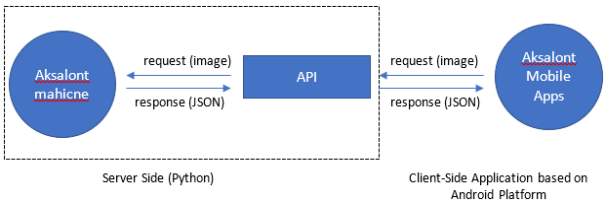


Figure 3 Interconnection architecture of the AKSALont.

In the third stage, the captured JSON response is parsed into an object that can be captured by the AKSALont mobile application activity fragment. This object is embedded in the AKSALont mobile interface component. The object that is captured from the JSON parsing response is in the form of data and probability fields. The data field contains the result of Latin transliteration from the AKSALont machine, while the probability field contains the probability value of transliteration accuracy. Figure 3 shows the interconnection architecture of the AKSALont machine and the AKSALont mobile application.

2.2. Software Interface Design

After the software architecture design step, the software interface will be designed for the AKSALont mobile application. In this step, a wireframe design is made for the AKSALont mobile application (Figure 4). The following wireframe specifications will then be implemented in the form of an Android-based application user interface component.

The first component is the Image Capture Interface Component with camera access. This component consists of two main interaction components, namely an image box that contains instructions for using the application and a button to trigger image capture with the camera. When the button is pressed it will trigger a trigger to activate the shooting API for the camera mode of the android device. This process will then continue at the image cropping stage with the Android Image Cropper API.

The second component is the Android Image Cropper API. The captured image from the camera is then processed in such a way by utilizing the Android Image Cropper API. This android API allows to access the image cropping feature on the Android platform. The wireframe screen for this component is to display a cropping window with full screen mode, so that it provides a wide area for the user to choose the desired image cropping area. Users will fully interact with the image crop feature on each android device. During this process, it is ensured that user has a large enough screen area to operate the image cropping feature. After the cropping process, user will then be taken to a screen fragment to show a preview of the cropped image.

The third component is Cropped Image Preview Interface Component. The AKSALont mobile application is able to display a preview of the cropped image performed by the user using the Android Image Cropper API. This is provided so that user can verify the results of the image cropping process that will be sent to and processed by the AKSALont machine. In this wireframe screen design, there will be three main interface components, namely 2 buttons and 1 image container. The image container provides a preview of the image cropping results. The first button is provided for user to recapture images with the camera, while the second button (Proceed button) is used to continue and activate the main process using the AKSALont machine. This wireframe design provides an opportunity for user to correct the image taken using his device's camera. When the Proceed button is touched, then the cropped image is then sent to the server to be further processed by the AKSALont machine.

The fourth component is the Transliteration Result Interface Component. The results of image transliteration processing by the AKSALont machine are then displayed on the same screen as the preview image, but there will be additional components in the form of two labels. The first label is used to show the transliteration result of the Lontar cropped image in Latin alphabets from the AKSALont machine, while the second label displays the information of the probability value of the transliteration accuracy performed by the Aksalon machine. The transliteration label is displayed just below the image preview box of the Lontar cropped image. The label is placed in the center position. The second label is placed below it.

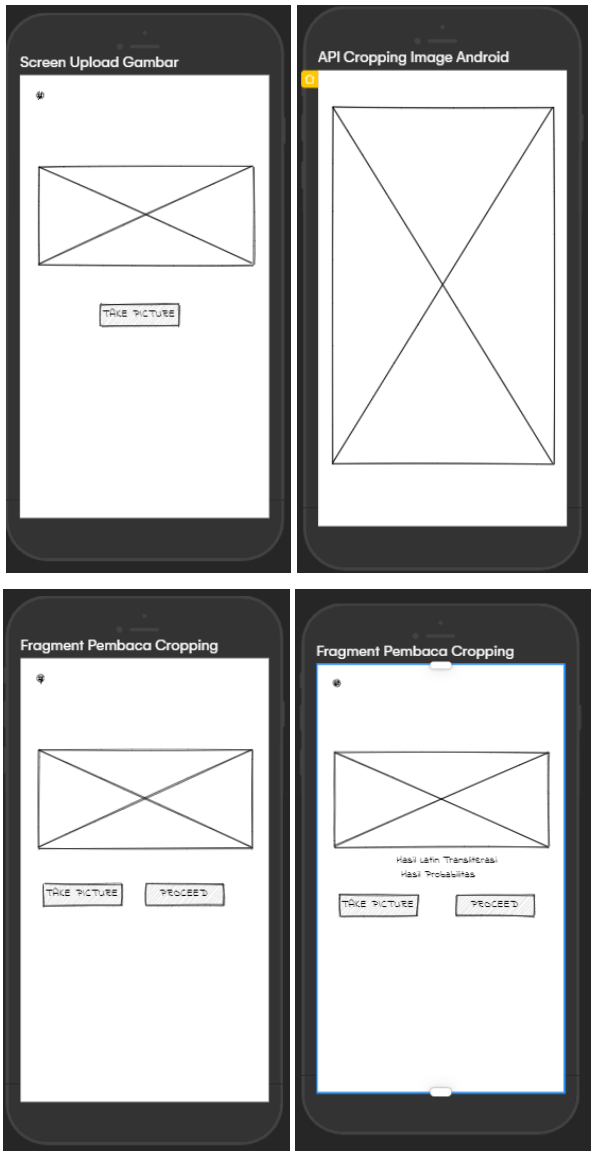


Figure 4 Wireframe design for the AKSALont mobile application.

3. SOFTWARE IMPLEMENTATION

Software implementation modules for AKSALont mobile application consist of three sub-modules.

3.1. Android Native Cropping API

The Android Native Cropping API was developed based on the image processing standardization performed by the Android platform [8]. This stage requires the ability to connect client-side scripting to the native image editor interface of the Android platform. Figure 5 shows a code snippet of the Android Native Cropping API development.

```
const openCamera = () => {
  var RNFetchBlob = require('react-native-fetch-blob').default;

  ImagePicker.openCamera({
    width: 200,
    height: 200,
    cropping: true,
    freeStyleCropEnabled: true,
    includeBase64: true,
    cropperToolbarTitle: 'Aksalont Croppper',
  }).then(async (image) => {
    let imgPreview = 'data:' + 'jpg' + ';base64,' + image.data;
    var fileUpload = await RNFetchBlob.fs.stat(image.path);
```

Figure 5 Android Native Cropping API.

```
  let imgPreview = 'data:' + 'jpg' + ';base64,' + image.data;
  var fileUpload = await RNFetchBlob.fs.stat(image.path);
  let fileToUpload = {
    uri: 'file://' + fileUpload.path,
    type: 'jpg',
    name: fileUpload.filename,
  };
  console.log('fileTopUpload', fileToUpload);
  console.log('home');
  let imgCropped = {
    uri: fileToUpload.uri,
    width: image.width,
    height: image.height,
    preview: imgPreview,
  };
  // console.log('imgCropped', imgCropped);
  console.log('imgPreview', imgPreview);
  setImgdata(imgCropped);
});
```

Figure 6 Data parameter preparation process.

The code is designed in such a way that when an image is taken, the image automatically enters the native image editor of the Android platform. Furthermore, on this screen, the user can use the cropping feature. When the cropping results have been confirmed, then the parameter data compilation process is carried out to make requests to the AKSALont machine API.

Figure 6 shows a code snippet that contains the data parameter preparation process carried out. The image will be processed and then it will proceeded with the encoding process. The results of the encoding will be placed in a data structure object that contains a file specification, image preview, width and height of the image. Image preview is the result of base64 image encoding processing. Base64 image encoding allows data parameters to be transferred without any data interruption. This is to ensure that AKSALont machine gets a complete image file for main processing.

3.2. AKSALont mobile application and AKSALont machine interconnection

The interconnection of AKSALont mobile application is supported by web services. Web services are created by the AKSALont machine, by receiving a series of parameters to be continued as transliteration processing input. Web services make requests to the AKSALont machine by using the POST method. This API is under the address of "/engine" and it requires two parameters, namely data and textFile parameters.

```
const proceed = () => {
  setIsLoading(true);
  let url = 'https://aksalont.mudratech.org:5000/engine';
  let formData = new FormData();
  formData.append('data', imgdata.preview);
  formData.append('textFile', 'aksalont');
  fetch(url, {
    method: 'POST',
    body: formData,
  })
    .then((resp) => resp.json())
    .then((payload) => {
      setIsLoading(false);
      setResult(payload);
    })
    .catch((err) => {
      setIsLoading(false);
      console.log(err);
    });
};
```

Figure 7 POST interconnection process to the AKSALont machine API.

```
{result.data && (
  <View>
    <Text style={styles.btnTxt}>{result.data}</Text>
    <Text style={styles.btnTxt}>Probabilitas : {result.probability}</Text>
  </View>
)}
```

Figure 8 JSON response parsing.

Figure 7 shows a snippet of the source code for the POST interconnection process to the AKSALont machine API. The data parameter contains the contents of the base64 image encoding, and textFile is an additional parameter to be used as a reference parameter that the request is made by AKSALont mobile application. Previously, AKSALont had been developed on a web platform. The textFile parameter can later be used as an aid to determining the source of the data request to the AKSALont machine. The AKSALont machine will then provide data response in JSON format [9]. The AKSALont mobile application parses the JSON data responses. The results of this parsing will then be rendered into the AKSALont display.

JSON response parsing results produce two response fields, namely the "data" and "probability" fields (Figure 8). Field data contains the transliteration text results in Latin alphabets, and Field probability records the probability of the transliteration results accuracy of the AKSALont machine.

3.3. AKSALont mobile application interface

The user interface implementation is based on the wireframe design. The user interface has a Material design concept. This concept provides a relatively comfortable colour and is filled with shades of shadow in some of its implementations. For example, the box shadow on the button component. Figure 9 shows the result of interface implementation of AKSALont mobile application based on the previous description of wireframe design.

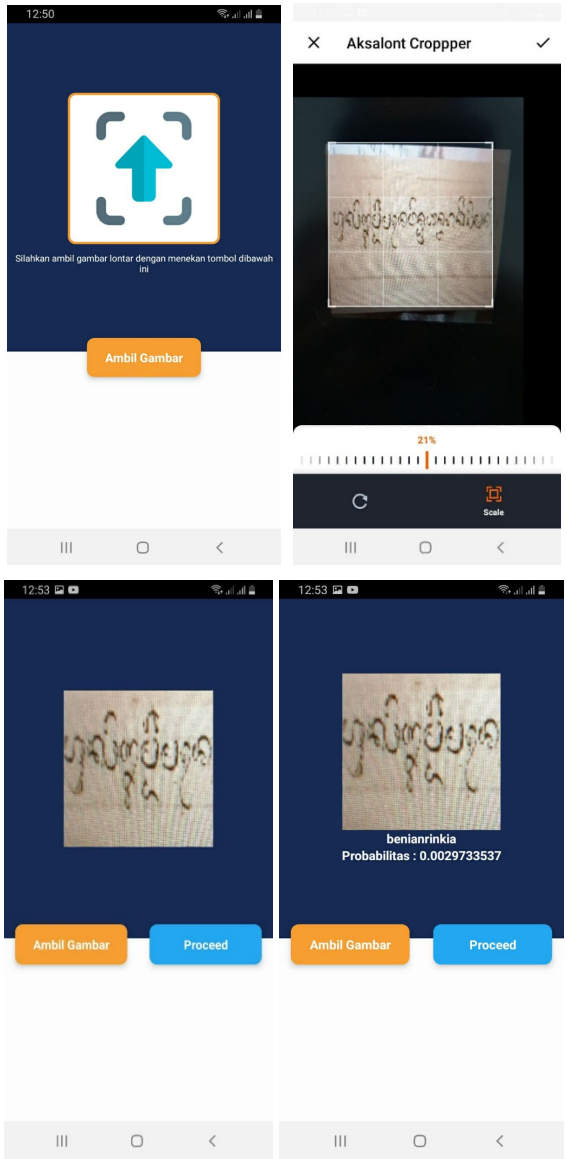
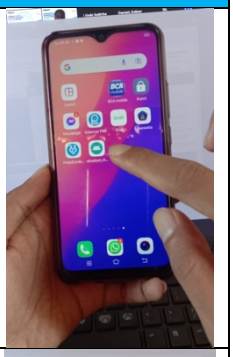
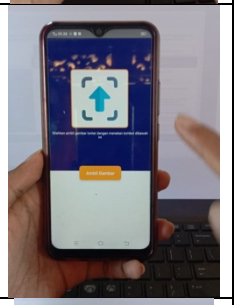
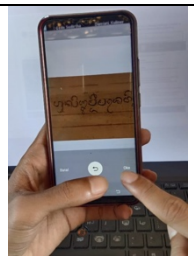
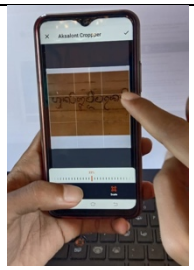


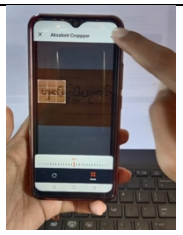
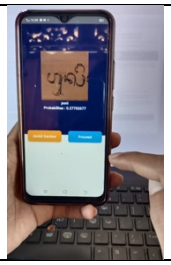
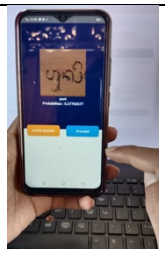
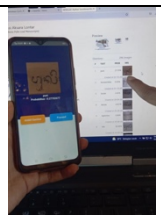
Figure 9 Result of interface implementation of AKSALont mobile application.

4. SOFTWARE TESTING

The black box testing method was performed in several scenarios. Table 1 shows the detail testing scenarios and results for AKSALont mobile application.

Table 1 Testing scenarios and results

No	Scenario	Target	Result
1	Installing the application on a mobile phone	The application is installed properly on the mobile phone, the application icon is displayed on the mobile phone screen	
2	Opening the application on the mobile phone	By clicking on the application icon, the application interface screen opens on the mobile phone screen	
3	Access or turn on the camera feature on the mobile phone by the application	By clicking the Take Picture button, the application will turn on the camera feature of the mobile phone, and can see the Lontar object that will be captured.	
4	Taking Lontar images with the camera via the application interface	By clicking the OK button, the part of the Lontar object that appears on the camera screen will be captured	
5	Displays the grid feature for the cropping process on the Lontar image that has been taken by the application	After the capturing process by the camera is carried out, the application will display a grid feature for the cropping process on the Lontar image.	

6	Set the position and size of the cropping grid on the Lontar image that has been taken by the application	Users can adjust the position and size of the cropping grid on the Lontar image, by pressing the corners and edges of the grid and then shifting them, both enlarging and reducing the grid area according to the user's wishes.	
7	Confirming the cropping process according to the position and size of the cropping grid on the Lontar image that has been taken by the application	The user clicks the V icon to confirm the grid area to be cropped from the Lontar image	
8	Displaying the cropped Lontar image on the application screen	The cropped Lontar image, according to the user's choice of grid area, is displayed on the application screen	
9	Perform the main process of transliterating the cropped Lontar image on the application screen	Users can press the Proceed button to transliterate the Lontar image seen on the application screen	
10	Displaying the transliteration of the Lontar image on the application screen	Transliteration results are displayed on the application screen	
11	Displays the predicted probability value of the transliteration of Lontar image	The result of the probability value of the prediction of the transliteration result is displayed on the application screen	
12	Sending and displaying Lontar images that have been processed on a mobile phone, along with the transliteration results and the predicted probability values of the transliterated results, integrated into the main screen of the web application	After the transliteration process on the mobile phone is successfully carried out, result will be integrated and stored on the web application screen	

5. CONCLUSIONS

Three main development phases for AKSALont mobile-based applications are presented. The first phase of software design consists of software architecture design and software interface design. The second phase of software implementation modules for AKSALont mobile application consists of three sub modules, namely image cropping module, machine interconnection module, and mobile application interface module. The final phase for software testing was conducted by performing several black box testing scenarios to test the AKSALont mobile based application. Results show that the mobile based interface of AKSALont is successfully integrated in an interconnection with the main web server and machine of AKSALont.

AUTHORS' CONTRIBUTIONS

Made Windu Antara Kesiman is the main researcher for the complete scheme of the development of AKSALont. Kadek Teguh Dermawan is the main programmer for mobile-based applications and Gede Aditra Pradnyana performed the testing phase for the application.

ACKNOWLEDGMENTS

This research is supported by Penelitian Dasar Unggulan Perguruan Tinggi (PDUPT) 2021 Ristekbrin Research Funding Program.

REFERENCES

[1] M. Suryani, E. Paulus, S. Hadi, U. A. Darsa, and J. Burie, "The Handwritten Sundanese Palm Leaf Manuscript Dataset from 15th Century," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017, pp. 796–800, doi: 10.1109/ICDAR.2017.135.

[2] M. W. A. Kesiman, J.-C. Burie, J.-M. Ogier, G. N. M. A. Wibawantara, and I. M. G. Sunarya, "AMADI_LontarSet: The First Handwritten Balinese Palm Leaf Manuscripts Dataset," in *15th International Conference on Frontiers in Handwriting Recognition 2016*, 2016, pp. 168–172, doi: 10.1109/ICFHR.2016.39.

[3] D. Valy, M. Verleysen, S. Chhun, and J.-C. Burie, "A New Khmer Palm Leaf Manuscript Dataset for Document Analysis and Recognition – SleukRith Set," 2017.

[4] M. W. A. Kesiman and K. T. Dermawan, "AKSALont: Aplikasi transliterasi aksara Lontar Bali dengan model LSTM," *J. Teknol. dan Sist. Komput.*, vol. 9, no. 3, pp. 142–149, 2021, [Online]. Available: https://doi.org/10.14710/jtsiskom.2021.13969.

[5] "No Title." https://aksalont.mudratech.org/.

[6] "No Title." https://research.undiksha.ac.id/vvip-rg/.

[7] M. W. A. Kesiman and I. M. D. Maysanjaya, "A model for post transliteration suggestion for balinese palm leaf manuscript with text generation and lstm model," *J. Phys. Conf. Ser.*, vol. 1810, no. 1, 2021, doi: 10.1088/1742-6596/1810/1/012011.

[8] "Android Developers." https://developer.android.com/ (accessed Oct. 20, 2021).

[9] "Introducing JSON." https://www.json.org/ (accessed Oct. 20, 2021).