Gmail

**Indra Laksmana <indra.puskom@gmail.com>**

# Persyaratan Pendaftaran Hak Cipta Software

1 pesan

**Elita Amrina** <elitaamrina@gmail.com> 24 September 2018 pukul 16.55
Kepada: indra.puskom@gmail.com

Assalamu'alaikum Pak Indra,

Terlampir Formulir Pendaftaran untuk HKI (Hak Cipta), ada 3 formulir, dibuat 1 lembar untuk masing-masing pendaftaran. Formulir 2 dan 3 pakai materai.

Selain formulir tersebut, persyaratan lainnya:
1. Bukti Hak Cipta yg didaftarkan: Buku Panduan Software (1 eksemplar) dan softcopy panduan (pdf/jpg),
2. Scan KTP dan NPWP Pencipta (semua pencipta) (pdf)

Hardcopy Formulir 1, 2, 3 dan buku panduan software dikumpulkan ke LPPM.
Softcopy buku panduan, scan KTP dan NPWP dapat dikirim ke email: pdti.unand@gmail.com.

Terima kasih.

Elita Amrina
PDTI LPPM UNAND

**5 lampiran**

- **Formulir Permohonan Hak Cipta ONLINE -LPPM UNAND 2017.doc**
  55K
- **SURYA AFNARIUS SURAT PERNYATAAN GIS MASJID.pdf**
  968K
- **SURYA AFNARIUS SURAT PENGALIHAN GIS MASJID.pdf**
  897K
- **SURYA AFNARIUS PROGRAM APLIKASI WEB GIS MASJID.pdf**
  1623K

**Herwandi, Permohonan Hak Cipta 'Desain Motif Batik Kabek Daun Kacang', 2017.doc**
92K

Gmail

**Indra Laksmana <indra.puskom@gmail.com>**

# Fwd: Billing Code Pembayaran (No-Reply)

3 pesan

**Elita Amrina** <elitaamrina@gmail.com> 19 November 2018 pukul 11.28
Kepada: Indra Laksmana <indra.puskom@gmail.com>

---------- Forwarded message ---------
From: **INFO HAKCIPTA** <hakcipta-noreply@dgip.go.id>
Date: Mon, Nov 19, 2018 at 11:20 AM
Subject: Billing Code Pembayaran (No-Reply)
To: <lppm.unand@gmail.com>

**KEMENTRIAN HUKUM DAN HAK ASASI MANUSIA**

**REPUBLIK INDONESIA**

**DIREKTORAT JENDERAL KEKAYAAN INTELEKTUAL**

Jl. H.R. Rasuna Said Kav 8-9 Jakarta Selatan 12940

Telepon: (021) 57905609 Faksimili: (021) 57905609

---

Yth,
Elita Amrina,

Silahkan lakukan pembayaran dengan billing code berikut untuk permohonan dengan nomor 201831049:

**820181119791525**

Jika ada pertanyaan lebih lanjut, silahkan menghubungi:
siki@dgip.go.id
Terima kasih telah mengajukan permohonan pembuatan pengguna di Aplikasi Permohonan Pendaftaran Ciptaan Secara Elektronik.

Direktorat Jenderal Kekayaan Intelektual

**Elita Amrina** <elitaamrina@gmail.com> 21 November 2018 pukul 07.53
Kepada: Indra Laksmana <indra.puskom@gmail.com>

**KEMENTRIAN HUKUM DAN HAK ASASI MANUSIA**

**REPUBLIK INDONESIA**

**DIREKTORAT JENDERAL KEKAYAAN INTELEKTUAL**

Jl. H.R. Rasuna Said Kav 8-9 Jakarta Selatan 12940

Telepon: (021) 57905609 Faksimili: (021) 57905609

Yth,
Elita Amrina,

Silahkan lakukan pembayaran dengan billing code berikut untuk permohonan dengan nomor 201831195:

**820181121865710**

[Kutipan teks disembunyikan]

**Indra Laksmana** <indra.puskom@gmail.com> 21 November 2018 pukul 08.13
Kepada: Tri Novita Zuhara Jingga <t.zuhara@gmail.com>

[Kutipan teks disembunyikan]

**Lampiran I**
**Peraturan Menteri Kehakiman R.I.**
**Nomor : M.01-HC.03.01 Tahun 1987**

Kepada Yth. :
Direktur Jenderal HKI
melalui Direktur Hak Cipta,
Desain Industri, Desain Tata Letak,
Sirkuit Terpadu dan Rahasia Dagang
di
Jakarta

# PERMOHONAN PENDAFTARAN CIPTAAN

I. Pencipta :

1. Nama : Indra Laksmana, S.Kom, M.Kom
   Kewarganegaraan : Indonesia
   Alamat : Jl.Raya Bukittinggi KM.4 No.12 Obay, Jorong Padang Lua 1 Kec. Banuhampu, Kab Agam. Sumatera Barat
   Telepon : -
   No. HP & E-mail : 085313767633 / indra.puskom@gmail.com

2. Nama : Rosda Syelly, S.Kom, M.Kom
   Kewarganegaraan : Indonesia
   Alamat : Sekolah Tinggi Teknologi Payakumbuh. Jl. Khatib Sulaiman Sawah Padang Payakumbuh 26227 Sumatera Barat.
   Telepon : 0752- 796063
   No. HP & E-mail : 085374107182 / rosdasyelly@gmail.com

3. Nama : Ir. Nurzarrah Tazar, MP
   Kewarganegaraan : Indonesia
   Alamat : Politeknik Pertanian Negeri Payakumbuh Jl. Raya Negara KM7 Tanjung Pati Kec. Harau Kab. Limapuluh Kota 26271
   Telepon : (0752) 7754192
   No. HP & E-mail : 081374069610 / nurzarrah.tazar@politanipyk.ac.id

4. Nama : Perdana Putera, ST, M.Eng
   Kewarganegaraan : Indonesia
   Alamat : Politeknik Pertanian Negeri Payakumbuh Jl. Raya Negara KM7 Tanjung Pati Kec. Harau Kab. Limapuluh Kota 26271
   Telepon : (0752) 7754192
   No. HP & E-mail : 082382238163 / Perdanaputera81@gmail.com

5. Nama : Amrizal, S.Kom, M.Kom
   Kewarganegaraan : Indonesia
   Alamat : Politeknik Pertanian Negeri Payakumbuh Jl. Raya Negara KM7 Tanjung Pati Kec. Harau Kab. Limapuluh Kota 26271
   Telepon : (0752) 7754192
   No. HP & E-mail : 08126797265 / amrizal.ch@gmail.com

II. Pemegang Hak Cipta :

| | | |
|---|---|---|
| 1. Nama | : | P3M Politeknik Pertanian Negeri Payakumbuh |
| 2. Kewarganegaraan | : | Indonesia |
| 3. Alamat | : | Politeknik Pertanian Negeri Payakumbuh Jl. Raya Negara KM7 Tanjung Pati Kec. Harau Kab. Limapuluh Kota 26271.SUMBAR |
| 4. Telepon | : | 0752-7754192 |
| 5. No. HP & E-mail | : | 081339163925 / lembagapenelitiandanpengabdian@gmail.com |

III. Kuasa :

| | | |
|---|---|---|
| 1. Nama | : | - |
| 2. Kewarganegaraan | : | - |
| 3. Alamat | : | - |
| 4. Telepon | : | - |
| 5. No. HP & E-mail | : | - |

IV. Jenis dan judul ciptaan yang dimohonkan : **Program Komputer / Sistem Identifikasi Kandungan Asam Cyanide Ubi Kayu dan Famili Tumbuhan Obat**

V. Tanggal dan tempat di-umumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indo-nesia : 9 Oktober 2018, di Manila Filipina

VI Uraian ciptaan : Porgram aplikasi ini untuk mengidentifikasi kandungan Asam Cyanide Ubi Kayu dan Famili Tumbuhan Obat berdasarkan morfologinya, sehingga dapat membantu dalam menentukan peruntukannya.

Tanjung Pati 13, November 2018
Kepala P3M

Arlizar, S.P, MP, P.hD

**SURAT PERNYATAAN**

Yang bertanda tangan dibawah ini :

| | | |
|---|---|---|
| N a m a | : | Aflizar, S.P, MP, P.hD |
| Kewarganegaraan | : | Indonesia |
| Alamat | : | Politeknik Pertanian Negeri Payakumbuh Jl. Raya Negara KM7 Tanjung Pati, Kec. Harau Kab. Limapuluh Kota Sumatera Barat 26271 |

Dengan ini menyatakan bahwa :

1. Karya Cipta yang saya mohonkan :

   Berupa : Program Komputer

   Berjudul : Sistem Identifikasi Kandungan Asam Cyanide Ubi Kayu dan Famili Tumbuhan Obat

   - Tidak meniru dan tidak sama secara esensial dengan Karya Cipta milik pihak lain atau obyek kekayaan intelektual lainnya sebagaimana dimaksud dalam Pasal 68 ayat (2);
   - Bukan merupakan Ekspresi Budaya Tradisional sebagaimana dimaksud dalam Pasal 38;
   - Bukan merupakan Ciptaan yang tidak diketahui penciptanya sebagaimana dimaksud dalam Pasal 39;
   - Bukan merupakan hasil karya yang tidak dilindungi Hak Cipta sebagaimana dimaksud dalam Pasal 41 dan 42;
   - Bukan merupakan Ciptaan seni lukis yang berupa logo atau tanda pembeda yang digunakan sebagai merek dalam perdagangan barang/jasa atau digunakan sebagai lambang organisasi, badan usaha, atau badan hukum sebagaimana dimaksud dalam Pasal 65 dan;
   - Bukan merupakan Ciptaan yang melanggar norma agama, norma susila, ketertiban umum, pertahanan dan keamanan negara atau melanggar peraturan perundang-undangan sebagaimana dimaksud dalam Pasal 74 ayat (1) huruf d Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

2. Sebagai pemohon mempunyai kewajiban untuk menyimpan asli contoh ciptaan yang dimohonkan dan harus memberikan apabila dibutuhkan untuk kepentingan penyelesaian sengketa perdata maupun pidana sesuai dengan ketentuan perundang-undangan.

3. Karya Cipta yang saya mohonkan pada Angka 1 tersebut di atas tidak pernah dan tidak sedang dalam sengketapidana dan/atau perdata di Pengadilan.

4. Dalam hal ketentuan sebagaimana dimaksud dalam Angka 1 dan Angka 2 tersebut di atas saya / kami langgar, maka saya / kami bersedia secara sukarela bahwa :
   a. permohonan karya cipta yang saya ajukan dianggap ditarik kembali; atau
   b. Karya Cipta yang telah terdaftar dalam Daftar Umum Ciptaan Direktorat Hak Cipta, Direktorat Jenderal Hak Kekayaan Intelektual, Kementerian Hukum Dan Hak Asasi Manusia R.I dihapuskan sesuai dengan ketentuan perundang-undangan yang berlaku.
   c. Dalam hal kepemilikan Hak Cipta yang dimohonkan secara elektronik sedang dalam berperkara dan/atau sedang dalam gugatan di Pengadilan maka status kepemilikan surat pencatatan elektronik tersebut ditangguhkan menunggu putusan Pengadilan yang berkekuatan hukum tetap

Demikian Surat pernyataan ini saya/kami buat dengan sebenarnya dan untuk dipergunakan sebagimana mestinya.

Tanjung Pati 13, November 2018


(Aflizar, S.P, MP, P.hD)

**SURAT PENGALIHAN HAK CIPTA**

Yang bertanda tangan dibawah ini :

1. N a m a : **Indra Laksmana, S.Kom, M.Kom**
   Alamat : Jl.Raya Bukittinggi KM.4 No.12 Obay, Jorong Padang Lua 1, Kec. Banuhampu, Kab Agam
2. N a m a : **Rosda Syelly, S.Kom, M.Kom**
   Alamat : Sekolah Tinggi Teknologi Payakumbuh. Jl. Khatib Sulaiman Sawah Padang Payakumbuh 26227 Sumatera Barat.
3. N a m a : **Ir. Nurzarrah Tazar, MP**
   Alamat : Politeknik Pertanian Negeri Payakumbuh Jl. Raya Negara KM7 Tanjung Pati Kec. Harau Kab. Limapuluh Kota 26271 Sumatera Barat
4. N a m a : **Perdana Putera, ST, M.Eng**
   Alamat : Politeknik Pertanian Negeri Payakumbuh Jl. Raya Negara KM7 Tanjung Pati Kec. Harau Kab. Limapuluh Kota 26271 Sumatera Barat
5. N a m a : **Amrizal, S.Kom, M.Kom**
   Alamat : Politeknik Pertanian Negeri Payakumbuh Jl. Raya Negara KM7 Tanjung Pati Kec. Harau Kab. Limapuluh Kota 26271 Sumatera Barat

Adalah **Pihak I** selaku pencipta, dengan ini menyerahkan karya ciptaan saya kepada :

N a m a : P3M Politeknik Pertanian Negeri Payakumbuh
Alamat : Politeknik Pertanian Negeri Payakumbuh Jl. Raya Negara KM7 Tanjung Pati Kec. Harau Kab. Limapuluh Kota 26271 Sumatera Barat

Adalah **Pihak II** selaku Pemegang Hak Cipta berupa **Program Komputer** dengan judul **Sistem Identifikasi Kandungan Asam Cyanide Ubi Kayu dan Famili Tumbuhan Obat** untuk didaftarkan di Direktorat Hak Cipta, Desain Industri, Desain Tata Letak dan Sirkuit Terpadu dan Rahasia Dagang, Direktorat Jenderal Hak Kekayaan Intelektual, Kementerian Hukum dan Hak Azasi Manusia R.I.

Demikianlah surat pengalihan hak ini kami buat, agar dapat dipergunakan sebagaimana mestinya.

Tanjung Pati 13, November 2018

Pemegang Hak Cipta

(Aflizar, S.P, MP, P.hD)

Pencipta

METERAI TEMPEL
2E8D0AFF243909157
6000 ENAM RIBU RUPIAH

(Indra Laksmana, S.Kom, M.Kom )

(Rosda Syelly, S.Kom, M.Kom)

(Ir. Nurzarrah Tazar, MP)

(Perdana Putera, ST, M.Eng)

(Amrizal, S.Kom, M.Kom)

# Petunjuk Penggunaan Aplikasi
# (User Manual)

SISTEM IDENTIFIKASI KANDUNGAN ASAM CYANIDE (HCN) UBI KAYU DAN FAMILI TUMBUHAN OBAT

TIM Penyusun:

Indra Laksmana, S.Kom, M.Kom
Rosda Syelly, S.Kom, M.Kom
Ir. Nurzarrah Tazar, MP
Perdana Putera, ST, M.Eng
Amrizal, S.Kom, M.Kom

# KATA PENGANTAR

Alhamdulillahi rabbil'alamin, puji dan syukur penulis panjatkan kehadirat Allah SWT, karena hanya dengan pertolongan dan rahmat-Nya sehingga kami dapat menyelelesaikan penelitian ini dan telah membuat buku panduan penggunaan aplikasi ini. Kami juga mengucapkan terima kasih kepada Kemenristek DIKTI yang telah mendanai penelitian ini dengan bantuan P3M Politeknik Pertanian Negeri Payakumbuh dan teman-teman peneliti yang telah membantu dalam proses penelitian ini serta semua pihak yang telah bekerjasama dengan kami.

Perkembangan teknologi yang terus meningkat telah memberikan banyak kemudahan dalam berbagai aspek. Kemudahan yang dirasakan seperti konsep kecerdasan dari sebuah komputer, kecerdasan buatan memiliki kemampuan menerka suatu jawaban seperti pada perilaku otak manusia. Kecerdasan buatan dapat digunakan untuk mengidentifikasi kandungan asam cyanide (HCN) ubi kayu dan famili tumbuhan Obat. Dengan aplikasi identifikasi yang dirancang ini dapat membantu masyarakat untuk mempercepat dan tepat dalam pengambilan suatu keputusan. System identifikasi yang telah dirancang dapat dibuka pada alamat website https://indralaksmana.com/sistem-identifikasi/

Aplikasi yang dirancang dalam dua versi yatu berbasis web dan berbasis android. Semoga buku panduan penggunaan aplikasi ini dapat memberikan manfaat dan Semoga hasil penelitian ini nantinya dapat dimanfaatkan bagi pembangunan ilmu pengetahuan serta bagi masyarakat luas.

Penulis

# DAFTAR ISI

## DAFTAR GAMBAR

## 1. PENDAHULUAN

Singkong atau ubikayu (Manihot Utilissima Crantz) merupakan salah satu sumber karbohidrat lokal Indonesia yang menduduki urutan ketiga terbesar setelah padi dan jagung. Tanaman ini merupakan bahan baku yang paling potensial untuk diolah, baik pengolahan jadi bahan makanan sayuran, pakan ternak, keripik dan lain-lain ataupun bioetanol melalui proses fermentasi. Komoditi unggulan Kabupaten Lima Puluh Kota yang capaian produksi yang tertinggi selain Komoditi Padi adalah Ubi kayu. Kabupaten Lima Puluh Kota khususnya Provinsi Sumatera Barat memiliki makanan khas seperti keripik sanjai, keripik balado, getuk, kacimuih dan masih banyak yang lain yang bahan bakunya adalah ubi kayu. Kebutuhan akan varietas ubi kayu unggul sebagai bahan baku, sangat di harapkan untuk menghasilkan produk yang berkualitas. Aplikasi ini dibuat untuk menghindari kesalahan dalam memilih bahan baku, karena salah dalam pemilihan bahan baku akan mempengaruhi produk yang dihasilkan.

Dokumen user manual ini dibuat dari dua hasil penelitian yang berjudul Rancang Bangun Model Sistem Identifikasi Ubi Kayu (Manihot Utilissima Crantz) Untuk Mengklasifikasi Varietas Unggul Tanaman dan Pemrograman Genetika untuk Sistem Identifikasi Famili Tumbuhan Obat. Dua penelitian ini menghasilkan rull atau model identifikasi. Model atau rule yang dihasilkan tersebut diterjemahkan menggunakan aplikasi yang dibuat ini, sehingga seluruh masyarakat yang membutuhkan proses identifikasi dapat menggunakan aplikasi ini.

Tujuan pembuatan aplikasi ini adalah mempermudah dan membantu pengguna dalam menterjemah rull atau model yang dihasilkan dari hasil penelitian dalam mengidentifikasi kandungan HCN ubi kayu serta mengidentifikasi famili tumbuhan obat. Pihak-pihak yang membutuhkan dokumen ini adalah petani, peternak, pengusa yang berbahan baku ubi dan peneliti. Sehingga dengan adanya aplikasi ini, petani dapat mementukan penanaman berdasarkan kebutuhan dan memami peruntukan hasil untuk didistribusikan. Peternak dapat memutuskan tumbuhan mana atau jenis ubi mana yang dapat dikonsumsi sebagai pakan ternak. Pengusaha dapat memutuskan umbi yang akan dicari sebagai bahan bakunya, sehingga dapat menetapkan umbi mana yang akan di jadikan tepung, tapai atau keripik maupun kerupuk.

## 2. SUMBER DAYA YANG DIBUTUHKAN

### Perangkat Lunak

Perangkat lunak yang digunakan dalam pengoprasian aplikasi berbasis web adalah aplikasi browser seperti Mozila Firefox, Google Chrom atau aplikasi browser lainnya. Sedangan aplikasi berbasis android adalah software android (apk), Sistem operasi android versi 4.2 (jelly bean) ke atas (versi 4.3, Android Kitkat version 4.4, Lollipop Version 5.0, Marshmallow version 6.0, Nougat Version dan seterusnya)

### Perangkat keras

Perangkat keras yang dibutuhkan dalam pengoprasian aplikasi ini adalah

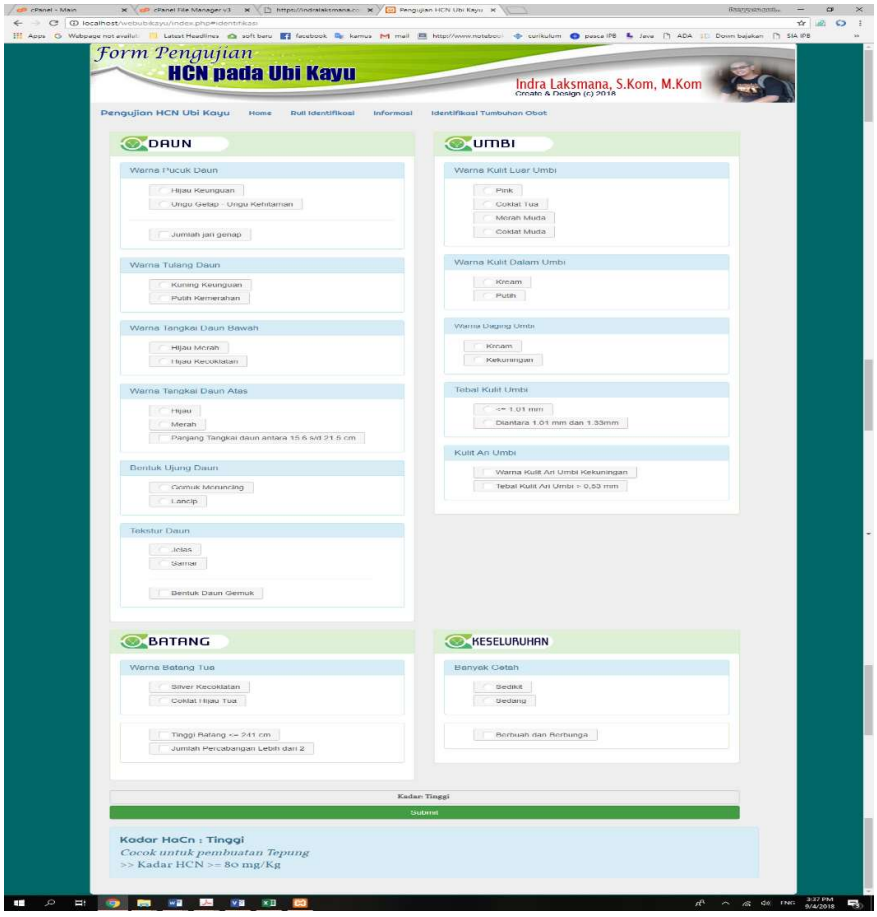
1. Komputer atau Laptop yang memiliki koneksi jaringan internet.
2. Mouse sebagai peralatan antarmuka
3. Monitor sebagai peralatan antarmuka
4. Keyboard sebagai peralatan antarmuka.
5. Tablet atau smart phone yang menggunakan system operasi android

### Sumber Daya Manusia

Aplikasi yang telah dirancang sesederhana mungkin, sehingga dalam proses pengoprasian aplikasi ini tidak membutuhkan sumber daya manusia yang spesifik, hanya sedikit memiliki pemahaman tentang komputer dan android.

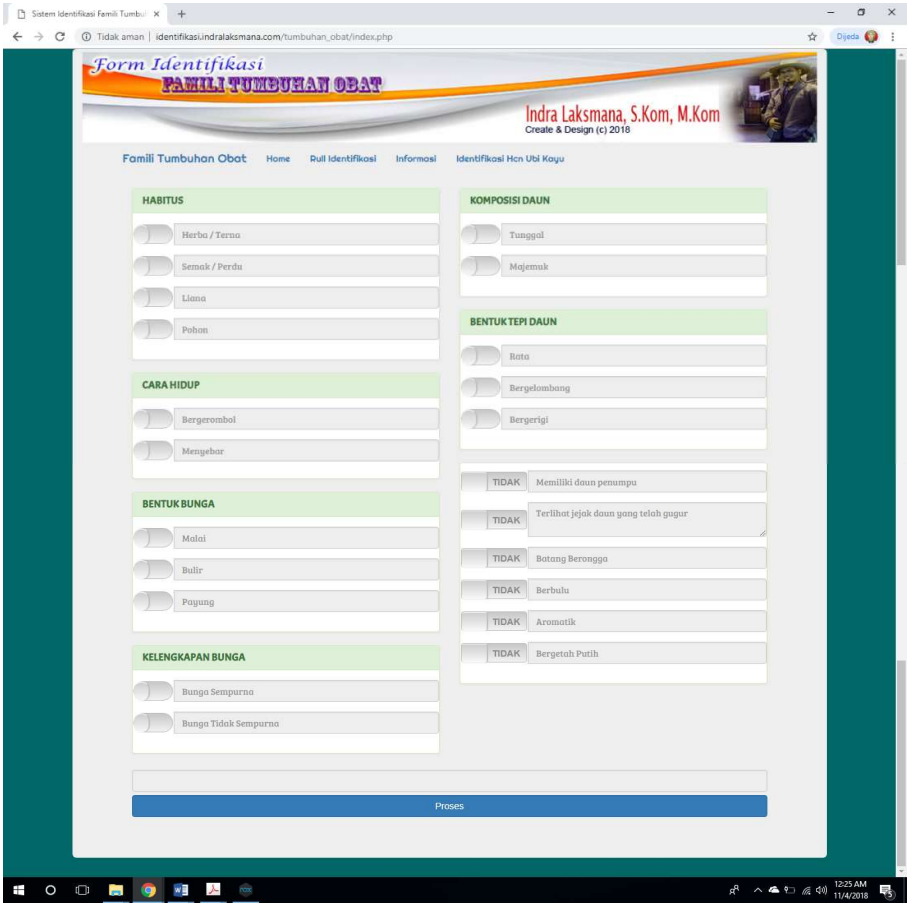
## 3. CARA PENGGUNAAN

Penggunaan aplikasi berbasis web dan android secara umum pengoprasiannya sama. Menu yang telah kami rancang juga tidak jauh berbeda. Seperti terlihat pada Gamabar 1, 2 dan Gambar 3, 4 di bawah ini.

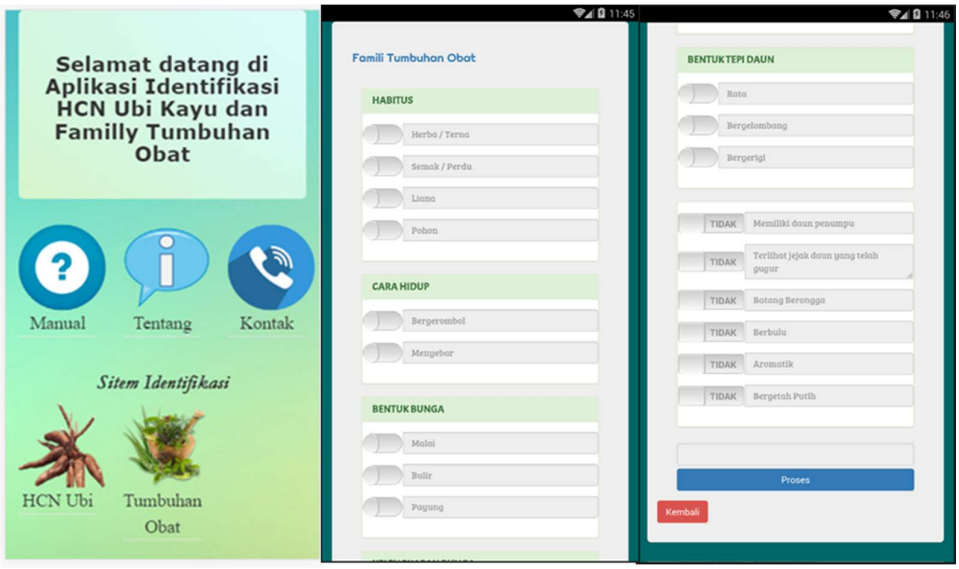



Gambar 1. Tampilan Aplikasi Identifikasi kandungan HCN ubi kayu berbasis web


Gambar 2. Tampilan Aplikasi Identifikasi kandungan HCN ubi kayu berbasis Android



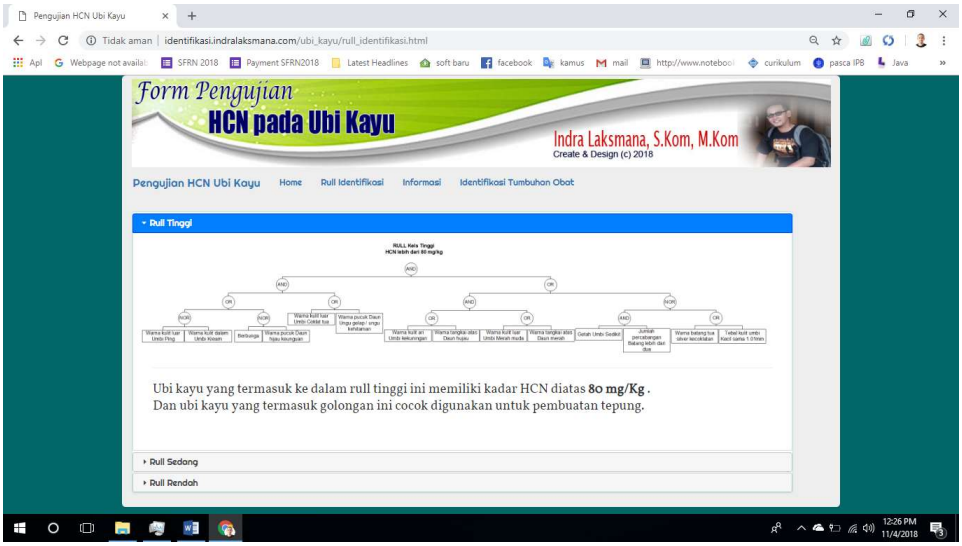

Gambar 3. Tampilan Aplikasi Identifikasi Famili tumbuhan obat berbasis web




Gambar 4. Tampilan Aplikasi Identifikasi Famili tumbuhan obat berbasis Android

Terlihat pada Gambar 1 Tampilan aplikasi system identifikasi kandungan HCN berbasis web, terdapat 4 empat menu. Menu tersebut berfungsi sebagai berikut;

a. Pengujian HCN Ubi Kayu: Menu ini berfungsi untuk melakukan refresh kembali jika terjadi kesalahan input.
b. Home: Menu ini mengembalikan atau melinkkan ke tampilan awal (http://indralaksmana.com )
c. Rull Identifikasi: Menu ini akan menampilkan rull atau aturan identifikasi yang menjadi acuan dalam aplikasi ini.




Gambar 5. Rull atau model Identifikasi

d. Informasi: Menu ini akan menampilkan Infromasi, tujuan, manfaat dan cara penggunaan rancangan aplikasi ini dibuat. Pada menu informasi ini juga terdapat link download untuk aplikasi android nya.
e. Identifikasi Tumbuhan Obat: Menu ini adalah link ke aplikasi system identifikasi family tumbuhan obat.

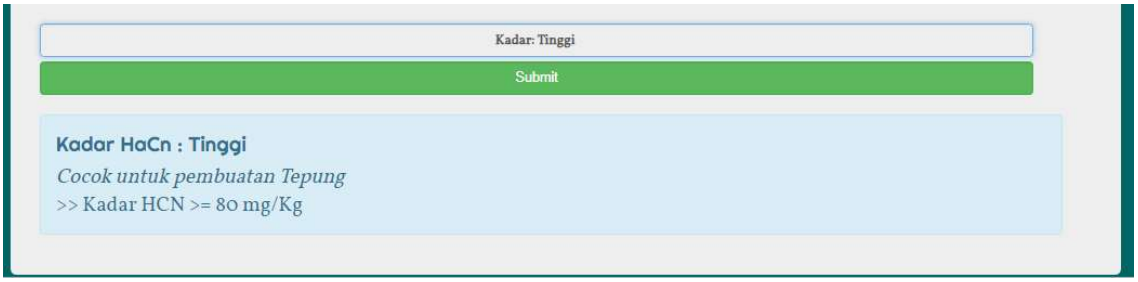
Selanjutnya pada form pengujian terdapat empat kelompok yaitu kelompok daun, kelompok batang, kelompok umbi dan kelompok keseluruhan. Sedangkan pada aplikasi pengujian identifikasi family tumbuhan obat terapat enam kelompok yaitu kelompok habitus, cara hidup, bentuk bunga, kelengkapan bunga, komposisi daun dan bentuk tepi daun. Kelompok ini adalah bagian terbesar dari tanaman yang akan di indentifikasi. Masing-masing kelompok tedapat pertanyaaan yang mesti dipilih sesuai tanaman yang diperhatikan akan diidentifikasi.

Gambar 6. Pucuk daun


Gambar 7. Opsi Warna pucuk daun

Terlihat pada Gambar 6 dan 7 di atas warna pucuk tanaman yang akan di identifikasi hitam gelap atau ungu kehitaman, maka pada aplikasi haya mengklik opsi yang ungu gelap-ungu kehitaman 

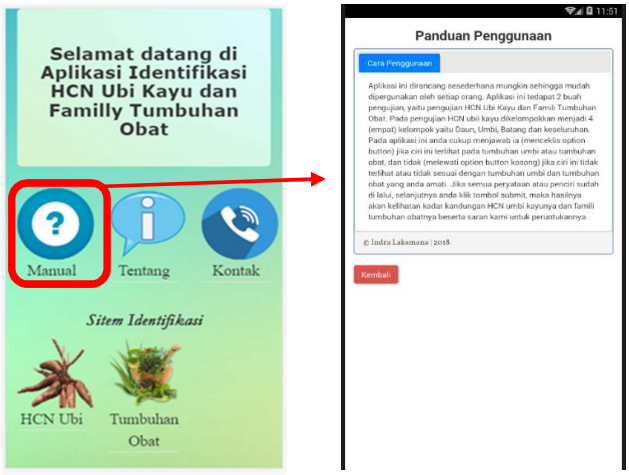
. Terlihat jumlah jari tumbuhan tiga maka pada bagian aplikasi jumlah jari genap tidak diklik. Untuk melakukan identifikasi tanaman, pengguna hanya memilih salah satu opsi yang tersedia, jika pilihan opsi tidak ditemukan, pengguna boleh tidak memilih. Hasil identifikasi akan tampil jika pengguna telah memilih opsi yang tersedia dan menekan tombol submit pada bagian akhir opsi, seperti terlihat pada Gambar 8 di bawah ini.



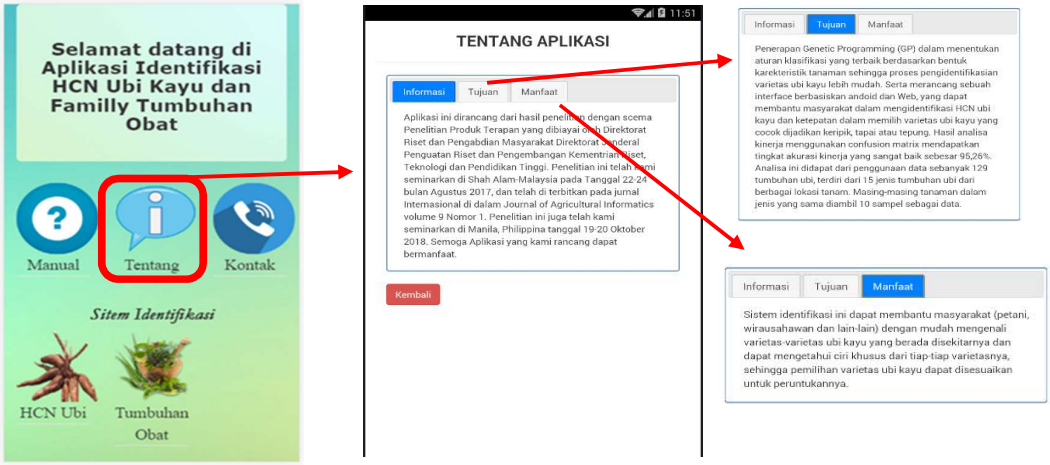

Gambar 8. Hasil identifikasi dan Informasi yang dihasilkan aplikasi

Hasil yang mungkin terjadi setelah pengguna mengklik submit pada aplikasi identifikasi kandungan HCN ubi kayu adalah Kadar HCN tinggi, sedang dan rendah atau kombinasi dari ketiganya. Pada bagian bawah terdapat informasi kandungan HCN dan saran yang paling cocok untuk apa ubi kayu digunakan.

Tampilan yang tidak jauh berbeda jika menggunakan aplikasi berbasis android. Menu yang ada pada tambilan android terlihat pada Gambar 9,10 dan Gambar 11 di bawah ini




Gambar 9. Tampilan aplikasi Android pada Menu Manual




Gambar 10. Tampilan aplikasi Android pada Menu Tentang


Gambar 11. Tampilan aplikasi Android pada Menu Kontak

# Source Code Aplikasi

SISTEM IDENTIFIKASI KANDUNGAN ASAM CYANIDE (HCN) UBI KAYU DAN FAMILI TUMBUHAN OBAT

TIM Penyusun:

Indra Laksmana, S.Kom, M.Kom
Rosda Syelly, S.Kom, M.Kom
Ir. Nurzarrah Tazar, MP
Perdana Putera, ST, M.Eng
Amrizal, S.Kom, M.Kom

```
package io.gonative.android;

import android.Manifest;
import android.annotation.TargetApi;
import android.app.Dialog;
import android.content.BroadcastReceiver;
import android.content.ClipData;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.ActivityInfo;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.graphics.Color;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import
android.support.v4.content.LocalBroadcastManager;
import android.support.v4.view.GravityCompat;
import android.support.v4.view.ViewCompat;
import android.support.v4.view.ViewPager;
import android.support.v4.widget.DrawerLayout;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.SearchView;
import android.telephony.PhoneStateListener;
import android.telephony.SignalStrength;
import android.telephony.TelephonyManager;
import android.util.Base64;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewParent;
import android.view.WindowManager;
import android.webkit.CookieManager;
import android.webkit.CookieSyncManager;
import android.webkit.JavascriptInterface;
import android.webkit.ValueCallback;
import android.webkit.WebChromeClient;
import android.widget.ExpandableListView;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.RelativeLayout;
import android.widget.Spinner;
import android.widget.Toast;

import com.astuetz.PagerSlidingTabStrip;
import com.facebook.appevents.AppEventsLogger;
import com.facebook.applinks.AppLinkData;
import com.onesignal.OneSignal;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.File;
import java.io.UnsupportedEncodingException;
import java.net.CookieHandler;
import java.net.URISyntaxException;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Observable;
import java.util.Observer;
import java.util.Stack;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import io.gonative.android.library.AppConfig;

public class MainActivity extends AppCompatActivity
implements Observer,
SwipeRefreshLayout.OnRefreshListener {
   public static final String webviewCacheSubdir =
"webviewAppCache";
   private static final String webviewDatabaseSubdir =
"webviewDatabase";
        private static final String TAG =
MainActivity.class.getName();
   public static final String INTENT_TARGET_URL =
"targetUrl";
   public static final String
EXTRA_WEBVIEW_WINDOW_OPEN =
"io.gonative.android.MainActivity.Extra.WEBVIEW_WIN
DOW_OPEN";
        public static final int REQUEST_SELECT_FILE =
100;
   private static final int
REQUEST_PERMISSION_READ_EXTERNAL_STORAGE =
101;
   private static final int
REQUEST_PERMISSION_GEOLOCATION = 102;
```

```
    private static final int
REQUEST_PERMISSION_WRITE_EXTERNAL_STORAGE =
103;
    private static final int
REQUEST_PERMISSION_GENERIC = 199;
    private static final int REQUEST_WEBFORM = 300;
    public static final int REQUEST_WEB_ACTIVITY = 400;
    private static final float ACTIONBAR_ELEVATION =
12.0f;

    private GoNativeWebviewInterface mWebview;
    private View webviewOverlay;
    boolean isPoolWebview = false;
    private Stack<String> backHistory = new Stack<>();

        private ValueCallback<Uri> mUploadMessage;
    private ValueCallback<Uri[]> uploadMessageLP;
    private Uri directUploadImageUri;
        private DrawerLayout mDrawerLayout;
        private View mDrawerView;
        private ExpandableListView mDrawerList;
    private ProgressBar mProgress;
    private Dialog splashDialog;
    private boolean splashDismissRequiresForce;
    private MySwipeRefreshLayout swipeRefresh;
    private RelativeLayout fullScreenLayout;
    private JsonMenuAdapter menuAdapter = null;
        private ActionBarDrawerToggle mDrawerToggle;
    private PagerSlidingTabStrip slidingTabStrip;
    private ImageView navigationTitleImage;
        private ConnectivityManager cm = null;
    private ProfilePicker profilePicker = null;
    private TabManager tabManager;
    private ActionManager actionManager;
    private boolean isRoot;
    private float hideWebviewAlpha = 0.0f;
    private boolean isFirstHideWebview = true;
    private boolean webviewIsHidden = false;
    private int urlLevel = -1;
    private int parentUrlLevel = -1;
    private Handler handler = new Handler();
    private Runnable statusChecker = new Runnable() {
      @Override
      public void run() {
        runOnUiThread(new Runnable() {
          @Override
          public void run() {
            checkReadyStatus();
          }
        });
        handler.postDelayed(statusChecker, 100); // 0.1
sec
      }
    };
    private FileDownloader fileDownloader = new
FileDownloader(this);
    private boolean startedLoading = false; // document
readystate checker
    private LoginManager loginManager;
    private RegistrationManager registrationManager;
    private ConnectivityChangeReceiver
connectivityReceiver;
    private BroadcastReceiver
navigationTitlesChangedReceiver;
    protected String postLoadJavascript;
    protected String postLoadJavascriptForRefresh;
    private Stack<Bundle>previousWebviewStates;
    private GeolocationPermissionCallback
geolocationPermissionCallback;
    private ArrayList<PermissionsCallbackPair>
pendingPermissionRequests = new ArrayList<>();
    private ArrayList<Intent>
pendingStartActivityAfterPermissions = new
ArrayList<>();
    private String connectivityCallback;
    private String connectivityOnceCallback;
    private PhoneStateListener phoneStateListener;
    private SignalStrength latestSignalStrength;

    @Override
        protected void onCreate(Bundle
savedInstanceState) {
      AppConfig appConfig = AppConfig.getInstance(this);
      GoNativeApplication application =
(GoNativeApplication)getApplication();

      setScreenOrientationPreference();

      if (appConfig.keepScreenOn) {

getWindow().addFlags(WindowManager.LayoutParams.
FLAG_KEEP_SCREEN_ON);
      }

      this.hideWebviewAlpha  =
appConfig.hideWebviewAlpha;

      super.onCreate(savedInstanceState);

      isRoot = getIntent().getBooleanExtra("isRoot", true);
      parentUrlLevel =
getIntent().getIntExtra("parentUrlLevel", -1);

      if (isRoot) {
```

```
        // Splash screen stuff
        boolean isFromLauncher =
getIntent().hasCategory(Intent.CATEGORY_LAUNCHER);
        // FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY
does not seem to be set when it should
        // for some devices. I have yet to find a good
workaround.
        boolean isFromRecents = (getIntent().getFlags() &
Intent.FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY) !=
0;
        boolean noSplash =
getIntent().getBooleanExtra("noSplash", false);

        if (!noSplash && isFromLauncher &&
!isFromRecents) {

showSplashScreen(appConfig.showSplashMaxTime,
appConfig.showSplashForceTime);
        }

        // html5 app cache (manifest)
        File cachePath = new File(getCacheDir(),
webviewCacheSubdir);
        if (!cachePath.mkdirs()) {
           Log.v(TAG, "cachePath " + cachePath.toString()
+ " exists");
        }
        File databasePath = new File(getCacheDir(),
webviewDatabaseSubdir);
        if (databasePath.mkdirs()) {
           Log.v(TAG, "databasePath " +
databasePath.toString() + " exists");

        }

        // url inspector
        UrlInspector.getInstance().init(this);

        // Register launch
        ConfigUpdater configUpdater = new
ConfigUpdater(this);
        configUpdater.registerEvent();

        // registration service
        this.registrationManager =
application.getRegistrationManager();
    }

    this.loginManager = application.getLoginManager();

    // webview pools
    application.getWebViewPool().init(this);

                cm = (ConnectivityManager)
getSystemService(CONNECTIVITY_SERVICE);

    if (isRoot &&
AppConfig.getInstance(this).showNavigationMenu)

        setContentView(R.layout.activity_gonative);
    else

setContentView(R.layout.activity_gonative_nonav);


    mProgress = findViewById(R.id.progress);
    this.fullScreenLayout =
findViewById(R.id.fullscreen);

    swipeRefresh = findViewById(R.id.swipe_refresh);
    swipeRefresh.setEnabled(appConfig.pullToRefresh);
    swipeRefresh.setOnRefreshListener(this);
    swipeRefresh.setCanChildScrollUpCallback(new
MySwipeRefreshLayout.CanChildScrollUpCallback() {
      @Override
      public boolean canSwipeRefreshChildScrollUp() {
        return mWebview.getScrollY() > 0;
      }
    });
    if (appConfig.pullToRefreshColor != null) {

swipeRefresh.setColorSchemeColors(appConfig.pullToRe
freshColor);
    }

    this.webviewOverlay =
findViewById(R.id.webviewOverlay);
    this.mWebview = findViewById(R.id.webview);
    setupWebview(this.mWebview);

    // profile picker
    if (isRoot &&
AppConfig.getInstance(this).showNavigationMenu) {
        Spinner profileSpinner =
findViewById(R.id.profile_picker);
        profilePicker = new ProfilePicker(this,
profileSpinner);

        Spinner segmentedSpinner =
findViewById(R.id.segmented_control);
        new SegmentedController(this,
segmentedSpinner);
    }
```

```
                // to save webview cookies to
permanent storage

        CookieSyncManager.createInstance(getApplicati
onContext());

                // proxy cookie manager for
httpUrlConnection (syncs to webview cookies)
    CookieHandler.setDefault(new
WebkitCookieManagerProxy());



    this.postLoadJavascript =
getIntent().getStringExtra("postLoadJavascript");
    this.postLoadJavascriptForRefresh =
this.postLoadJavascript;

    this.previousWebviewStates = new Stack<>();

    // tab navigation
    ViewPager pager = findViewById(R.id.view_pager);
    this.slidingTabStrip = findViewById(R.id.tabs);
    this.tabManager = new TabManager(this, pager);
    pager.setAdapter(this.tabManager);
    this.slidingTabStrip.setViewPager(pager);

this.slidingTabStrip.setTabClickListener(this.tabManager)
;

    // custom colors
    if (appConfig.tabBarBackgroundColor != null)

this.slidingTabStrip.setBackgroundColor(appConfig.tabB
arBackgroundColor);
    if (appConfig.tabBarTextColor != null)

this.slidingTabStrip.setTextColor(appConfig.tabBarTextC
olor);
    if (appConfig.tabBarIndicatorColor != null)

this.slidingTabStrip.setIndicatorColor(appConfig.tabBarIn
dicatorColor);
    hideTabs();

    if (!appConfig.showActionBar &&
getSupportActionBar() != null) {
        getSupportActionBar().hide();
    }

    // actions in action bar
    this.actionManager = new ActionManager(this);

    Intent intent = getIntent();
    // load url
    String url = null;
    // first check intent in case it was created from push
notification
    String targetUrl =
intent.getStringExtra(INTENT_TARGET_URL);
    if (targetUrl != null && !targetUrl.isEmpty()){
      url = targetUrl;
    }

    if (Intent.ACTION_VIEW.equals(intent.getAction())) {
      Uri uri = intent.getData();
      if (uri != null &&
(uri.getScheme().endsWith(".http") ||
uri.getScheme().endsWith(".https"))) {
        Uri.Builder builder = uri.buildUpon();
        if (uri.getScheme().endsWith(".https")) {
          builder.scheme("https");
        } else if (uri.getScheme().endsWith(".http")) {
          builder.scheme("http");
        }
        url = builder.build().toString();
      } else {
        url = intent.getDataString();
      }
    }

    if (url == null && savedInstanceState != null) url =
savedInstanceState.getString("url");
    if (url == null && isRoot) url = appConfig.initialUrl;
    // url from intent (hub and spoke nav)
    if (url == null) url = intent.getStringExtra("url");

    if (url != null) {
      // Crosswalk does not give us callbacks when
location is requested.
      // Ask for it up front, then load the page.
      if (LeanWebView.isCrosswalk() &&
appConfig.usesGeolocation) {
        final String urlLoadAfterLocation = url;

        this.getRuntimeGeolocationPermission(new
GeolocationPermissionCallback() {
          @Override
          public void onResult(boolean granted) {
            // ignore result
            mWebview.loadUrl(urlLoadAfterLocation);
          }
        });
      } else {
        this.mWebview.loadUrl(url);
```

```
        }
    } else if
(intent.getBooleanExtra(EXTRA_WEBVIEW_WINDOW_O
PEN, false)){
        // no worries, loadUrl will be called when this new
web view is passed back to the message
    } else {
        Log.e(TAG, "No url specified for MainActivity");
    }

    if (isRoot && appConfig.facebookEnabled) {
        AppLinkData.fetchDeferredAppLinkData(this, new
AppLinkData.CompletionHandler() {
            @Override
            public void
onDeferredAppLinkDataFetched(AppLinkData
appLinkData) {
                if (appLinkData == null) return;
                Uri uri = appLinkData.getTargetUri();
                if (uri == null) return;
                String url;
                if (uri.getScheme().endsWith(".http") ||
uri.getScheme().endsWith(".https")) {
                    Uri.Builder builder = uri.buildUpon();
                    if (uri.getScheme().endsWith(".https")) {
                        builder.scheme("https");
                    } else if (uri.getScheme().endsWith(".http"))
{
                        builder.scheme("http");
                    }
                    url = builder.build().toString();
                } else {
                    url = uri.toString();
                }
                if (url != null) {
                    final String finalUrl = url;
                    new
Handler(MainActivity.this.getMainLooper()).post(new
Runnable() {
                        @Override
                        public void run() {
                            mWebview.loadUrl(finalUrl);
                        }
                    });
                }
            }
        });
    }

    if (isRoot && appConfig.showNavigationMenu) {
        // do the list stuff
        mDrawerLayout =
findViewById(R.id.drawer_layout);
        mDrawerView = findViewById(R.id.left_drawer);
        mDrawerList = findViewById(R.id.drawer_list);

        // set shadow

mDrawerLayout.setDrawerShadow(R.drawable.drawer_
shadow, GravityCompat.START);

        mDrawerToggle = new
ActionBarDrawerToggle(this, mDrawerLayout,
                R.string.drawer_open,
R.string.drawer_close){
            //Called when a drawer has settled in a
completely closed state.
            public void onDrawerClosed(View view) {
                invalidateOptionsMenu(); // creates call to
onPrepareOptionsMenu()
            }

            //Called when a drawer has settled in a
completely open state.
            public void onDrawerOpened(View
drawerView) {
                invalidateOptionsMenu(); // creates call to
onPrepareOptionsMenu()
            }
        };

mDrawerLayout.addDrawerListener(mDrawerToggle);

        setupMenu();

        // update the menu
        if (appConfig.loginDetectionUrl != null) {
            this.loginManager.addObserver(this);
        }
    }

                    if (getSupportActionBar() != null) {
        if (!isRoot || appConfig.showNavigationMenu) {

getSupportActionBar().setDisplayHomeAsUpEnabled(tru
e);
        }


showLogoInActionBar(appConfig.shouldShowNavigation
TitleImageForUrl(url));
    }
```

```
    // style sidebar
    if (mDrawerView != null &&
AppConfig.getInstance(this).sidebarBackgroundColor !=
null) {

mDrawerView.setBackgroundColor(AppConfig.getInstan
ce(this).sidebarBackgroundColor);
    }

    // respond to navigation titles processed
    this.navigationTitlesChangedReceiver = new
BroadcastReceiver() {
      @Override
      public void onReceive(Context context, Intent
intent) {
        if
(AppConfig.PROCESSED_NAVIGATION_TITLES.equals(int
ent.getAction())) {
          String url = mWebview.getUrl();
          if (url == null) return;

          String title = titleForUrl(mWebview.getUrl());
          if (title == null) return;

          setTitle(title);
        }
      }
    };

LocalBroadcastManager.getInstance(this).registerReceiv
er(this.navigationTitlesChangedReceiver,
      new
IntentFilter(AppConfig.PROCESSED_NAVIGATION_TITLES
));
  }

  protected void onPause() {
    super.onPause();
    stopCheckingReadyStatus();
    this.mWebview.onPause();

    // unregister connectivity
    if (this.connectivityReceiver != null) {
      unregisterReceiver(this.connectivityReceiver);
    }

    if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {
      CookieManager.getInstance().flush();
    }

  }

  @Override
  protected void onStart() {
    super.onStart();
    if (AppConfig.getInstance(this).oneSignalEnabled) {
      OneSignal.clearOneSignalNotifications();
    }
  }

  @Override
  protected void onResume() {
    super.onResume();
    this.mWebview.onResume();

    retryFailedPage();
    // register to listen for connectivity changes
    this.connectivityReceiver = new
ConnectivityChangeReceiver();
    registerReceiver(this.connectivityReceiver,
        new
IntentFilter(ConnectivityManager.CONNECTIVITY_ACTIO
N));

    // check login status
    this.loginManager.checkLogin();

    if (AppConfig.getInstance(this).facebookEnabled) {
      AppEventsLogger.activateApp(getApplication());
    }
  }

  @Override
  protected void onStop() {
    super.onStop();

    if (isRoot) {
      if (AppConfig.getInstance(this).clearCache) {
        this.mWebview.clearCache(true);
      }
    }
  }

  @Override
  protected void onDestroy() {
    super.onDestroy();

    // destroy webview
    if (this.mWebview != null) {
      this.mWebview.stopLoading();
      // must remove from view hierarchy to destroy
      ViewGroup parent = (ViewGroup)
this.mWebview.getParent();
```

```
        if (parent != null) {
            parent.removeView((View)this.mWebview);
        }
        if (!this.isPoolWebview) this.mWebview.destroy();
    }

    this.loginManager.deleteObserver(this);

    if (this.navigationTitlesChangedReceiver != null) {

LocalBroadcastManager.getInstance(this).unregisterRec
eiver(this.navigationTitlesChangedReceiver);
    }
  }

  private void retryFailedPage() {
    // skip if webview is currently loading
    if (this.mWebview.getProgress() < 100) return;

    // skip if webview has a page loaded
    String currentUrl = this.mWebview.getUrl();
    if (currentUrl != null &&
!currentUrl.equals("file:///android_asset/offline.html"))
return;

    // skip if there is nothing in history
    if (this.backHistory.isEmpty()) return;

    // skip if no network connectivity
    if (this.isDisconnected()) return;

    // finally, retry loading the page
    this.loadUrl(this.backHistory.pop());
  }

  protected void onSaveInstanceState (Bundle outState)
{
    outState.putString("url", mWebview.getUrl());
    outState.putInt("urlLevel", urlLevel);
    super.onSaveInstanceState(outState);
  }

  public void addToHistory(String url) {
    if (url == null) return;

    if (this.backHistory.isEmpty() ||
!this.backHistory.peek().equals(url)) {
      this.backHistory.push(url);
    }

    checkNavigationForPage(url);

    // this is a little hack to show the webview after
going back in history in single-page apps.
    // We may never get onPageStarted or
onPageFinished, hence the webview would be forever
    // hidden when navigating back in single-page apps.
We do, however, get an updatedHistory callback.
    showWebview(0.3);
  }

  private boolean canGoBack() {
    return this.mWebview.canGoBack();
  }

  private void goBack() {
    if (LeanWebView.isCrosswalk()) {
      // not safe to do for non-crosswalk, as we may
never get a page finished callback
      // for single-page apps
      hideWebview();
    }

    this.mWebview.goBack();
  }

  public void sharePage(String optionalUrl) {
    String shareUrl;
    String currentUrl = this.mWebview.getUrl();
    if (optionalUrl == null || optionalUrl.isEmpty()) {
      shareUrl = currentUrl;
    } else {
      try {
        java.net.URI optionalUri = new
java.net.URI(optionalUrl);
        if (optionalUri.isAbsolute()) {
          shareUrl = optionalUrl;
        } else {
          java.net.URI currentUri = new
java.net.URI(currentUrl);
          shareUrl =
currentUri.resolve(optionalUri).toString();
        }
      } catch (URISyntaxException e) {
        shareUrl = optionalUrl;
      }
    }

    if (shareUrl == null || shareUrl.isEmpty()) return;

    Intent share = new Intent(Intent.ACTION_SEND);
    share.setType("text/plain");
    share.putExtra(Intent.EXTRA_TEXT, shareUrl);
```

```
        startActivity(Intent.createChooser(share,
getString(R.string.action_share)));
    }

    private void logout() {
        this.mWebview.stopLoading();

        // log out by clearing all cookies and going to home
page
        CookieManager cookieManager =
CookieManager.getInstance();
        cookieManager.removeAllCookie();
        CookieSyncManager.getInstance().sync();

        updateMenu(false);
        this.loginManager.checkLogin();

this.mWebview.loadUrl(AppConfig.getInstance(this).initi
alUrl);
    }

    public void loadUrl(String url) {
        loadUrl(url, false);
    }

    public void loadUrl(String url, boolean isFromTab) {
        if (url == null) return;

        this.postLoadJavascript = null;
        this.postLoadJavascriptForRefresh = null;

        if (url.equalsIgnoreCase("gonative_logout"))
            logout();
        else
            this.mWebview.loadUrl(url);

        if (!isFromTab && this.tabManager != null)
this.tabManager.selectTab(url, null);
    }

    public void loadUrlAndJavascript(String url, String
javascript) {
        loadUrlAndJavascript(url, javascript, false);
    }

    public void loadUrlAndJavascript(String url, String
javascript, boolean isFromTab) {
        String currentUrl = this.mWebview.getUrl();

        if (url != null && currentUrl != null &&
url.equals(currentUrl)) {
//          hideWebview();
            runJavascript(javascript);
            this.postLoadJavascriptForRefresh = javascript;
//          showWebview();
        } else {
            this.postLoadJavascript = javascript;
            this.postLoadJavascriptForRefresh = javascript;
            this.mWebview.loadUrl(url);
        }

        if (!isFromTab && this.tabManager != null)
this.tabManager.selectTab(url, javascript);
    }

    public void runJavascript(String javascript) {
        if (javascript == null) return;
        this.mWebview.runJavascript(javascript);
    }

            public boolean isDisconnected(){
                    NetworkInfo ni =
cm.getActiveNetworkInfo();
        return ni == null || !ni.isConnected();
            }

            // configures webview settings
            private void
setupWebview(GoNativeWebviewInterface wv){
        WebViewSetup.setupWebviewForActivity(wv, this);
            }

    private void showSplashScreen(double maxTime,
double forceTime) {
        splashDialog = new Dialog(this,
R.style.SplashScreen);
        if (splashDialog.getWindow() != null) {

splashDialog.getWindow().getAttributes().windowAnima
tions = R.style.SplashScreenAnimation;
        }

splashDialog.setContentView(R.layout.splash_screen);
        splashDialog.setCancelable(false);
        splashDialog.show();

        double delay;

        if (forceTime > 0) {
            delay = forceTime;
            splashDismissRequiresForce = true;
        } else {
            delay = maxTime;
            splashDismissRequiresForce = false;
```

```
    }

    new Handler().postDelayed(new Runnable() {
      @Override
      public void run() {
        hideSplashScreen(true);
      }
    }, (long) (delay * 1000));
  }

  private void hideSplashScreen(boolean isForce) {
    if (splashDialog != null &&
(!splashDismissRequiresForce || isForce)) {
      splashDialog.dismiss();
      splashDialog = null;
    }
  }

  public void hideWebview() {
    if (AppConfig.getInstance(this).disableAnimations)
return;

    this.webviewIsHidden = true;
    mProgress.setAlpha(1.0f);
    mProgress.setVisibility(View.VISIBLE);

    if (this.isFirstHideWebview) {
      this.webviewOverlay.setAlpha(1.0f);
    } else {
      this.webviewOverlay.setAlpha(1 -
this.hideWebviewAlpha);
    }
  }

  private void showWebview(double delay) {
    hideSplashScreen(false);

    if (delay > 0) {
      handler.postDelayed(new Runnable() {
        @Override
        public void run() {
          showWebview();
        }
      }, (int) (delay * 1000));
    } else {
      showWebview();
    }
  }

  // shows webview with no animation
  public void showWebviewImmediately() {
    hideSplashScreen(false);

    this.isFirstHideWebview = false;
    webviewIsHidden = false;
    startedLoading = false;
    stopCheckingReadyStatus();
    this.webviewOverlay.setAlpha(0.0f);
    this.mProgress.setVisibility(View.INVISIBLE);

    if (Build.VERSION.SDK_INT ==
Build.VERSION_CODES.KITKAT) {
      injectCSSviaJavascript();
    }
  }

  public void showWebview() {
    hideSplashScreen(false);

    this.isFirstHideWebview = false;
    startedLoading = false;
    stopCheckingReadyStatus();

    if (!webviewIsHidden) {
      // don't animate if already visible
      mProgress.setVisibility(View.INVISIBLE);
      return;
    }

    if (Build.VERSION.SDK_INT ==
Build.VERSION_CODES.KITKAT) {
      injectCSSviaJavascript();
    }

    webviewIsHidden = false;

    webviewOverlay.animate().alpha(0.0f)
        .setDuration(300)
        .setStartDelay(150);

    mProgress.animate().alpha(0.0f)
        .setDuration(60);
  }

  private void injectCSSviaJavascript() {
    AppConfig appConfig = AppConfig.getInstance(this);
    if (appConfig.customCSS == null ||
appConfig.customCSS.isEmpty()) return;

    try {
      String encoded =
Base64.encodeToString(appConfig.customCSS.getBytes("
utf-8"), Base64.NO_WRAP);
      String js = "(function() {" +
```

```
            "var parent =
document.getElementsByTagName('head').item(0);" +
            "var style =
document.createElement('style');" +
            "style.type = 'text/css';" +
            // Tell the browser to BASE64-decode the
string into your script !!!
            "style.innerHTML = window.atob('" +
encoded + "');" +
            "parent.appendChild(style)" +
            "})()";
        runJavascript(js);
      } catch (Exception e) {
        Log.e(TAG, "Error injecting customCSS via
javascript", e);
      }
    }

    public void showLogoInActionBar(boolean show) {
      ActionBar actionBar = getSupportActionBar();
      if (actionBar == null) return;

      actionBar.setDisplayOptions(show ? 0 :
ActionBar.DISPLAY_SHOW_TITLE,
ActionBar.DISPLAY_SHOW_TITLE);

      if (show) {
        // disable text title
        actionBar.setDisplayOptions(0,
ActionBar.DISPLAY_SHOW_TITLE);

        // why use a custom view and not
setDisplayUseLogoEnabled and setLogo?
        // Because logo doesn't work!
        actionBar.setDisplayShowCustomEnabled(true);
        if (this.navigationTitleImage == null) {
          this.navigationTitleImage = new
ImageView(this);

this.navigationTitleImage.setImageResource(R.drawable.
ic_actionbar);
        }

actionBar.setCustomView(this.navigationTitleImage);
      } else {

actionBar.setDisplayOptions(ActionBar.DISPLAY_SHOW_
TITLE, ActionBar.DISPLAY_SHOW_TITLE);
        actionBar.setDisplayShowCustomEnabled(false);
      }
    }
```

```
        public void updatePageTitle() {
      if (AppConfig.getInstance(this).useWebpageTitle) {
        setTitle(this.mWebview.getTitle());
      }
    }

    public void update (Observable sender, Object data) {
      if (sender instanceof LoginManager) {
        updateMenu(((LoginManager)
sender).isLoggedIn());
      }
    }

        public void updateMenu(){
      this.loginManager.checkLogin();
        }

    private void updateMenu(boolean isLoggedIn){
      if (menuAdapter == null)
        setupMenu();

      try {
        if (isLoggedIn)
          menuAdapter.update("loggedIn");
        else
          menuAdapter.update("default");
      } catch (Exception e) {
        Log.e(TAG, e.getMessage(), e);
      }
    }

        private boolean isDrawerOpen() {
      return mDrawerLayout != null &&
mDrawerLayout.isDrawerOpen(mDrawerView);
    }

    private void setDrawerEnabled(boolean enabled) {
      if (!isRoot) return;

      AppConfig appConfig = AppConfig.getInstance(this);
      if (!appConfig.showNavigationMenu) return;

      if (mDrawerLayout != null) {
        mDrawerLayout.setDrawerLockMode(enabled ?
DrawerLayout.LOCK_MODE_UNLOCKED :

DrawerLayout.LOCK_MODE_LOCKED_CLOSED);
      }

      ActionBar actionBar = getSupportActionBar();
      if (actionBar != null) {
        actionBar.setDisplayHomeAsUpEnabled(enabled);
```

```
    }
  }

        private void setupMenu(){
    menuAdapter = new JsonMenuAdapter(this);
    try {
      menuAdapter.update("default");
      mDrawerList.setAdapter(menuAdapter);
    } catch (Exception e) {
      Log.e(TAG, "Error setting up menu", e);
    }


mDrawerList.setOnGroupClickListener(menuAdapter);

mDrawerList.setOnChildClickListener(menuAdapter);
        }


        @Override
        protected void onPostCreate(Bundle
savedInstanceState) {

        super.onPostCreate(savedInstanceState);
                // Sync the toggle state after
onRestoreInstanceState has occurred.
    if (mDrawerToggle != null)
                    mDrawerToggle.syncState();
        }

  @Override
  public void onConfigurationChanged(Configuration
newConfig) {
    super.onConfigurationChanged(newConfig);
   // Pass any configuration change to the drawer
toggles
    if (mDrawerToggle != null)

mDrawerToggle.onConfigurationChanged(newConfig);
  }

        @Override
  @TargetApi(21)
  // Lollipop target API for
REQEUST_SELECT_FILE_LOLLIPOP
  protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
    if (data != null && data.getBooleanExtra("exit",
false))
      finish();

    String url = null;
    boolean success = false;
    if (data != null) {
      url = data.getStringExtra("url");
      success = data.getBooleanExtra("success", false);
    }

    if (requestCode == REQUEST_WEBFORM &&
resultCode == RESULT_OK) {
      if (url != null)
        loadUrl(url);
      else {
        // go to initialURL without login/signup override
        this.mWebview.setCheckLoginSignup(false);

this.mWebview.loadUrl(AppConfig.getInstance(this).initi
alUrl);
      }

      if
(AppConfig.getInstance(this).showNavigationMenu) {
        updateMenu(success);
      }
    }

    if (requestCode == REQUEST_WEB_ACTIVITY &&
resultCode == RESULT_OK) {
      if (url != null) {
        int urlLevel = data.getIntExtra("urlLevel", -1);
        if (urlLevel == -1 || parentUrlLevel == -1 ||
urlLevel > parentUrlLevel) {
          // open in this activity
          this.postLoadJavascript =
data.getStringExtra("postLoadJavascript");
          loadUrl(url);
        } else {
          // urlLevel <= parentUrlLevel, so pass up the
chain
          setResult(RESULT_OK, data);
          finish();
        }
      }
    }

    if (requestCode == REQUEST_SELECT_FILE) {
      if (resultCode != RESULT_OK) {
        cancelFileUpload();
        return;
      }

      // from documents (and video camera)
      if (data != null && data.getData() != null) {
        if (mUploadMessage != null) {
```

```
mUploadMessage.onReceiveValue(data.getData());
            mUploadMessage = null;
          }

          if (uploadMessageLP != null) {

uploadMessageLP.onReceiveValue(WebChromeClient.FileChooserParams.parseResult(resultCode, data));
            uploadMessageLP = null;
          }

          return;
        }

        // we may get clip data for multi-select documents
        if (data != null && data.getClipData() != null) {
          ClipData clipData = data.getClipData();
          ArrayList<Uri> files = new ArrayList<>(clipData.getItemCount());
          for (int i = 0; i < clipData.getItemCount(); i++) {
            ClipData.Item item = clipData.getItemAt(i);
            if (item.getUri() != null) {
              files.add(item.getUri());
            }
          }

          if (mUploadMessage != null) {
            // shouldn never happen, but just in case, send the first item
            if (files.size() > 0) {

mUploadMessage.onReceiveValue(files.get(0));
            } else {
              mUploadMessage.onReceiveValue(null);
            }
            mUploadMessage = null;
          }

          if (uploadMessageLP != null) {

uploadMessageLP.onReceiveValue(files.toArray(new Uri[files.size()]));
            uploadMessageLP = null;
          }

          return;
        }

        // from camera
        if (this.directUploadImageUri != null) {
          // check if we have external storage permissions
          if (ContextCompat.checkSelfPermission(this, android.Manifest.permission.READ_EXTERNAL_STORAGE) !=
              PackageManager.PERMISSION_GRANTED) {
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,

android.Manifest.permission.READ_EXTERNAL_STORAGE)) {
              Toast.makeText(this, R.string.external_storage_explanation, Toast.LENGTH_LONG).show();
            }

            ActivityCompat.requestPermissions(this,
                new String[]{android.Manifest.permission.READ_EXTERNAL_STORAGE},

REQUEST_PERMISSION_READ_EXTERNAL_STORAGE);
            // wait for the onRequestPermissionsResult callback
            return;
          }


          if (mUploadMessage != null) {

mUploadMessage.onReceiveValue(this.directUploadImageUri);
            mUploadMessage = null;
          }
          if (uploadMessageLP != null) {
            uploadMessageLP.onReceiveValue(new Uri[]{this.directUploadImageUri});
            uploadMessageLP = null;
          }
          this.directUploadImageUri = null;

          return;
        }

        // Should not reach here.
        cancelFileUpload();
      }
    }

    public void cancelFileUpload() {
      if (mUploadMessage != null) {
```

```
        mUploadMessage.onReceiveValue(null);
        mUploadMessage = null;
      }

      if (uploadMessageLP != null) {
        uploadMessageLP.onReceiveValue(null);
        uploadMessageLP = null;
      }

      this.directUploadImageUri = null;
    }

    @Override
    protected void onNewIntent(Intent intent) {
      String targetUrl =
intent.getStringExtra(INTENT_TARGET_URL);
      if (targetUrl != null && !targetUrl.isEmpty()){
        loadUrl(targetUrl);
      }
    }

    @Override
          public boolean onKeyDown(int keyCode,
KeyEvent event) {
                    if ((keyCode ==
KeyEvent.KEYCODE_BACK)) {
        if (this.mWebview.exitFullScreen()) {
          return true;
        }

                              if (isDrawerOpen()){

          mDrawerLayout.closeDrawers();
                                        return true;
                              }
        else if (canGoBack()) {
          goBack();
          return true;
        }
        else if (!this.previousWebviewStates.isEmpty()) {
          Bundle state = previousWebviewStates.pop();
          LeanWebView webview = new
LeanWebView(this);
          webview.restoreStateFromBundle(state);
          switchToWebview(webview, /* isPool */ false,
/* isBack */ true);
          return true;
        }
                    }

                    return super.onKeyDown(keyCode,
event);
          }

    // isPoolWebView is used to keep track of whether we
are showing a pooled webview, which has implications
    // for page navigation, namely notifying the pool to
disown the webview.
    // isBack means the webview is being switched in as
part of back navigation behavior. If isBack=false,
    // then we will save the state of the old one switched
out.
    public void
switchToWebview(GoNativeWebviewInterface
newWebview, boolean isPoolWebview, boolean isBack) {
      setupWebview(newWebview);

      // scroll to top
      ((View)newWebview).scrollTo(0, 0);

      View prev = (View)this.mWebview;

      if (!isBack) {
        // save the state for back button behavior
        Bundle stateBundle = new Bundle();
        this.mWebview.saveStateToBundle(stateBundle);
        this.previousWebviewStates.add(stateBundle);
      }

      // replace the current web view in the parent with
the new view
      if (newWebview != prev) {
        // a view can only have one parent, and
attempting to add newWebview if it already has
        // a parent will cause a runtime exception. So be
extra safe by removing it from its parent.
        ViewParent temp = newWebview.getParent();
        if (temp instanceof  ViewGroup) {
          ((ViewGroup)
temp).removeView((View)newWebview);
        }

        ViewGroup parent = (ViewGroup)
prev.getParent();
        int index = parent.indexOfChild(prev);
        parent.removeView(prev);
        parent.addView((View) newWebview, index);

((View)newWebview).setLayoutParams(prev.getLayoutP
arams());

        // webviews can still send some extraneous
events to this activity if we do not remove
        // its callbacks
```

```
        WebViewSetup.removeCallbacks((LeanWebView)
prev);

        if (!this.isPoolWebview) {
          ((GoNativeWebviewInterface)prev).destroy();
        }
      }

      this.isPoolWebview = isPoolWebview;
      this.mWebview = newWebview;

      if (this.postLoadJavascript != null) {
        runJavascript(this.postLoadJavascript);
        this.postLoadJavascript = null;
      }
    }

          @Override
          public boolean onCreateOptionsMenu(Menu
menu) {
                    // Inflate the menu; this adds items to
the action bar if it is present.

          getMenuInflater().inflate(R.menu.topmenu,
menu);

      AppConfig appConfig = AppConfig.getInstance(this);

      // search item in action bar
                    final MenuItem searchItem =
menu.findItem(R.id.action_search);
      if (appConfig.searchTemplateUrl != null) {
        // make it visible
        searchItem.setVisible(true);

        final SearchView searchView = (SearchView)
searchItem.getActionView();
        if (searchView != null) {
          SearchView.SearchAutoComplete
searchAutoComplete =
searchView.findViewById(android.support.v7.appcompa
t.R.id.search_src_text);
          if (searchAutoComplete != null) {

searchAutoComplete.setTextColor(appConfig.actionbarF
oregroundColor);
              int hintColor =
appConfig.actionbarForegroundColor;
              hintColor = Color.argb(192,
Color.red(hintColor), Color.green(hintColor),
                  Color.blue(hintColor));
searchAutoComplete.setHintTextColor(hintColor);
          }

          ImageView closeButtonImage =
searchView.findViewById(android.support.v7.appcompa
t.R.id.search_close_btn);
          if (closeButtonImage != null) {

closeButtonImage.setColorFilter(appConfig.actionbarFor
egroundColor);
          }


          // listener to process query
          searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String
query) {
              searchItem.collapseActionView();

              try {
                String q = URLEncoder.encode(query,
"UTF-8");

loadUrl(AppConfig.getInstance(getApplicationContext()).
searchTemplateUrl + q);
              } catch (UnsupportedEncodingException e)
{
                return true;
              }

              return true;
            }

            @Override
            public boolean onQueryTextChange(String
newText) {
              // do nothing
              return true;
            }
          });

          // listener to collapse action view when soft
keyboard is closed

searchView.setOnQueryTextFocusChangeListener(new
View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View v, boolean
hasFocus) {
```

```
                if (!hasFocus) {
                    searchItem.collapseActionView();
                }
            }
        });
    }
}

if (!appConfig.showRefreshButton) {
    MenuItem refreshItem =
menu.findItem(R.id.action_refresh);
    if (refreshItem != null) {
        refreshItem.setVisible(false);
    }
}

if (this.actionManager != null) {
    this.actionManager.addActions(menu);
}

                return true;
        }

        @Override
        public boolean
onOptionsItemSelected(MenuItem item) {
    // Pass the event to ActionBarDrawerToggle, if it
returns
    // true, then it has handled the app icon touch event

    if (mDrawerToggle != null) {
        if (mDrawerToggle.onOptionsItemSelected(item))
{
         return true;
        }
    }

    // actions
    if (this.actionManager != null) {
        if
(this.actionManager.onOptionsItemSelected(item)) {
            return true;
        }
    }

    // handle other items
    switch (item.getItemId()){
        case android.R.id.home:
            finish();
            return true;
                case R.id.action_search:
                    return true;
                case R.id.action_refresh:
            onRefresh();
                    return true;
            default:
             return super.onOptionsItemSelected(item);
    }
}

@Override
public void onRefresh() {
    refreshPage();
    // let the refreshing spinner stay for a little bit if the
native show/hide is disabled
    // otherwise there isn't enough of a user
confirmation that the page is refreshing
    if (AppConfig.getInstance(this).disableAnimations) {
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                swipeRefresh.setRefreshing(false);
            }
        }, 1000); // 1 second
    } else {
        this.swipeRefresh.setRefreshing(false);
    }
}

private void refreshPage() {
    String url = this.mWebview.getUrl();
    if (url != null &&
url.startsWith("file:///android_asset/offline")){
        this.mWebview.goBack();
        updateMenu();
    }
    else {
        this.postLoadJavascript =
this.postLoadJavascriptForRefresh;
        this.mWebview.loadUrl(url);
    }
}

// onPageFinished
public void checkNavigationForPage(String url) {
    // don't change anything on navigation if the url that
just finished was a file download
    if
(url.equals(this.fileDownloader.getLastDownloadedUrl())
) return;

    if (this.tabManager != null) {
        this.tabManager.checkTabs(url);
    }
```

```
    if (this.actionManager != null) {
      this.actionManager.checkActions(url);
    }

    if (this.registrationManager != null) {
      this.registrationManager.checkUrl(url);
    }
  }

  // onPageStarted
  public void checkPreNavigationForPage(String url) {
    if (this.tabManager != null) {
      this.tabManager.autoSelectTab(url);
    }

    AppConfig appConfig = AppConfig.getInstance(this);

setDrawerEnabled(appConfig.shouldShowSidebarForUrl(
url));
  }

  public int urlLevelForUrl(String url) {
    ArrayList<Pattern> entries =
AppConfig.getInstance(this).navStructureLevelsRegex;
    if (entries != null) {
      for (int i = 0; i < entries.size(); i++) {
        Pattern regex = entries.get(i);
        if (regex.matcher(url).matches()) {
          return
AppConfig.getInstance(this).navStructureLevels.get(i);
        }
      }
    }

    // return unknown
    return -1;
  }

  public String titleForUrl(String url) {
    ArrayList<HashMap<String,Object>> entries =
AppConfig.getInstance(this).navTitles;
    String title = null;

    if (entries != null) {
      for (HashMap<String,Object> entry : entries) {
        Pattern regex = (Pattern)entry.get("regex");

        if (regex.matcher(url).matches()) {
          if (entry.containsKey("title")) {
            title = (String)entry.get("title");
          }
          if (title == null &&
entry.containsKey("urlRegex")) {
            Pattern urlRegex =
(Pattern)entry.get("urlRegex");
            Matcher match = urlRegex.matcher(url);
            if (match.find() && match.groupCount() >=
1) {
              String temp = match.group(1);
              // dashes to spaces, capitalize
              temp = temp.replace("-", " ");
              temp = LeanUtils.capitalizeWords(temp);

              title = temp;
            }

            // remove words from end of title
            if (title != null &&
entry.containsKey("urlChompWords") &&
                (Integer)entry.get("urlChompWords")
> 0) {
              int chompWords =
(Integer)entry.get("urlChompWords");
              String[] words = title.split("\\s+");
              StringBuilder sb = new StringBuilder();
              for (int i = 0; i < words.length -
chompWords - 1; i++){
                sb.append(words[i]);
                sb.append(" ");
              }
              if (words.length > chompWords) {
                sb.append(words[words.length -
chompWords - 1]);
              }
              title = sb.toString();
            }
          }

          break;
        }
      }
    }

    return title;
  }

  public void closeDrawers() {
    mDrawerLayout.closeDrawers();
  }

  public boolean isNotRoot() {
    return !isRoot;
```

```
    }

    public int getParentUrlLevel() {
        return parentUrlLevel;
    }

    public int getUrlLevel() {
        return urlLevel;
    }

    public void setUrlLevel(int urlLevel) {
        this.urlLevel = urlLevel;
    }

    public ProfilePicker getProfilePicker() {
        return profilePicker;
    }

    public FileDownloader getFileDownloader() {
        return fileDownloader;
    }

    public StatusCheckerBridge getStatusCheckerBridge() {
        return new StatusCheckerBridge();
    }

    @Override
    public void setTitle(CharSequence title) {
        super.setTitle(title);

        if (getSupportActionBar() != null) {
            getSupportActionBar().setTitle(title);
        }
    }

    public void startCheckingReadyStatus() {
        statusChecker.run();
    }

    private void stopCheckingReadyStatus() {
        handler.removeCallbacks(statusChecker);
    }

    private void checkReadyStatus() {
        this.mWebview.runJavascript("if
(gonative_status_checker && typeof
gonative_status_checker.onReadyState === 'function')
gonative_status_checker.onReadyState(document.ready
State);");
    }

    private void checkReadyStatusResult(String status) {
        // if interactiveDelay is specified, then look for
readyState=interactive, and show webview
        // with a delay. If not specified, wait for
readyState=complete.
        double interactiveDelay =
AppConfig.getInstance(this).interactiveDelay;

        if (status.equals("loading") ||
(Double.isNaN(interactiveDelay) &&
status.equals("interactive"))) {
            startedLoading = true;
        }
        else if ((!Double.isNaN(interactiveDelay) &&
status.equals("interactive"))
                || (startedLoading &&
status.equals("complete"))) {

            if (status.equals("interactive")) {
                showWebview(interactiveDelay);
            } else {
                showWebview();
            }
        }
    }

    public void showTabs() {
        this.slidingTabStrip.setVisibility(View.VISIBLE);
        ActionBar actionBar = this.getSupportActionBar();
        if (actionBar != null) {
            actionBar.setElevation(0);
        }
        ViewCompat.setElevation(this.slidingTabStrip,
ACTIONBAR_ELEVATION);
    }

    public void hideTabs() {
        this.slidingTabStrip.setVisibility(View.GONE);
        ActionBar actionBar = this.getSupportActionBar();
        if (actionBar != null) {
            actionBar.setElevation(ACTIONBAR_ELEVATION);
        }
    }

    public void toggleFullscreen(boolean fullscreen) {
        ActionBar actionBar = this.getSupportActionBar();
        View decorView = getWindow().getDecorView();
        int visibility = decorView.getSystemUiVisibility();
        int fullscreenFlags =
View.SYSTEM_UI_FLAG_LOW_PROFILE |
                View.SYSTEM_UI_FLAG_HIDE_NAVIGATION;

        if (Build.VERSION.SDK_INT >= 16) {
```

```
        fullscreenFlags |=
View.SYSTEM_UI_FLAG_FULLSCREEN |

View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION;
      }

      if (Build.VERSION.SDK_INT >= 19) {
        fullscreenFlags |=
View.SYSTEM_UI_FLAG_IMMERSIVE |
            View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY;
      }

      if (fullscreen) {
        visibility |= fullscreenFlags;
        if (actionBar != null) actionBar.hide();
      } else {
        visibility &= ~fullscreenFlags;
        if (actionBar != null &&
AppConfig.getInstance(this).showActionBar)
actionBar.show();
      }

      decorView.setSystemUiVisibility(visibility);

      // Full-screen is used for playing videos.
      // Allow sensor-based rotation when in full screen
(even overriding user rotation preference)
      if (fullscreen) {

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTA
TION_SENSOR);
      } else {
        setScreenOrientationPreference();
      }
    }

    @Override
    public void onRequestPermissionsResult(int
requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
      switch (requestCode) {
        case
REQUEST_PERMISSION_READ_EXTERNAL_STORAGE:
          if (grantResults.length > 0 && grantResults[0]
== PackageManager.PERMISSION_GRANTED) {
            if (this.directUploadImageUri == null) {
              cancelFileUpload();
              return;
            }

            if (mUploadMessage != null) {
mUploadMessage.onReceiveValue(this.directUploadIma
geUri);
              mUploadMessage = null;
            }
            if (uploadMessageLP != null) {
              uploadMessageLP.onReceiveValue(new
Uri[]{this.directUploadImageUri});
              uploadMessageLP = null;
            }

            this.directUploadImageUri = null;
          } else {
            cancelFileUpload();
          }
          break;
        case REQUEST_PERMISSION_GEOLOCATION:
          if (this.geolocationPermissionCallback != null) {
            if (grantResults.length >= 2 &&
                grantResults[0] ==
PackageManager.PERMISSION_GRANTED &&
                grantResults[1] ==
PackageManager.PERMISSION_GRANTED) {

this.geolocationPermissionCallback.onResult(true);
            } else {

this.geolocationPermissionCallback.onResult(false);
            }
            this.geolocationPermissionCallback = null;
          }
          break;
        case
REQUEST_PERMISSION_WRITE_EXTERNAL_STORAGE:
          if (grantResults.length > 0 && grantResults[0]
== PackageManager.PERMISSION_GRANTED) {

this.fileDownloader.gotExternalStoragePermissions(true)
;
          }
          break;
        case REQUEST_PERMISSION_GENERIC:
          Iterator<PermissionsCallbackPair> it =
pendingPermissionRequests.iterator();
          while (it.hasNext()) {
            PermissionsCallbackPair pair = it.next();
            if (pair.permissions.length !=
permissions.length) continue;
            boolean skip = false;
            for (int i = 0; i < pair.permissions.length && i
< permissions.length; i++) {
```

```
                if
(!pair.permissions[i].equals(permissions[i])) {
                    skip = true;
                    break;
                }
            }
            if (skip) continue;

            // matches PermissionsCallbackPair
            if (pair.callback != null) {

pair.callback.onPermissionResult(permissions,
grantResults);
            }
            it.remove();
        }

        if (pendingPermissionRequests.size() == 0 &&
pendingStartActivityAfterPermissions.size() > 0) {
            Iterator<Intent> i =
pendingStartActivityAfterPermissions.iterator();
            while (i.hasNext()) {
                Intent intent = i.next();
                startActivity(intent);
                i.remove();
            }
        }
        break;
    }
  }

  public void setUploadMessage(ValueCallback<Uri>
mUploadMessage) {
    this.mUploadMessage = mUploadMessage;
  }

  public void setUploadMessageLP(ValueCallback<Uri[]>
uploadMessageLP) {
    this.uploadMessageLP = uploadMessageLP;
  }

  public void setDirectUploadImageUri(Uri
directUploadImageUri) {
    this.directUploadImageUri = directUploadImageUri;
  }

  public RelativeLayout getFullScreenLayout() {
    return fullScreenLayout;
  }

  public class StatusCheckerBridge {
    @JavascriptInterface
    public void onReadyState(final String state) {
      runOnUiThread(new Runnable() {
        @Override
        public void run() {
          checkReadyStatusResult(state);
        }
      });
    }
  }

  private class ConnectivityChangeReceiver extends
BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent
intent) {
      retryFailedPage();
      if (connectivityCallback != null) {
        sendConnectivity(connectivityCallback);
      }
    }
  }

  public void getRuntimeGeolocationPermission(final
GeolocationPermissionCallback callback) {
    int checkFine =
ContextCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_FINE_LOCATION);
    int checkCoarse =
ContextCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_COARSE_LOCATIO
N);

    if (checkFine ==
PackageManager.PERMISSION_GRANTED &&
checkCoarse ==
PackageManager.PERMISSION_GRANTED) {
      callback.onResult(true);
    }

    if
(ActivityCompat.shouldShowRequestPermissionRational
e(this, Manifest.permission.ACCESS_FINE_LOCATION) ||

ActivityCompat.shouldShowRequestPermissionRationale
(this, Manifest.permission.ACCESS_COARSE_LOCATION))
{
      Toast.makeText(this,
R.string.request_permission_explanation_geolocation,
Toast.LENGTH_SHORT).show();
    }

    this.geolocationPermissionCallback = callback;
```

```
      ActivityCompat.requestPermissions(this, new
String[]{
            Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION
      }, REQUEST_PERMISSION_GEOLOCATION);
   }

   public void getExternalStorageWritePermission() {
      // check external storage permission
      if (ContextCompat.checkSelfPermission(this,

Manifest.permission.WRITE_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED) {

         if
(ActivityCompat.shouldShowRequestPermissionRational
e(this,

Manifest.permission.WRITE_EXTERNAL_STORAGE)) {
            Toast.makeText(this,
R.string.request_permission_explanation_storage,
Toast.LENGTH_LONG).show();
         }

         ActivityCompat.requestPermissions(this,
               new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAG
E},

REQUEST_PERMISSION_WRITE_EXTERNAL_STORAGE);
      } else {

this.fileDownloader.gotExternalStoragePermissions(true)
;
      }
   }

   public void getPermission(String[] permissions,
PermissionCallback callback) {
      boolean needToRequest = false;
      for (String permission : permissions) {
         if (ContextCompat.checkSelfPermission(this,
permission) != PackageManager.PERMISSION_GRANTED)
{
            needToRequest = true;
            break;
         }
      }

      if (needToRequest) {
         if (callback != null) {
            pendingPermissionRequests.add(new
PermissionsCallbackPair(permissions, callback));
         }

         ActivityCompat.requestPermissions(this,
permissions, REQUEST_PERMISSION_GENERIC);
      } else {
         // send all granted result
         if (callback != null) {
            int[] results = new int[permissions.length];
            for (int i = 0; i < results.length; i++) {
               results[i] =
PackageManager.PERMISSION_GRANTED;
            }
            callback.onPermissionResult(permissions,
results);
         }
      }
   }

   public void startActivityAfterPermissions(Intent intent)
{
      if (pendingPermissionRequests.size() == 0) {
         startActivity(intent);
      } else {
         pendingStartActivityAfterPermissions.add(intent);
      }
   }

   private void setScreenOrientationPreference() {
      AppConfig appConfig = AppConfig.getInstance(this);

      switch (appConfig.forceScreenOrientation) {
         case UNSPECIFIED:

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTA
TION_UNSPECIFIED);
            break;
         case PORTRAIT:

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTA
TION_PORTRAIT);
            break;
         case LANDSCAPE:

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTA
TION_SENSOR_LANDSCAPE);
            break;
      }
   }

   public TabManager getTabManager() {
```

```
    return tabManager;
  }

  public interface PermissionCallback {
    void onPermissionResult(String[] permissions, int[]
grantResults);
  }

  private class PermissionsCallbackPair {
    String[] permissions;
    PermissionCallback callback;

    PermissionsCallbackPair(String[] permissions,
PermissionCallback callback) {
      this.permissions = permissions;
      this.callback = callback;
    }
  }

  public void enableSwipeRefresh() {
    if (this.swipeRefresh != null) {
      this.swipeRefresh.setEnabled(true);
    }
  }

  public void restoreSwipRefreshDefault() {
    if (this.swipeRefresh != null) {
      AppConfig appConfig =
AppConfig.getInstance(this);

this.swipeRefresh.setEnabled(appConfig.pullToRefresh);
    }
  }

  public void deselectTabs() {
    this.slidingTabStrip.deselect();
  }

  private void listenForSignalStrength() {
    if (this.phoneStateListener != null) return;

    this.phoneStateListener = new PhoneStateListener()
{
      @Override
      public void
onSignalStrengthsChanged(SignalStrength
signalStrength) {
        latestSignalStrength = signalStrength;
        sendConnectivityOnce();
        if (connectivityCallback != null) {
          sendConnectivity(connectivityCallback);
        }
      }
    };

    try {
      TelephonyManager telephonyManager =
(TelephonyManager)this.getSystemService(Context.TELE
PHONY_SERVICE);
      if (telephonyManager == null) {
        Log.e(TAG, "Error getting system telephony
manager");
      } else {

telephonyManager.listen(this.phoneStateListener,
PhoneStateListener.LISTEN_SIGNAL_STRENGTHS);
      }
    } catch (Exception e) {
      Log.e(TAG, "Error listening for signal strength", e);
    }

  }

  public void sendConnectivityOnce(String callback) {
    if (callback == null) return;

    this.connectivityOnceCallback = callback;
    if (this.phoneStateListener != null) {
      sendConnectivity(callback);
    } else {
      listenForSignalStrength();
      new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
          sendConnectivityOnce();
        }
      }, 500);
    }
  }

  private void sendConnectivityOnce() {
    if (this.connectivityOnceCallback == null) return;
    sendConnectivity(this.connectivityOnceCallback);
    this.connectivityOnceCallback = null;
  }

  private void sendConnectivity(String callback) {
    NetworkInfo activeNetwork =
cm.getActiveNetworkInfo();
    boolean connected = activeNetwork != null &&
activeNetwork.isConnected();
    String typeString;
    if (activeNetwork != null) {
      typeString = activeNetwork.getTypeName();
```

```
      } else {
        typeString = "DISCONNECTED";
      }

      try {
        JSONObject data = new JSONObject();
        data.put("connected", connected);
        data.put("type", typeString);

        if (this.latestSignalStrength != null) {
          JSONObject signalStrength = new JSONObject();

          signalStrength.put("cdmaDbm",
latestSignalStrength.getCdmaDbm());
          signalStrength.put("cdmaEcio",
latestSignalStrength.getCdmaEcio());
          signalStrength.put("evdoDbm",
latestSignalStrength.getEvdoDbm());
          signalStrength.put("evdoEcio",
latestSignalStrength.getEvdoEcio());
          signalStrength.put("evdoSnr",
latestSignalStrength.getEvdoSnr());
          signalStrength.put("gsmBitErrorRate",
latestSignalStrength.getGsmBitErrorRate());
          signalStrength.put("gsmSignalStrength",
latestSignalStrength.getGsmSignalStrength());
          if (Build.VERSION.SDK_INT >= 23) {
            signalStrength.put("level",
latestSignalStrength.getLevel());
          }
          data.put("cellSignalStrength", signalStrength);
        }

        String js = LeanUtils.createJsForCallback(callback,
data);
        runJavascript(js);
      } catch (JSONException e) {
        Log.e(TAG, "JSON error sending connectivity", e);
      }
    }

    public void subscribeConnectivity(final String callback)
{
      this.connectivityCallback = callback;
      listenForSignalStrength();
      new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
          sendConnectivity(callback);
        }
      }, 500);
    }

    public void unsubscribeConnectivity() {
      this.connectivityCallback = null;
    }

    public interface GeolocationPermissionCallback {
      void onResult(boolean granted);
    }
  }
```