

# PERANGKAT LUNAK LOGIN OTOMATIS UNTUK *Captive Portal* WI-FI

YOHANES MARIO CHANDRA—2011730031

## 1 Data Skripsi

Pembimbing utama/tunggal: **Pascal Alfadian**

Pembimbing pendamping: -

Kode Topik : **PAS4105**

Topik ini sudah dikerjakan selama : **2 semester**

Pengambilan pertama kali topik ini pada : Semester **41 - Ganjil 16/17**

Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**

Tipe Laporan : **C -** Dokumen pendukung untuk **pengambilan ke-2 di Skripsi 2**

## 2 Detail Perkembangan Pengerjaan Skripsi

Detail bagian pekerjaan skripsi sesuai dengan rencan kerja/laporan perkembangan terakhir :

1. Melakukan studi literatur mengenai hal-hal yang berkaitan dengan perancangan dan pembuatan aplikasi.

**status** : Ada sejak rencana kerja skripsi.

**hasil** : Sudah dilakukan studi literatur. *Captive Portal*: *router* atau *gateway* yang akan menutup koneksi eksternal sampai klien yang bersangkutan sudah terotentikasi. Protokol HTTP mengatur kode status yang dapat digunakan untuk mengindikasikan adanya *captive portal*, yaitu kode status 511. *.NET Framework*: platform yang bersifat *general purpose* untuk membangun aplikasi. *.NET Framework* memanfaatkan *Common Language Infrastructure* untuk memberikan kebebasan kepada *developer* untuk memilih bahasa yang ingin digunakan selama bahasa tersebut didukung oleh *.NET Framework*. *Universal Windows Platform* (UWP): arsitektur aplikasi umum untuk Windows. Memungkinkan aplikasi dapat dijalankan pada desktop dan Windows Phone. Kelas *WebBrowser* tidak jadi dipelajari, karena UWP tidak mendukung *WebBrowser*. Oleh karena itu, dipelajari kelas *WebView*. *WebView* memungkinkan *developer* untuk menampung konten HTML pada suatu aplikasi. *WebView* tidak mendukung masukkan pengguna seperti *key-down*, *key-up*, dan *pointer-pressed*. Oleh karena itu, dibutuhkan metode lain yang melibatkan *InvokeScriptAsync* dengan fungsi *eval javascript* untuk menggunakan HTML event handler dan fungsi *window.external.notify* untuk menangani event dari HTML pada aplikasi. *PasswordVault*: kelas yang merepresentasikan pengunci kredensial. Kredensial yang disimpan menggunakan kelas ini hanya dapat diakses oleh aplikasi atau service yang bersangkutan.

2. Melakukan analisis perangkat lunak sejenis.

**status** : Ada sejak rencana kerja skripsi.

**hasil** : Perangkat lunak sejenis pada Windows belum dapat ditemukan pada saat penelitian ini dilakukan. Oleh karena itu, perangkat lunak atau aplikasi sejenis yang dianalisis adalah aplikasi yang diciptakan untuk sistem operasi Android. Aplikasi tersebut bernama *WiFi Web Login*. Langkah-langkah yang harus ditempuh untuk melakukan *login* wifi berbasis web pada aplikasi ini adalah:

- (a) Deteksi sambungan dengan wifi yang bersangkutan.
- (b) Deteksi hubungan dengan internet.

- (c) Jika tidak terjadi hubungan dengan internet, deteksi apakah tersimpan informasi login untuk wifi yang bersangkutan.
- (d) Jika terdapat informasi login untuk wifi yang bersangkutan maka lakukan login otomatis.
- (e) Jika tidak terdapat informasi login untuk wifi yang bersangkutan maka rekam informasi login dan lakukan login.

Setelah pengguna melalui sudah pernah melakukan login pertama kali menggunakan aplikasi tersebut, maka aplikasi akan melakukan login otomatis setiap kali terhubung dengan wifi yang bersangkutan.

### 3. Melakukan analisis kebutuhan untuk mengimplementasikan mekanisme login otomatis ini

**status :** Ada sejak rencana kerja skripsi.

**hasil :** Ada beberapa metode yang diperlukan untuk mengimplementasikan mekanisme login otomatis:

- Metode Penyimpanan Informasi Login

Penyimpanan informasi login dapat dilakukan dengan beberapa metode, diantaranya adalah dengan menggunakan file teks atau menggunakan PasswordVault. Penyimpanan informasi menggunakan file teks berarti informasi disimpan dalam bentuk *plaintext* dalam file yang diberikan *access permission* tertentu. Sementara itu, penyimpanan informasi menggunakan PasswordVault memanfaatkan kelas API yang terdapat pada *Universal Windows Platform* (UWP).

Informasi yang perlu disimpan untuk dapat melakukan login otomatis adalah *connection fingerprint* (seperti SSID WiFi, url, dan potongan unik dokumen html), username, password, dan langkah-langkah login seperti menekan tombol. Oleh karena itu, metode penyimpanan menggunakan credential locker dan file teks perlu dianalisis untuk dapat ditentukan metode mana yang paling cocok untuk digunakan dalam penelitian ini.

PasswordVault dapat menyimpan informasi yang berisi *resource* (biasanya berupa nama aplikasi atau string unik lainnya), username, dan password. Informasi yang perlu disimpan selain username dan password adalah *connection fingerprint* dan langkah-langkah login. *Connection fingerprint* dapat disimpan pada resource karena sifatnya yang unik. Sementara itu, langkah-langkah login dapat disimpan pada username atau password karena langkah-langkah login dapat disimpan dalam bentuk String. Akan tetapi, langkah-langkah login sebaiknya disimpan pada password, dipisahkan dengan karakter pemisah tertentu, agar username dapat digunakan sebagai *identifier* unik. PasswordVault memiliki batasan 10 kredensial yang dapat disimpan per aplikasi. Jika aplikasi mencoba menyimpan lebih dari 10 kredensial maka akan terjadi exception. Oleh karena hal ini, PasswordVault menjadi pilihan yang kurang baik untuk kebutuhan perangkat lunak pada penelitian ini.

Metode penyimpanan lainnya adalah dengan menggunakan file teks. Penyimpanan informasi menggunakan file teks dapat dilakukan untuk informasi berbasis teks apapun dan tidak ada batasan banyaknya informasi yang dapat disimpan (kecuali batasan perangkat keras seperti kapasitas hard disk). Akan tetapi, file teks dapat dibaca oleh aplikasi manapun, sehingga penyimpanan informasi sensitif tidak dapat dilakukan tanpa adanya metode pengamanan tertentu. Salah satu metode pengamanan yang dapat dilakukan adalah dengan mendeklarasikan *file access permission*. Akan tetapi, karena Windows memiliki *security model* per pengguna dan bukan per aplikasi, maka aplikasi lain yang dijalankan oleh pengguna tersebut memiliki akses yang sama kepada file yang bersangkutan. Metode pengamanan lainnya adalah dengan melakukan enkripsi pada file yang bersangkutan sehingga hanya pemegang kunci yang dapat membaca file tersebut. Enkripsi file pada windows dapat dilakukan menggunakan kelas *CryptographicEngine*. Kunci enkripsi dan dekripsi dapat disimpan menggunakan PasswordVault atau dengan meminta pengguna untuk memasukkan kunci tersebut setiap kali aplikasi dijalankan.

Metode penyimpanan yang digunakan pada penelitian ini adalah metode penyimpanan menggunakan file teks. Akan tetapi, seperti yang sudah dijabarkan sebelumnya, diperlukan metode pengamanan untuk file teks tersebut. Metode pengamanan yang digunakan adalah enkripsi file teks yang bersangkutan. Kunci enkripsi dan dekripsi dibangun secara random saat aplikasi pertama kali dijalankan dan disimpan menggunakan PasswordVault.

- Metode Rekam dan Kirim Informasi Login

Kelas WebView pada *Universal Windows Platform* (UWP) hanya dapat dihubungkan dengan kode C# menggunakan javascript. *Method* yang digunakan untuk melakukan eksekusi javascript pada WebView adalah `InvokeScriptAsync`. Metode ini memiliki parameter string berupa nama fungsi javascript yang ingin dipanggil dan array of string yang berisi argumen yang ingin dikirimkan ke dalam fungsi tersebut. Salah satu fungsi yang dapat dipanggil adalah *eval*. Dengan menggunakan *eval*, ekspresi javascript apapun dapat dijalankan pada WebView. Untuk mengirimkan data dari javascript ke kode C#, dapat dijalankan fungsi `window.external.notify` dengan parameter berupa string. Oleh karena itu, diperlukan *encoding* tertentu (seperti JSON<sup>1</sup>) untuk memasukkan lebih dari satu argumen.

`InvokeScriptAsync` dapat digunakan untuk memanggil fungsi *eval* dengan parameter berupa function yang dapat digunakan untuk menekan tombol atau memasukkan nilai pada *text field* tertentu. Selain itu, dapat dimasukkan event listener yang dapat memanggil `window.external.notify` menggunakan cara ini. Fungsi `window.external.notify` dapat membantu mengirimkan event-event seperti mouse click, keypress, atau perubahan nilai pada *text field* yang ada pada halaman HTML pada WebView tersebut.

- Metode Deteksi *Captive Portal*

Kode status HTTP 511 digunakan untuk memberitahu klien bahwa respon yang didapat bukan berasal dari server tujuan dan diperlukan otentikasi jaringan. Akan tetapi, pada prakteknya, tidak semua *captive portal* melakukan implementasi kode status HTTP 511.

*Captive portal* milik Unpar mengirimkan kode status 302, dan bukan 511. Oleh karena adanya perbedaan implementasi seperti ini, deteksi *captive portal* menggunakan kode status HTTP tidak dapat dilakukan.

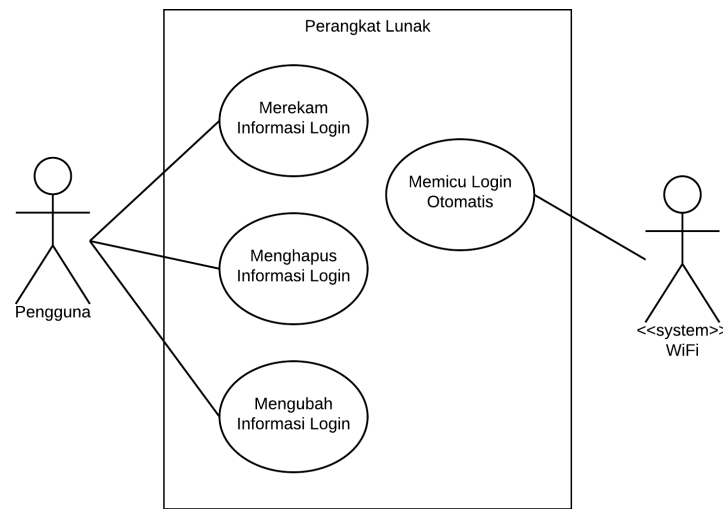
Persamaan implementasi yang dimiliki oleh setiap *captive portal* adalah dibutuhkannya *redirection* yang dapat dikenali oleh *web browser* agar klien selalu diarahkan ke halaman *captive portal* tersebut sebelum melakukan otentikasi. Oleh karena itu, untuk setiap HTTP *request* yang dilakukan oleh klien sebelum melakukan otentikasi, HTTP *response* yang diberikan bukanlah berasal dari server tujuan. Sifat ini dapat dimanfaatkan untuk keperluan deteksi *captive portal* dengan mengirimkan *request* ke server yang sudah ditentukan sebelumnya, dan mendeteksi apakah *response* yang diberikan sesuai dengan harapan. Jika *response* yang diberikan tidak sesuai dengan harapan, maka dapat diasumsikan bahwa klien dibatasi oleh suatu *captive portal*.

Kelas WebView pada *Universal Windows Platform* (UWP) memiliki kemampuan deteksi *redirection* dan *response* yang tidak sesuai harapan seperti *web browser* pada umumnya. Oleh karena itu, kelas WebView digunakan untuk melakukan deteksi *captive portal* pada penelitian ini tanpa perlu memeriksa kode status HTTP setiap *response*. Penggunaan kelas WebView juga akan mempermudah proses otomatisasi login karena WebView dapat melakukan hal-hal yang dapat dilakukan oleh *web browser* pada umumnya seperti menjalankan javascript dan menampilkan halaman HTML.

---

<sup>1</sup><https://tools.ietf.org/html/rfc7159>

Diagram *Use Case*:



Gambar 1: Diagram *use case* untuk perangkat lunak yang dibangun.

#### **Skenario merekam informasi login**

Nama: Merekam informasi login

Aktor: Pengguna

Kondisi awal: Perangkat lunak mendeteksi adanya *captive portal*.

Deskripsi: Pengguna menyimpan informasi login.

Kondisi Akhir: Informasi login tersimpan di dalam perangkat lunak.

Skenario:

- (a) Pengguna memasukkan informasi login ke dalam form HTML.
- (b) Sistem menyimpan informasi login yang dimasukkan oleh pengguna.

#### **Skenario menghapus informasi login**

Nama: Menghapus informasi login

Aktor: Pengguna

Kondisi awal: Perangkat lunak sudah dijalankan.

Deskripsi: Pengguna menghapus informasi login.

Kondisi Akhir: Informasi login dihapus dari perangkat lunak.

Skenario:

- (a) Pengguna memilih untuk menghapus informasi login.
- (b) Sistem menghapus informasi login.

#### **Skenario mengubah informasi login**

Nama: Mengubah informasi login

Aktor: Pengguna

Kondisi awal: Perangkat lunak sudah dijalankan.

Deskripsi: Pengguna mengubah informasi login.

Kondisi Akhir: Informasi login diubah dari perangkat lunak.

Skenario:

- (a) Pengguna memilih untuk mengubah informasi login.

- (b) Sistem menampilkan form ubah informasi login.
- (c) Pengguna memasukkan informasi login baru.
- (d) Sistem menghapus informasi login lama dan menyimpan informasi login baru.

### Skenario memicu login otomatis

Nama: Memicu login otomatis

Aktor: WiFi

Kondisi awal: Perangkat lunak sudah dijalankan.

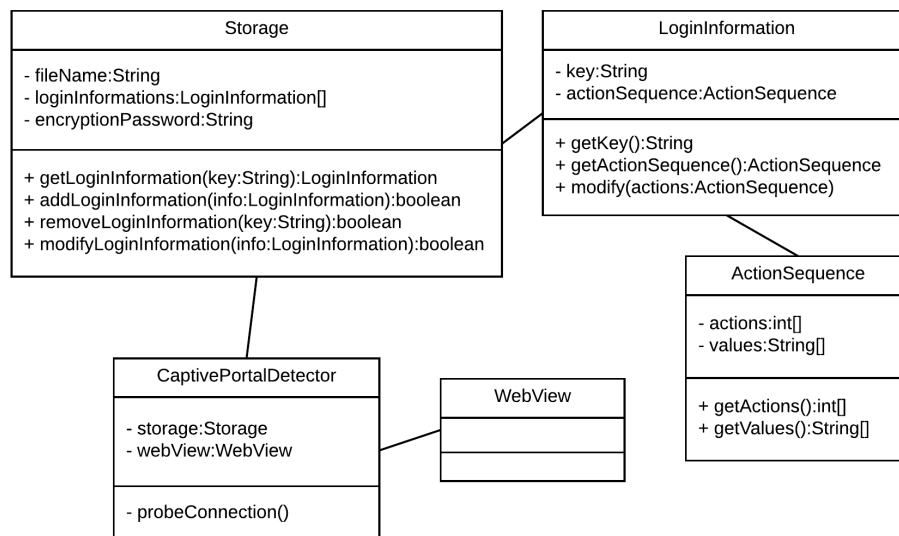
Deskripsi: Sistem memicu login otomatis berdasarkan perubahan status WiFi.

Kondisi Akhir: Klien terotentikasi pada *captive portal*.

Skenario:

- (a) Sistem mendeteksi adanya koneksi WiFi yang terjadi.
- (b) Sistem mendeteksi adanya *captive portal*.
- (c) Sistem mendeteksi adanya informasi login untuk *captive portal* tersebut.
- (d) Sistem mengirimkan informasi login kepada *captive portal*.
- (e) Sistem mendeteksi koneksi dengan internet dan klien sudah terotentikasi pada *captive portal* tersebut.

Diagram Kelas:



Gambar 2: Diagram kelas untuk perangkat lunak yang dibangun.

Kelas-kelas yang dibutuhkan pada perangkat lunak ini adalah:

- **CaptivePortalDetector**

Kelas ini digunakan untuk mendeteksi keberadaan *captive portal*. Jika *captive portal* terdeteksi, maka informasi login yang tersimpan dalam **Storage** digunakan. Jika tidak terdapat informasi login dalam **Storage**, maka direkam informasi login baru.

- **Storage**

Kelas ini digunakan untuk menyimpan seluruh informasi login dalam bentuk file teks yang sudah terenkripsi.

- **LoginInformation**

Kelas ini digunakan untuk menyimpan informasi login dalam bentuk key atau fingerprint, serta *ActionSequence*

- **ActionSequence**

Kelas ini digunakan untuk menyimpan langkah-langkah login dalam bentuk urutan aksi dan nilai-nilai yang berkaitan dengan aksi tersebut.

#### 4. Merancang perangkat lunak login otomatis ini

**status :** Ada sejak rencana kerja skripsi.

**hasil :** Perancangan yang dilakukan mencakupi perancangan kelas, diagram *sequence*, serta penjelasan mengenai hasil analisis yang tidak mungkin diimplementasikan dan cara lain yang dilakukan untuk mendapatkan hasil yang serupa.

- **Perancangan Kelas**

Gambar 3 menjelaskan mengenai kelas-kelas dalam perangkat lunak yang dibuat. Beberapa kelas utama yang perlu dijelaskan antara lain:

**MainPage :** Kelas ini merupakan kelas yang berperan sebagai tampilan utama aplikasi. Atribut-atribut yang dimiliki oleh kelas ini adalah:

- **cpd**  
Atribut untuk menyimpan *instance* *CaptivePortalDetector*.
- **timeoutTimer**  
Atribut untuk menyimpan timer yang digunakan untuk menghitung *connection timeout*.
- **loaded**  
Atribut untuk menyimpan status *loading* suatu halaman.

Metode-metode yang dimiliki oleh kelas ini adalah:

- **setup**  
Metode ini digunakan untuk melakukan *setup* awal saat aplikasi dieksekusi. Fungsinya adalah untuk menyimpan *instance* *CaptivePortalDetector* dan memanggil metode *setup()* pada *instance* tersebut.
- **MainWebView\_LoadCompleted**  
Metode ini dipanggil saat *WebView* selesai melakukan *loading* halaman. Fungsinya adalah untuk memanggil metode *onLoad()* pada *CaptivePortalDetector*.
- **MainWebView\_NavigationStarting**  
Metode ini dipanggil saat *WebView* baru akan memulai navigasi ke halaman baru. Fungsinya adalah untuk memulai *timer* untuk *timeout* dan memasukkan objek *ScriptNotifyHandler*.
- **MainWebView\_NewWindowRequested**  
Metode ini dipanggil saat *WebView* melakukan *request* untuk membuka window baru. Fungsinya adalah untuk memanggil metode *queueUri()* pada *CaptivePortalDetector*.
- **MainWebView\_NavigationCompleted**  
Metode ini dipanggil saat *WebView* selesai melakukan navigasi ke halaman baru, namun belum selesai melakukan *loading* halaman tersebut. Fungsinya adalah untuk melakukan *override* fungsi-fungsi JavaScript seperti *window.open()* dan *open()* agar bisa diakses dari JavaScript tanpa aksi langsung dari pengguna.

**NetChangeDetectorBackgroundTask :** Kelas ini merupakan kelas yang digunakan untuk melakukan deteksi perubahan jaringan yang nantinya digunakan untuk menampilkan notifikasi apabila terdeteksi adanya jaringan yang terhubung tanpa adanya internet. Atribut yang dimiliki oleh kelas ini adalah:

- lastSSID

Atribut ini menyimpan SSID terakhir yang nantinya akan dibandingkan dengan SSID terbaru untuk mendeteksi adanya perubahan SSID.

Metode-metode yang dimiliki oleh kelas ini adalah:

- Run

Metode ini dipanggil saat Windows mengalami perubahan jaringan. Fungsinya adalah untuk menampilkan notifikasi apabila kondisi `connectionChanged() && lastSSID!=null && hasNoInternetAccess()` terpenuhi.

- hasNoInternetAccess

Metode ini digunakan untuk mendeteksi tidak adanya akses internet menggunakan API yang diberikan oleh UWP.

- connectionChanged

Metode ini digunakan untuk mendeteksi perubahan SSID.

**ScriptNotifyHandler** : Kelas ini merupakan kelas yang digunakan untuk menghubungkan sisi javascript pada WebView dengan kode C#. Metode-metode yang dimiliki oleh kelas ini adalah:

- passAction

Metode ini dipanggil saat *listener* yang sudah disisipkan ke dalam WebView mendeteksi *action* yang dapat direkam. *Action* yang direkam berupa teks yang berisi kode javascript yang dapat mereplikasi *action* tersebut.

- windowOpen

Metode ini dipanggil saat javascript pada WebView memanggil fungsi `window.open` atau fungsi `open`.

**CaptivePortalDetector** : Kelas ini merupakan kelas utama yang berfungsi untuk melakukan deteksi *captive portal*, menyisipkan kode *listener* pada WebView, merekam *action sequence* yang dilakukan oleh pengguna, dan menjalankan kembali *action sequence* yang sudah tersimpan. Kelas ini menggunakan *design pattern* singleton agar kelas-kelas lainnya bisa mengakses *instance* yang sama pada setiap *session*. Atribut yang dimiliki oleh kelas ini adalah:

- instance

Atribut ini menyimpan *instance* CaptivePortalDetector.

- storage

Atribut ini menyimpan objek Storage yang digunakan untuk menyimpan dan mengakses informasi login.

- ssid

Atribut ini menyimpan SSID saat ini.

- uriQueue

Atribut ini menyimpan queue Uri yang perlu diakses.

- webView

Atribut ini menyimpan WebView yang digunakan untuk melakukan deteksi *captive portal*.

- textBlock

Atribut ini menyimpan TextBlock yang digunakan untuk menampilkan pesan kepada pengguna.

- comboBox

Atribut ini menyimpan ComboBox yang digunakan untuk menampilkan daftar SSID yang sudah tersimpan kepada pengguna.

- currentFingerprint

Atribut ini menyimpan fingerprint saat ini.

- `currentActionSequence`  
Atribut ini menyimpan `ActionSequence` yang terkait dengan fingerprint saat ini.
- `timer`  
Atribut ini menyimpan timer yang digunakan untuk mengatur waktu akses Uri dalam `uriQueue`.

Metode-metode yang dimiliki oleh kelas ini adalah:

- `getInstance`  
Metode ini digunakan untuk mendapatkan *instance* dari `CaptivePortalDetector`.
- `setup`  
Metode ini digunakan untuk melakukan *setup* awal yang menyimpan `WebView`, `TextBlock`, dan `ComboBox` ke dalam *instance* `CaptivePortalDetector`.
- `refreshList`  
Metode ini digunakan untuk melakukan *refresh* tampilan `ComboBox`.
- `updateWebView`  
Metode ini digunakan untuk menentukan melakukan deteksi *captive portal* jika terhubung dengan koneksi WiFi, atau menampilkan pesan kepada pengguna jika tidak terhubung dengan koneksi WiFi.
- `onLoad`  
Metode ini dipanggil saat `WebView` sudah selesai melakukan *loading* halaman. Fungsinya adalah untuk melakukan `deployListener()`, menjalankan aksi-aksi yang sudah terekam pada informasi login, dan mendeteksi sudah atau belum terjadinya koneksi dengan internet.
- `navigationStarting`  
Metode ini dipanggil saat `WebView` mulai melakukan navigasi ke halaman baru. Fungsinya adalah untuk membatalkan timer untuk membuka halaman-halaman popup dari halaman sebelumnya.
- `passAction`  
Metode ini digunakan untuk menyimpan *action* yang dikirimkan oleh `ScriptNotifyHandler` ke dalam `currentActionSequence`.
- `updateSSID`  
Metode ini digunakan untuk mendapatkan SSID terbaru.
- `queueUri`  
Metode ini digunakan untuk memasukkan Uri baru ke dalam `uriQueue`.
- `timeout`  
Metode ini digunakan untuk menyatakan bahwa terjadi *connection timeout*.
- `removeLoginInformation`  
Metode ini digunakan untuk menghapus seluruh informasi login yang terkait dengan SSID tertentu.

**Storage** : Kelas ini digunakan untuk menyimpan informasi login dan password yang digunakan untuk melakukan enkripsi. Atribut yang dimiliki oleh kelas ini adalah:

- `fileName`  
Atribut ini menyimpan nama file yang digunakan untuk menyimpan informasi login yang terenkripsi.
- `password`  
Atribut ini menyimpan password yang digunakan untuk melakukan enkripsi.



- loginInfo

Atribut ini menyimpan objek LoginInformation yang digunakan untuk menyimpan seluruh informasi login.

Metode-metode yang dimiliki oleh kelas ini adalah:

- setup

Metode ini digunakan untuk melakukan *setup* awal seperti membuka file dan melakukan dekripsi.

- getLoginInfo

Metode ini digunakan untuk mendapatkan objek LoginInformation.

- saveData

Metode ini digunakan untuk menyimpan data yang ada pada objek LoginInformation ke dalam file dan melakukan enkripsi pada file tersebut.

**LoginInformation** : Kelas ini digunakan untuk merepresentasikan informasi login. Atribut yang dimiliki oleh kelas ini adalah:

- actionSequences

Atribut ini merupakan pasangan *key-value* antara suatu fingerprint dengan ActionSequence.

Metode-metode yang dimiliki oleh kelas ini adalah:

- getActionSequence

Metode ini digunakan untuk mendapatkan ActionSequence berdasarkan fingerprint tertentu.

- addActionSequence

Metode ini digunakan untuk menambahkan ActionSequence untuk fingerprint tertentu.

- removeBySSID

Metode ini digunakan untuk menghapus ActionSequence milik fingerprint tertentu.

- getList

Metode ini digunakan untuk mendapatkan daftar SSID yang sudah direkam.

**ActionSequence** : Kelas ini digunakan untuk merepresentasikan urutan *action*. Atribut yang dimiliki oleh kelas ini adalah:

- actions

Atribut ini merupakan daftar *action* yang berupa kode javascript.

Metode-metode yang dimiliki oleh kelas ini adalah:

- add

Metode ini digunakan untuk menambahkan *action* ke dalam daftar ini.

- getEnumerable

Metode ini digunakan untuk mendapatkan enumerable dari daftar *actions*, sehingga mempermudah eksekusi *actions*.

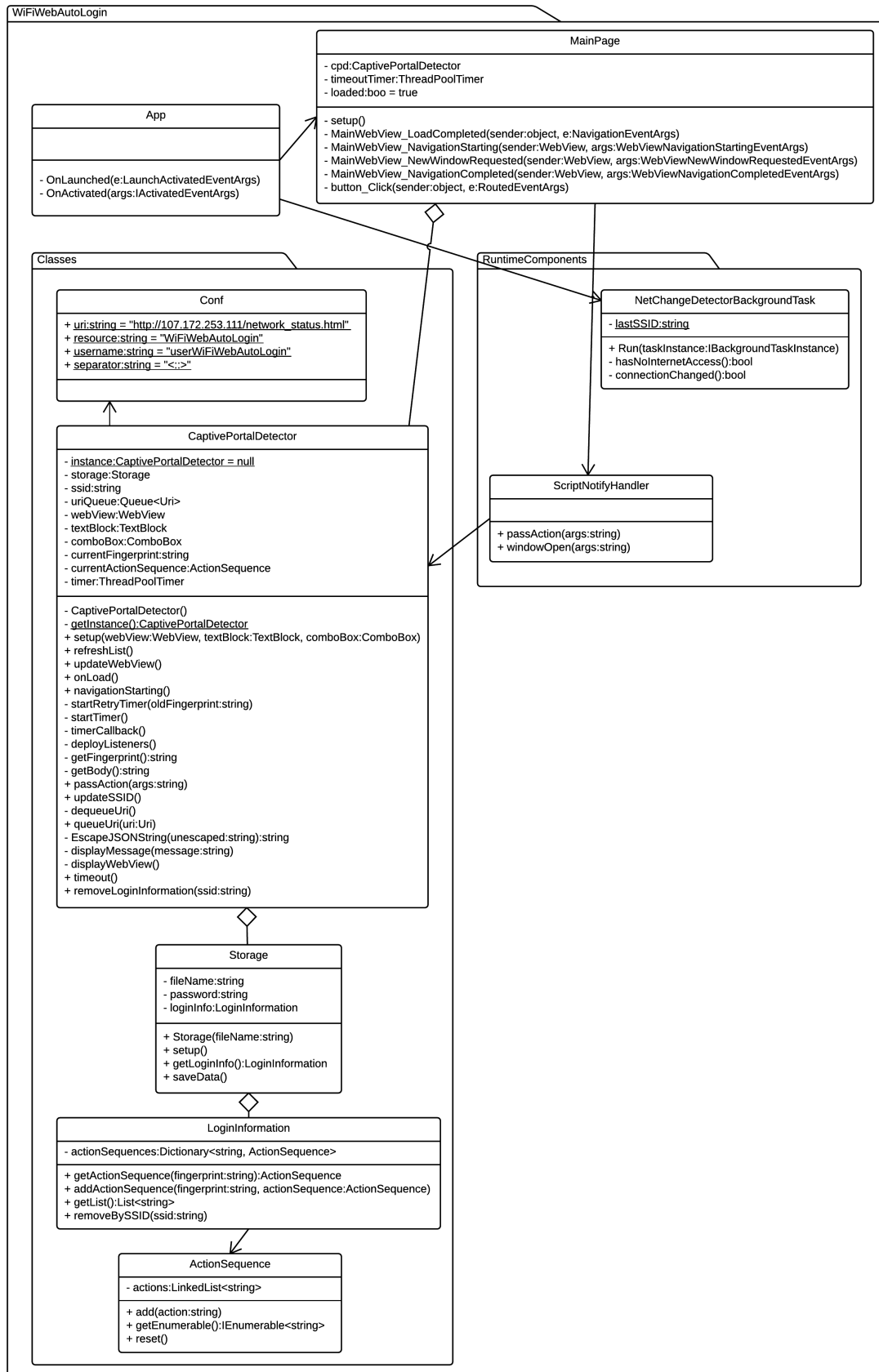
- reset

Metode ini digunakan untuk menghapus seluruh *action* yang ada pada daftar ini.

## • Perancangan Algoritma dan Struktur Data

- **Algoritma Deteksi *Captive Portal***

Algoritma yang digunakan untuk melakukan deteksi *captive portal* adalah sebagai berikut:



Gambar 3: Diagram Kelas Rinci.

```

if Network Detected and No Internet Connection then
    Ask user to run application via notification;
    if "Yes" button clicked then
        Access a web page which can only be opened if there is an internet connection;
        if Redirected then
            Captive portal detected;
        end
    end
end

```

Algoritma di atas menjelaskan deteksi *captive portal* dilakukan dengan melakukan deteksi jaringan yang tidak terhubung dengan internet. Jika ditemukan jaringan yang tidak terhubung dengan internet, maka akan muncul notifikasi yang memungkinkan pengguna untuk menjalankan perangkat lunak. Saat perangkat lunak dijalankan, perangkat lunak akan mencoba untuk mengakses halaman panicingan yang beralamatkan pada:

`http://107.172.253.111/network_status.html`

Jika didapat respon HTTP yang berupa *redirect*, maka pada jaringan tersebut terdapat *captive portal*. Algoritma ini akan didaftarkan pada sistem saat perangkat lunak pertama kali dijalankan dan dipanggil saat ada perubahan status jaringan.

#### – Struktur Data dan Format *Fingerprint*

*Fingerprint* suatu halaman terdiri dari SSID, uri, serta isi *tag* head pada halaman tersebut. Data ini disimpan dalam satu buah string dan dipisahkan oleh separator "<::>" (tanpa tanda petik). Salah satu contoh *fingerprint* adalah:

UNPAR9<::>https://cas.unpar.ac.id/login<::><title>Halaman Login</title>

yang memiliki arti bahwa *fingerprint* tersebut berasal dari WiFi dengan:

- \* SSID UNPAR9,
- \* uri `https://cas.unpar.ac.id/login`,
- \* serta isi *tag* head `<title>Halaman Login</title>`.

#### – Struktur Data *LoginInformation*

Kelas *LoginInformation* memiliki daftar objek bertipe *ActionSequence* yang disimpan pada properti *actionSequences* bertipe *Dictionary*. Kelas *Dictionary* dapat menyimpan data yang berupa pasangan *key-value*. *Key* yang digunakan bertipe string dan merupakan *fingerprint* suatu halaman. *Value* yang disimpan adalah objek bertipe *ActionSequence* yang merupakan list aksi-aksi yang perlu dilakukan untuk halaman tersebut.

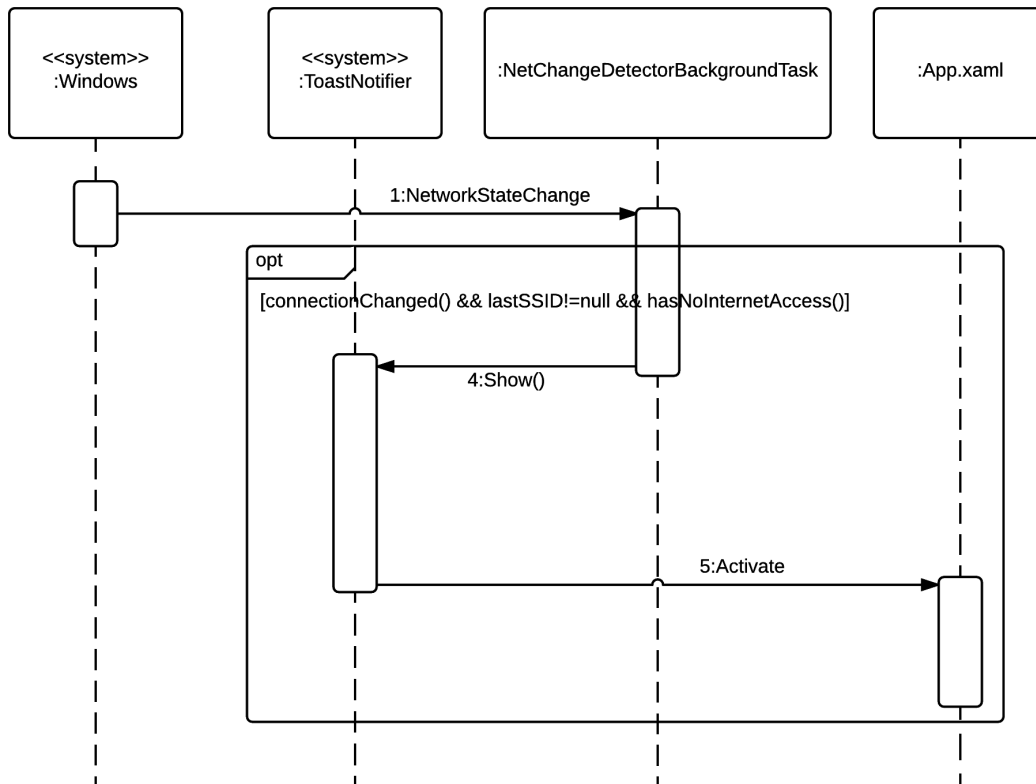
Kelas *ActionSequence* memiliki properti *actions* yang bertipe *LinkedList<string>*. Setiap elemen *LinkedList* tersebut menyimpan string yang merupakan kode JavaScript yang akan dieksekusi pada halaman yang bersangkutan. *Username* dan *password* juga tersimpan di dalam kode JavaScript tersebut. Salah satu contoh string yang disimpan dalam properti *actions* adalah `document.getElementsByTagName("input")[0].value = "username"`; yang berarti ubah isi elemen input pertama dengan "username".

### • Perancangan Interaksi Perangkat Lunak

#### – Perancangan Interaksi Deteksi Jaringan

Gambar 4 menjelaskan mengenai interaksi antar objek dalam perangkat lunak untuk melakukan deteksi perubahan jaringan. Interaksi yang terjadi adalah sebagai berikut:

- (a) Saat komputer mengalami perubahan jaringan (tidak terhubung menjadi terhubung dan sebaliknya, atau terjadi perubahan *cost* jaringan), *trigger* *NetworkStateChange* akan di-



Gambar 4: Diagram Interaksi Deteksi Jaringan.

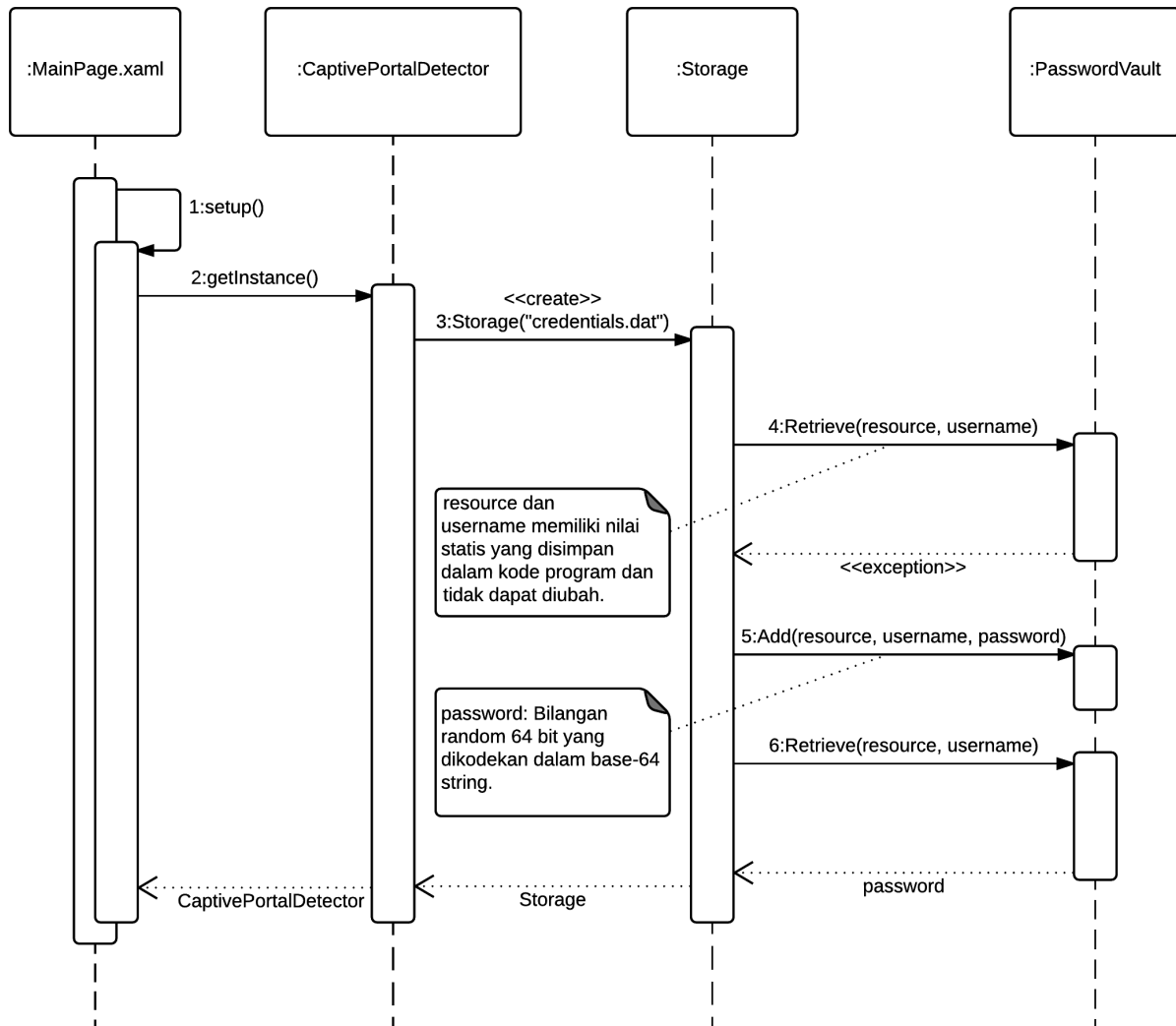
aktifkan oleh Windows, dan objek NetChangeDetectorBackgroundTask yang sudah di-daftarkan akan menerima *trigger* tersebut.

- (b) Jika kondisi `connectionChanged() && lastSSID!=null && hasNoInternetAccess()` terpenuhi, maka:
  - i. NetChangeDetectorBackgroundTask memerintahkan ToastNotifier untuk memunculkan notifikasi menggunakan method `Show()`.
  - ii. Saat user menekan tombol "Yes" pada notifikasi, App.xaml diaktivasi.

#### – Perancangan Interaksi Penciptaan Password

Gambar 5 menjelaskan mengenai interaksi antar objek dalam perangkat lunak untuk menciptakan password random saat perangkat lunak pertama kali dijalankan. Interaksi yang terjadi adalah sebagai berikut:

- (a) MainPage.xaml melakukan `setup()`.
- (b) MainPage.xaml memanggil metode `getInstance()` pada CaptivePortalDetector untuk mendapatkan *instance* CaptivePortalDetector.
- (c) CaptivePortalDetector menciptakan objek Storage baru pada *constructor*-nya.
- (d) Objek Storage berusaha untuk mendapatkan password dengan memanggil metode `Retrieve()` pada objek PasswordVault, namun mendapatkan exception karena belum ada password yang disimpan.
- (e) Objek Storage memasukkan password baru yang diciptakan secara random menggunakan metode `Add()` pada PasswordVault.
- (f) Objek Storage memanggil metode `Retrieve()` kembali pada objek PasswordVault, dan mendapatkan password yang baru saja diciptakan. Setelah itu, CaptivePortalDetector



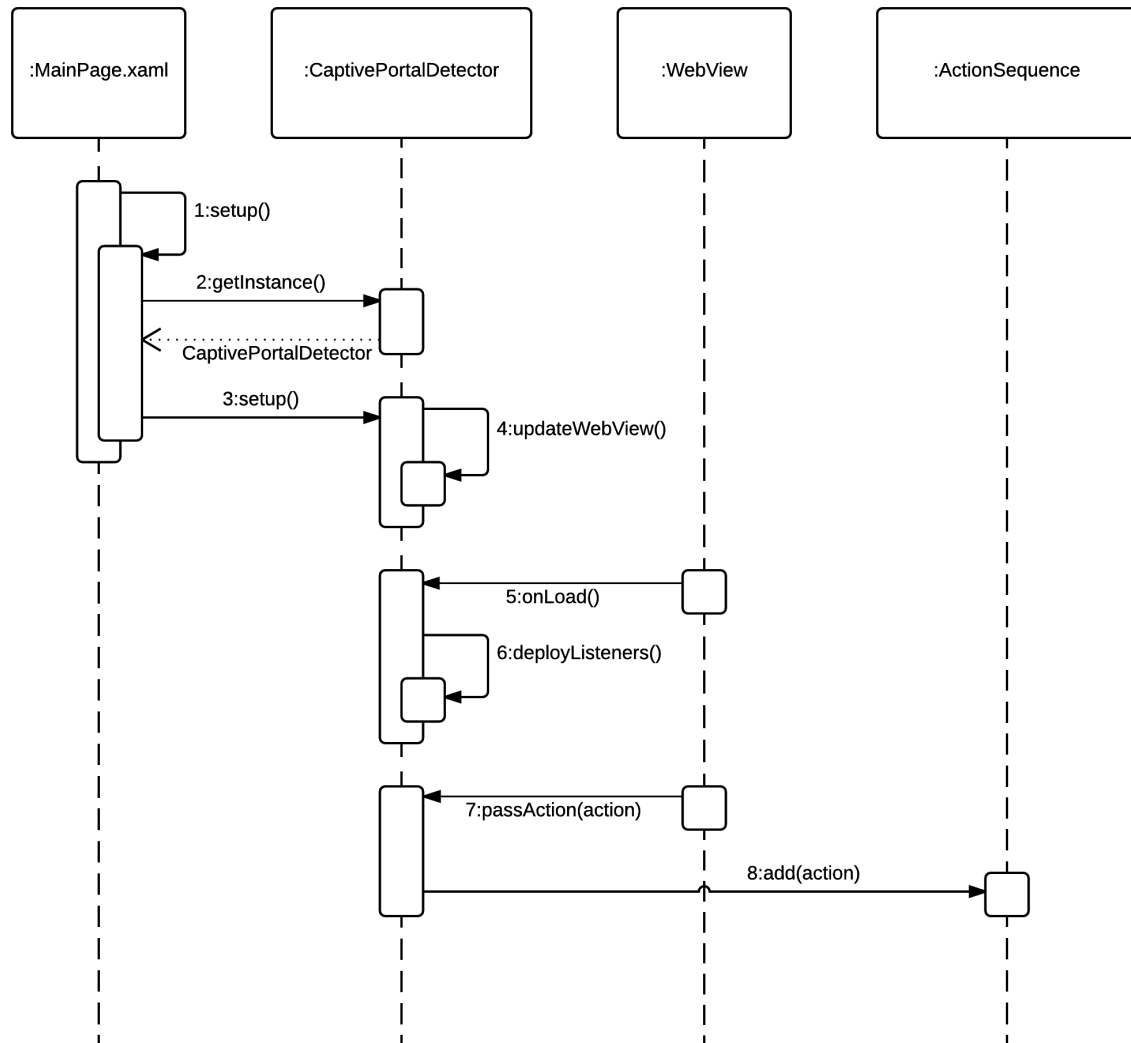
Gambar 5: Diagram Interaksi Penciptaan Password.

mendapatkan objek **Storage**, dan **MainPage.xaml** mendapatkan *instance* **CaptivePortalDetector**.

#### – Perancangan Interaksi Penyimpanan Informasi Login

Gambar 6 menjelaskan mengenai interaksi antar objek dalam perangkat lunak untuk menyimpan informasi login. Interaksi yang terjadi adalah sebagai berikut:

- MainPage.xaml** melakukan **setup()**.
- MainPage.xaml** memanggil metode **getInstance()** pada kelas **CaptivePortalDetector** untuk mendapatkan *instance* **CaptivePortalDetector**. **MainPage.xaml** mendapatkan *instance* **CaptivePortalDetector**.
- MainPage.xaml** memanggil metode **setup()** pada objek **CaptivePortalDetector**.
- CaptivePortalDetector** melakukan **updateWebView()** untuk mengarahkan **WebView** ke **URI** yang digunakan untuk melakukan deteksi koneksi internet.
- WebView** memanggil metode **onLoad()** pada objek **CaptivePortalDetector** saat halaman selesai dimuat.
- Jika halaman tidak berisi teks "connected" (tanpa tanda petik), **CaptivePortalDetector** melakukan **deployListeners()** untuk menangkap semua *event* yang mungkin dilakukan oleh



Gambar 6: Diagram Interaksi Penyimpanan Informasi Login.

pengguna pada halaman tersebut.

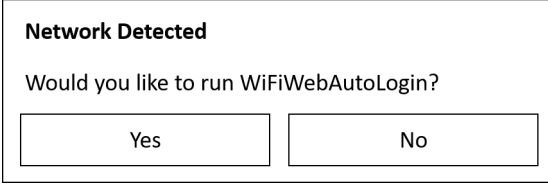
- (g) Metode `passAction()` dipanggil pada objek `CaptivePortalDetector` saat pengguna melakukan klik atau input teks untuk mengirimkan aksi yang baru saja dilakukan oleh pengguna.
- (h) `CaptivePortalDetector` memanggil metode `add()` pada objek `ActionSequence` untuk menyimpan aksi tersebut.

#### • Perancangan Antarmuka

Pengguna memerlukan antarmuka untuk berinteraksi dengan perangkat lunak. Antarmuka yang diperlukan adalah:

##### – Antarmuka Notifikasi

Antarmuka notifikasi muncul setiap kali terhubung dengan WiFi yang menggunakan *captive portal*. Gambar 7 menampilkan desain antarmuka notifikasi. Desain antarmuka notifikasi menggunakan desain notifikasi standar windows dengan dua tombol, "Yes" dan "No". Jika tombol "Yes" ditekan, maka notifikasi akan hilang dan aplikasi akan dijalankan. Jika tombol "No" ditekan, maka notifikasi akan hilang. Antarmuka notifikasi adalah antarmuka yang pertama kali akan muncul dalam siklus aplikasi karena kelas `NetChangeDetectorBackgroundTask`



**Network Detected**


Would you like to run WiFiWebAutoLogin?

Yes No

Gambar 7: Rancangan Antarmuka Notifikasi.

didaftarkan pada sistem untuk mendeteksi perubahan jaringan.

– **Antarmuka Message Box**



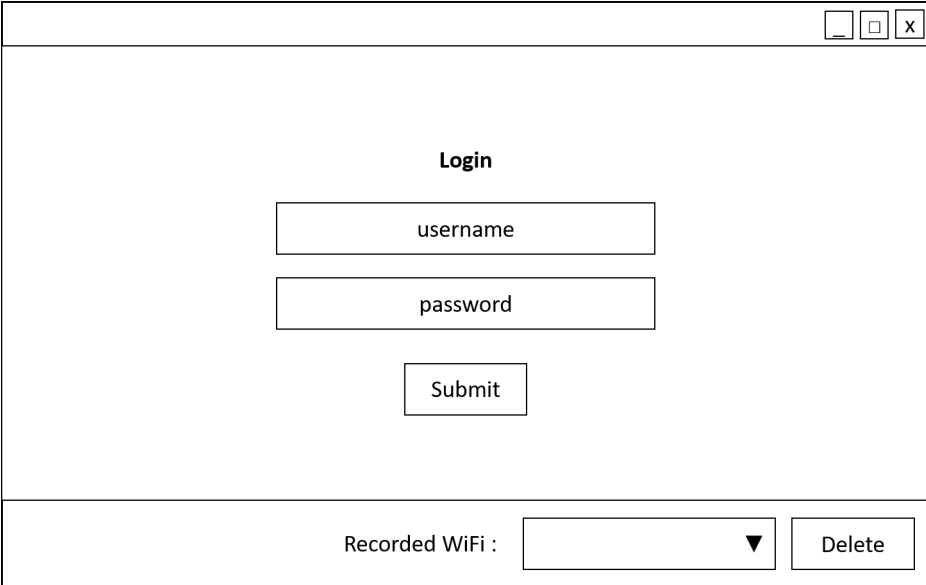
<<message>>

Recorded WiFi :  ▼

Gambar 8: Rancangan Antarmuka Message Box.

Gambar 8 menampilkan desain antarmuka message box. Selain label untuk menaruh pesan, antarmuka ini juga memiliki *selector* untuk dapat menghapus WiFi yang sudah terekam. Dengan menghapus WiFi yang terdapat pada *selector* ini, pengguna dapat merekam ulang informasi login pada WiFi tersebut.

– **Antarmuka Web Browser**



**Login**

username

password

Submit

Recorded WiFi :  ▼

Gambar 9: Rancangan Antarmuka Web Browser.

Gambar 9 menampilkan desain antarmuka web browser. Antarmuka ini digunakan untuk menampilkan halaman web yang berkaitan dengan login *captive portal*. Pada antarmuka inilah aksi pengguna akan direkam secara otomatis. Selain itu, pada antarmuka ini terdapat komponen yang sama dengan antarmuka message box, yaitu komponen *selector* untuk menghapus

SSID WiFi yang sudah terekam.

5. Melakukan pengujian terhadap perangkat lunak untuk menghasilkan perbaikan terhadap perangkat lunak tersebut.

**status :** Ada sejak rencana kerja skripsi.

**hasil :** Setelah dilakukan implementasi pertama, ditemukan beberapa masalah yang membuat rancangan perangkat lunak tidak dapat sepenuhnya didasarkan pada hasil analisis, di antaranya:

- Fungsi `window.external.notify` tidak berperilaku sebagaimana yang diperkirakan. Fungsi ini diharapkan dapat dipanggil secara langsung di dalam kode javascript, namun ternyata tidak bisa. Setiap halaman yang ingin memanfaatkan fungsi ini harus didaftarkan pada `Package.appxmanifest`. Metode yang digunakan untuk mendapatkan hasil yang sama dengan fungsi yang diberikan oleh `window.external.notify` adalah dengan menggunakan kelas bertipe `RuntimeComponent` yang diizinkan untuk dapat diakses oleh JavaScript pada `WebView`. Kelas ini adalah `ScriptNotifyHandler` pada gambar 3.
- Fungsi `window.open` dan fungsi `open` tidak dapat dijalankan secara otomatis sehingga popup tidak muncul. Metode yang digunakan untuk mendapatkan hasil yang sama dari yang direncanakan sebelumnya adalah dengan melakukan *override* fungsi `window.open` dan fungsi `open` pada saat halaman selesai dimuat, dan mengubungkannya dengan kelas `ScriptNotifyHandler`. Akan tetapi, metode ini masih kurang memadai karena kode JavaScript yang memanggil fungsi-fungsi tersebut pada saat halaman dimuat akan dieksekusi sebelum *override* terjadi.

Perancangan yang sudah dibuat merupakan hasil revisi dari pengujian ini.

6. Melakukan pengujian (dan eksperimen) yang melibatkan responden untuk menilai hasil simulasi secara kualitatif

**status :** Ada sejak rencana kerja skripsi. Namun responden tidak dibutuhkan untuk keperluan pengujian.

**hasil :** Pengujian dibagi menjadi 2 bagian, yaitu pengujian fungsional dan pengujian eksperimental.

- **Pengujian Fungsional**

Pengujian fungsional dilakukan pada jaringan WiFi di kost di jalan Ciumbuleuit nomor 149, Bandung, dengan SSID "C149Net".

- **Rencana Pengujian Fungsional**

Pengujian fungsional dilakukan menggunakan teknik *black box*. Pengujian fungsional dilakukan untuk memastikan fungsi-fungsi utama dalam perangkat lunak sudah berjalan dengan baik. Fungsi-fungsi yang akan diuji mencakupi:

- \* Deteksi perubahan jaringan.
- \* Deteksi *captive portal*.
- \* Login otomatis.

Setiap fungsi yang diuji diberikan kasus pengujian positif dan pengujian negatif.

- **Hasil Pengujian Fungsional**

Hasil-hasil pengujian fungsional adalah sebagai berikut:

- \* **Pengujian deteksi perubahan jaringan**

- **Pengujian positif**

**Kasus:** Menghubungkan komputer dengan WiFi yang terhubung dengan *captive portal*.

**Hasil yang diharapkan:** Muncul notifikasi.

**Hasil yang didapatkan:** Muncul notifikasi "Network Detected" dengan pesan "Would



you like to run WiFiWebAutoLogin?".

**Kesimpulan:** Fungsi berjalan sesuai harapan.

- **Pengujian negatif**

**Kasus:** Menghubungkan komputer dengan WiFi yang tidak terhubung dengan *captive portal*.

**Hasil yang diharapkan:** Tidak muncul notifikasi apapun.

**Hasil yang didapatkan:** Tidak muncul notifikasi apapun.

**Kesimpulan:** Fungsi berjalan sesuai harapan.

- \* **Pengujian deteksi *captive portal***

- **Pengujian positif**

**Kasus:** Menghubungkan komputer dengan WiFi yang terhubung dengan *captive portal* dan menekan tombol "Yes" pada notifikasi.

**Hasil yang diharapkan:** Muncul halaman login.

**Hasil yang didapatkan:** Muncul halaman login *captive portal*.

**Kesimpulan:** Fungsi berjalan sesuai harapan.

- **Pengujian negatif**

**Kasus:** Menghubungkan komputer dengan WiFi yang tidak terhubung dengan *captive portal* maupun internet dan menekan tombol "Yes" pada notifikasi

**Hasil yang diharapkan:** Muncul pesan *timeout*.

**Hasil yang didapatkan:** Muncul pesan "Operation timeout. Check your network connection."

**Kesimpulan:** Fungsi berjalan sesuai harapan.

- \* **Pengujian login otomatis**

- **Pengujian positif**

**Kasus:** Menghubungkan komputer dengan WiFi yang terhubung dengan *captive portal* yang sudah pernah dijalankan login secara manual.

**Hasil yang diharapkan:** Muncul pesan "Connected."

**Hasil yang didapatkan:** Muncul pesan "Executing recorded actions...", lalu setelah beberapa saat, muncul pesan "Connected."

**Kesimpulan:** Fungsi berjalan sesuai harapan.

- **Pengujian negatif**

**Kasus:** Menghubungkan komputer dengan WiFi yang terhubung dengan *captive portal* yang belum pernah dijalankan login secara manual.

**Hasil yang diharapkan:** Muncul halaman login.

**Hasil yang didapatkan:** Muncul halaman login *captive portal*.

**Kesimpulan:** Fungsi berjalan sesuai harapan.

- **Pengujian Eksperimental**

Pengujian eksperimental dilakukan untuk memeriksa apakah perangkat lunak dapat berjalan pada beragam *captive portal*.

- **Rencana Pengujian Eksperimental**

Pengujian eksperimental dilakukan pada *captive portal* pada jaringan WiFi dengan SSID:

- \* *C149Net* pada kost di jalan Ciumbuleuit nomor 149, Bandung.
- \* *Starbucks@wifi.id* pada Starbucks Dipatiukur, Bandung.
- \* *wifi.id* pada Starbucks Dipatiukur, Bandung.
- \* *UNPAR9* pada gedung 10 Universitas Katolik Parahyangan, Bandung.
- \* *FTIS.cisco* pada gedung 9 Universitas Katolik Parahyangan, Bandung.

## – Hasil Pengujian Eksperimental

Hasil pengujian eksperimental akan dijelaskan untuk setiap SSID yang diuji. Penjelasan berupa narasi hasil yang didapatkan berdasarkan langkah-langkah yang sama dengan pengujian fungsional *black box*.

### \* Hasil Pengujian WiFi C149Net

Pengujian pada WiFi dengan SSID C149Net yang berlokasi pada kost di jalan Ciumbuleuit nomor 149, Bandung, mendapatkan hasil sesuai harapan. Notifikasi muncul pada saat WiFi pertama kali terhubung. Halaman login muncul setelah tombol "Yes" pada notifikasi ditekan. Setelah memasukkan *username* dan *password*, pesan "Connected." muncul. Jika informasi login sudah tersimpan, pesan "Connected." akan langsung muncul setelah menekan tombol "Yes" pada notifikasi.

### \* Hasil Pengujian WiFi Starbucks@wifi.id

Pengujian pada WiFi dengan SSID Starbucks@wifi.id yang berlokasi pada Starbucks Dipatiukur, Bandung, mendapatkan hasil sesuai harapan. Notifikasi muncul pada saat WiFi pertama kali terhubung. Halaman *captive portal* muncul setelah tombol "Yes" pada notifikasi ditekan. Setelah menekan tombol "continue" pada halaman tersebut, lalu menekan tombol "lanjutkan" pada halaman selanjutnya, pesan "Connected." muncul. Jika informasi login sudah tersimpan, pesan "Connected." akan langsung muncul setelah menekan tombol "Yes" pada notifikasi.

### \* Hasil Pengujian WiFi wifi.id

Pengujian pada WiFi dengan SSID wifi.id yang berlokasi pada Starbucks Dipatiukur, Bandung, mendapatkan hasil yang tidak sesuai harapan. Notifikasi muncul pada saat WiFi pertama kali terhubung. Halaman login muncul setelah tombol "Yes" pada notifikasi ditekan. Akan tetapi, login tidak dapat dilakukan karena perlu membeli voucher setiap kali ingin melakukan login.

### \* Hasil Pengujian WiFi UNPAR9

Pengujian pada WiFi dengan SSID UNPAR9 yang berlokasi pada gedung 10 Universitas Katolik Parahyangan, Bandung, mendapatkan hasil yang tidak sesuai harapan. Notifikasi muncul pada saat WiFi pertama kali terhubung. Halaman login muncul setelah tombol "Yes" pada notifikasi ditekan. Setelah login dilakukan, proses terhenti pada halaman <https://portal.unpar.ac.id/home>. Hal ini dikarenakan WiFi di Universitas Katolik Parahyangan menggunakan CAS<sup>2</sup>. Selain itu, CAS ini menggunakan *pop-up* untuk menampilkan halaman <https://wireless.unpar.ac.id/status> dan halaman tersebut adalah halaman yang membuka popup yang menuju ke halaman tujuan. WebView pada UWP tidak memperbolehkan pemanggilan fungsi `open()`, `window.open()`, `el.click()` dan `form.submit()` yang bukan merupakan aksi langsung oleh pengguna. *Override* tiap fungsi tersebut dimungkinkan setelah halaman selesai dimuat, namun itu berarti fungsi hasil *override* tidak akan digunakan jika fungsi tersebut dipanggil pada saat halaman pertama kali dimuat. Hal ini mencakup *onload event* dan script yang dipanggil langsung pada *script tag* baik pada *body* maupun *head*.

### \* Hasil Pengujian WiFi FTIS.cisco

Pengujian pada WiFi dengan SSID FTIS.cisco yang berlokasi pada gedung 9 Universitas Katolik Parahyangan, Bandung, mendapatkan hasil yang tidak sesuai harapan. Hal ini terjadi karena WiFi ini merupakan jaringan internal Fakultas Teknologi dan Sains yang menggunakan jaringan WiFi Universitas Katolik Parahyangan untuk akses internet. Proses terhenti pada halaman yang sama dengan WiFi UNPAR9, yaitu pada halam-

---

<sup>2</sup> Central Authorization Service

an <https://portal.unpar.ac.id/home>. Penyebab terjadinya hal ini juga sama dengan yang terjadi pada WiFi UNPAR9.

7. Membuat kesimpulan dari hasil penelitian dan saran untuk penelitian selanjutnya.

**status** : Ada sejak rencana kerja skripsi.

**hasil** : Kesimpulan dan saran yang dihasilkan dari penelitian ini adalah:

- **Kesimpulan**

- Implementasi perangkat lunak untuk melakukan login otomatis pada *captive portal* berhasil dilakukan walaupun memiliki keterbatasan tidak dapat melakukan login otomatis jika *captive portal* bergantung pada pop-up untuk mengakses halaman tujuan.
- *Username* dan *password* sudah disimpan secara aman dalam file yang dienkripsi menggunakan kunci yang diciptakan secara random per aplikasi.
- SSID, uri, dan konten *tag* head adalah informasi yang dibutuhkan untuk membedakan antar halaman pada setiap *captive portal*.

- **Saran**

Saran dari peneliti yang dapat dilakukan untuk mengembangkan penelitian ini adalah gunakan *platform* lain selain UWP, seperti Windows Form, Java, Android, atau iOS. Hal ini dapat dilakukan untuk menghindari keterbatasan WebView pada UWP dalam menangani pop-up. Jika ingin tetap menggunakan UWP, maka penerus penelitian ini harus menciptakan WebView sendiri. Hal ini dapat dilakukan menggunakan Canvas yang sudah disediakan oleh UWP dan menggabungkannya dengan teknologi-teknologi yang sudah ada seperti webkit atau gecko.

8. Menulis dokumen skripsi.

**status** : Ada sejak rencana kerja skripsi.

**hasil** : Sudah dikerjakan sampai dengan bab 6 (Kesimpulan dan Saran)<sup>3</sup>.

---

<sup>3</sup><https://github.com/yohanesmario/Skripsi/raw/master/doc/DokumenSkripsi/main.pdf>

### 3 Pencapaian Rencana Kerja<sup>4</sup>

Persentase penyelesaian skripsi sampai dengan dokumen ini dibuat dapat dilihat pada tabel berikut :

1*	2*(%)	3*(%)	4*(%)	5*	6*(%)
1	5	5			5
2	10	10			10
3	15	10	5	kaji ulang singkat kebutuhan pada S2	15
4	10		10		10
5	15		15		15
6	5		5		5
7	10		10		10
8	30	15	15	sebagian bab 1 dan 2, serta bagian awal analisis di S1	30
Total	100	40	60		100

Keterangan (\*)

1 : Bagian pengerjaan Skripsi (nomor disesuaikan dengan detail pengerjaan di bagian 5)

2 : Persentase total

3 : Persentase yang akan diselesaikan di Skripsi 1

4 : Persentase yang akan diselesaikan di Skripsi 2

5 : Penjelasan singkat apa yang dilakukan di S1 (Skripsi 1) atau S2 (skripsi 2)

6 : Persentase yang sudah diselesaikan sampai saat ini

Bandung, 30/04/2017

Yohanes Mario Chandra

Menyetujui,

Nama: Pascal Alfadian

Pembimbing Tunggal

---

<sup>4</sup>Terdapat kesalahan pada dokumen rencana kerja terdahulu di mana terdapat 8 poin rencana kerja, sementara hanya tertera 7 poin pada tabel rencana kerja. Poin-poin rencana kerja yang benar adalah yang ada pada dokumen ini.