

SKRIPSI

PERANGKAT LUNAK LOGIN OTOMATIS
UNTUK *CAPTIVE PORTAL* WIFI



YOHANES MARIO CHANDRA

NPM: 2011730031

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2018

UNDERGRADUATE THESIS

**AUTOMATED LOGIN SOFTWARE
FOR WIFI CAPTIVE PORTAL**



YOHANES MARIO CHANDRA

NPM: 2011730031

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2018**

LEMBAR PENGESAHAN

PERANGKAT LUNAK LOGIN OTOMATIS
UNTUK *CAPTIVE PORTAL* WIFI

YOHANES MARIO CHANDRA

NPM: 2011730031

Bandung, 3 Januari 2018

Menyetujui,

Pembimbing Tunggal

Pascal Alfadian, M.Comp.

Ketua Tim Penguji

Anggota Tim Penguji

Mariskha Tri Adithia, P.D.Eng

Chandra Wijaya, M.T.

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PERANGKAT LUNAK LOGIN OTOMATIS UNTUK *CAPTIVE PORTAL* WIFI

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 3 Januari 2018

Meterai

Yohanes Mario Chandra
NPM: 2011730031

ABSTRAK

Captive portal yang banyak digunakan pada WiFi publik membutuhkan login berulang-ulang setiap kali pengguna ingin menggunakan WiFi tersebut. Oleh karena itu, dibutuhkan otomasi untuk mempermudah penggunaan WiFi dengan *captive portal*. Perangkat lunak diciptakan untuk melakukan otomasi tersebut menggunakan UWP (*Universal Windows Platform*) dengan bahasa pemrograman C#. Perangkat lunak ini harus memiliki kemampuan untuk melakukan deteksi captive portal, rekam dan kirim informasi login, serta melakukan login otomatis jika sudah ada informasi login yang tersimpan untuk *captive portal* tersebut. Implementasi berhasil dilakukan, hanya saja ditemukan adanya keterbatasan yang diakibatkan oleh keterbatasan *platform* yaitu tidak dapat melakukan login otomatis pada *captive portal* yang membutuhkan popup. Informasi login tersimpan secara aman pada sebuah file yang dienkripsi menggunakan password *random* yang diciptakan saat perangkat lunak pertama kali dijalankan. Perangkat lunak dapat melakukan identifikasi *captive portal* menggunakan SSID, URI, serta konten tag *head* pada halaman tersebut.

Kata-kata kunci: *captive portal*, login, otomasi

ABSTRACT

Captive portals which are widely used on public WiFi require repeated login every time a user wants to use the WiFi. Therefore, it requires automation to facilitate the use of WiFi with captive portal. The software was created to do the automation using UWP (Universal Windows Platform) with C# programming language. This software must have capability to detect captive portal, record and send login information, and login automatically if there is already login information stored for captive portal. Implementation is successful, it's just found the limitations caused by the limitations of the platform which can not perform automatic login on captive portals which requires a popup. The login information is stored securely on an encrypted file using a random password generated when the software was first run. The software can identify captive portals using SSID, URI, and tag head content on the page.

Keywords: captive portal, login, otomation

Untuk Mama, Papa, Gaby, Angel, dan Nadia yang kucintai.

KATA PENGANTAR

Puji dan syukur saya ucapkan untuk Tuhan atas terselesaikannya skripsi ini. Berkat rahmatNya, skripsi ini dapat terselesaikan guna memenuhi tugas akhir program studi Teknik Informatika Universitas Katolik Parahyangan.

Banyaknya penggunaan *captive portal* dan masalah-masalah yang ditimbulkan oleh karenanya menjadi motivasi bagi peneliti untuk menyusun skripsi ini. Harapannya ialah agar kesulitan-kesulitan yang dialami oleh peneliti selama menggunakan *captive portal* dapat teratasi dan tidak perlu dialami oleh orang lain dengan terselesaikannya skripsi ini.

Dalam pengerjaan skripsi ini tentunya ada beberapa pihak yang secara khusus ingin saya berikan ucapan terima kasih, yaitu:

- Teresa Nadia Susanto yang selalu menemani dan memberi semangat dalam mengerjakan skripsi.
- Ilona Tjahjono sebagai teman yang membantu simulasi presentasi review skripsi 1 dan memberi beberapa masukan untuk perbaikan presentasi.
- Dosen pembimbing yang sudah membimbing dengan sabar meskipun begitu banyak kekurangan.
- Dosen-dosen penguji yang sudah memberikan masukan-masukan pada saat sidang yang sangat berguna.
- Semua pihak yang sudah membantu terselesaikannya skripsi ini tapi tidak dapat disebutkan satu per satu.

Bandung, Januari 2018

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan Penelitian	2
1.4 Batasan Masalah	2
1.5 Metodologi Penelitian	2
1.6 Sistematika Pembahasan	3
2 DASAR TEORI	5
2.1 <i>Captive Portal</i>	5
2.2 <i>.NET Framework</i>	6
2.3 <i>Common Language Infrastructure</i> (CLI)	7
2.4 <i>Universal Windows Platform</i> (UWP)	8
2.5 Kelas WebView Pada UWP	8
2.6 <i>Interface IBackgroundTask</i> Pada UWP	9
2.7 Kelas PasswordVault Pada Windows	9
3 ANALISIS	11
3.1 Analisis Metode Penyimpanan Informasi Login	11
3.2 Analisis Metode Rekam dan Kirim Informasi Login	12
3.3 Analisis Metode Deteksi <i>Captive Portal</i>	12
3.4 Analisis Perangkat Lunak	13
3.4.1 Analisis Kelas	14
3.4.2 Analisis <i>Use Case</i>	15
3.4.3 Analisis Alir	17
3.5 Analisis Perangkat Lunak Sejenis	18
4 PERANCANGAN	21
4.1 Perancangan Kelas	21
4.2 Perancangan Algoritma dan Struktur Data	26
4.2.1 Algoritma Deteksi <i>Captive Portal</i>	27
4.2.2 Struktur Data dan Format <i>Fingerprint</i>	27
4.2.3 Struktur Data LoginInformation	28
4.3 Perancangan Interaksi Perangkat Lunak	28
4.3.1 Perancangan Interaksi Deteksi Jaringan	28
4.3.2 Perancangan Interaksi Penciptaan Password	29
4.3.3 Perancangan Interaksi Penyimpanan Informasi Login	30

4.4	Perancangan Antarmuka	31
4.4.1	Antarmuka Notifikasi	31
4.4.2	Antarmuka Aplikasi	32
5	IMPLEMENTASI DAN PENGUJIAN	35
5.1	Lingkungan Implementasi dan Pengujian	35
5.2	Masalah Implementasi dan Solusinya	35
5.3	Pengujian Fungsional	36
5.3.1	Rencana Pengujian Fungsional	36
5.3.2	Hasil Pengujian Fungsional	36
5.4	Pengujian Eksperimental	37
5.4.1	Rencana Pengujian Eksperimental	37
5.4.2	Hasil Pengujian Eksperimental	38
6	KESIMPULAN DAN SARAN	41
6.1	Kesimpulan	41
6.2	Saran	41
	DAFTAR REFERENSI	43
A	KODE SUMBER	45
A.1	Namespace: WiFiWebAutoLogin	45
A.2	Namespace: WiFiWebAutoLogin.Classes	56
A.3	Namespace: WiFiWebAutoLogin.RuntimeComponents	69

DAFTAR GAMBAR

2.1	Diagram alir proses yang dilalui oleh kode program dalam CLI.	8
3.1	<i>Screenshot</i> HTTP <i>header</i> yang dikirimkan oleh <i>captive portal</i> milik Unpar.	13
3.2	Diagram kelas untuk perangkat lunak yang dibangun.	14
3.3	Diagram <i>use case</i> untuk perangkat lunak yang dibangun.	16
3.4	Diagram alir untuk perangkat lunak yang dibangun.	17
3.5	Tampilan aplikasi <i>WiFi Web Login</i>	18
3.6	Diagram alir proses yang perlu dilalui oleh aplikasi <i>WiFi Web Login</i>	19
4.1	Diagram Kelas Rinci.	23
4.2	Diagram Interaksi Deteksi Jaringan.	28
4.3	Diagram Interaksi Penciptaan Password.	29
4.4	Diagram Interaksi Penyimpanan Informasi Login.	31
4.5	Rancangan Antarmuka Notifikasi.	32
4.6	Rancangan Antarmuka Message Box.	32
4.7	Rancangan Antarmuka Message Box Saat Menu Delete Informasi Login Sedang Diakses.	32
4.8	Rancangan Antarmuka Web Browser.	33
4.9	Rancangan Antarmuka Web Browser Saat Menu Delete Informasi Login Sedang Diakses.	34

BAB 1

PENDAHULUAN

Bab ini menjelaskan mengenai latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, metodologi penelitian, dan sistematika pembahasan.

1.1 Latar Belakang

Internet adalah salah satu hal yang sulit dipisahkan dari keseharian manusia masa kini. Salah satu cara seseorang dapat mengakses internet adalah dengan menggunakan teknologi WiFi. WiFi mengharuskan pengguna terhubung pada suatu *access point*. *Access point* tersebut dapat memiliki dua status, yaitu terproteksi atau tidak terproteksi. Proteksi pada *access point* dapat dilakukan dengan beberapa cara, yaitu menggunakan protokol IEEE 802.11[1], atau menggunakan *captive portal*. Alat yang sudah pernah terhubung dengan *access point* yang diproteksi dengan protokol IEEE 802.11 akan dengan mudah terhubung kembali dengan *access point* tersebut karena alat tersebut biasanya sudah menyimpan *password* untuk *access point* yang bersangkutan. Alat yang akan terhubung dengan *access point* yang diproteksi menggunakan *captive portal* belum memiliki cara untuk mengingat *username* dan *password* untuk *captive portal* tersebut sehingga login otomatis belum dapat dilakukan untuk *access point* jenis ini.

Captive portal banyak digunakan untuk proteksi *access point* pada tempat-tempat umum seperti lingkungan universitas, *starbucks*, *McDonald's*, dan beberapa tempat yang dapat diakses melalui *@wifi.id*, *free@wifi.id* dan *access point* sejenis. Oleh karena itu, dibutuhkan mekanisme yang bisa membantu proses login untuk *access point* tipe ini. Terdapat dua cara untuk menciptakan mekanisme ini, yaitu dengan mengintegrasikannya dengan sistem operasi, atau menggunakan perangkat lunak pihak ketiga. Untuk dapat melakukan pengintegrasian mekanisme tersebut dengan sistem operasi, dibutuhkan akses kepada kode sumber sistem operasi tersebut. Oleh karena itu, pilihan yang lebih bijak sebagai seseorang yang tidak memiliki akses tersebut adalah dengan menciptakan perangkat lunak pihak ketiga.

Perangkat lunak yang dibuat harus mampu menyimpan informasi login dan mengirimkannya kembali secara otomatis kepada *captive portal*. Perangkat lunak tersebut juga harus mampu mengamankan informasi login yang sudah disimpan agar tidak dapat diakses oleh sembarang orang. Perangkat lunak ini juga harus mampu melakukan identifikasi *captive portal* yang berbeda agar tidak salah memasukkan informasi login.

1.2 Rumusan Masalah

Rumusan masalah yang dibahas pada penelitian ini adalah sebagai berikut:

- Bagaimana caranya melakukan implementasi login otomatis pada *captive portal* yang memiliki tingkat kenyamanan yang setara dengan login otomatis pada proteksi WiFi berbasis protokol IEEE 802.11?
- Apa saja yang perlu dilakukan untuk mengamankan *username* dan *password* yang disimpan oleh user?
- Informasi apa saja yang dibutuhkan untuk menciptakan identitas unik untuk setiap *captive portal* pada jaringan yang berbeda?

1.3 Tujuan Penelitian

Tujuan penelitian ini adalah sebagai berikut:

- Melakukan implementasi login otomatis pada *captive portal* yang memiliki tingkat kenyamanan yang setara dengan login otomatis pada proteksi WiFi berbasis protokol IEEE 802.11.
- Memastikan *username* dan *password* pengguna disimpan secara aman.
- Menentukan informasi yang dibutuhkan untuk menciptakan identitas unik untuk setiap *captive portal* pada jaringan yang berbeda.

1.4 Batasan Masalah

Batasan masalah dari penelitian ini adalah sebagai berikut:

- Perangkat lunak dibangun untuk sistem operasi Windows 8 sampai dengan Windows 10.
- Perangkat lunak dibangun menggunakan bahasa pemrograman C#.
- Elemen keamanan informasi yang diimplementasikan pada perangkat lunak ini adalah enkripsi *username* dan *password* yang disimpan oleh user.

1.5 Metodologi Penelitian

Metodologi penelitian yang dilakukan pada penelitian ini adalah sebagai berikut:

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

Cara kerja dan protokol-protokol yang terkait dengan *captive portal*.

Pemrograman menggunakan *.NET framework*.

Universal Windows Platform (UWP).

Penggunaan kelas WebBrowser pada C#.

Penggunaan objek PasswordVault pada C#.

2. Melakukan analisis perangkat lunak sejenis.
3. Melakukan analisis kebutuhan untuk mengimplementasikan mekanisme login otomatis ini.
4. Merancang perangkat lunak login otomatis ini.
5. Melakukan implementasi hasil rancangan dengan bahasa pemrograman C# pada sistem operasi Windows 10.
6. Melakukan pengujian terhadap perangkat lunak untuk menghasilkan perbaikan terhadap perangkat lunak tersebut.
7. Membuat kesimpulan dari hasil penelitian dan saran untuk penelitian selanjutnya.

1.6 Sistematika Pembahasan

Laporan skripsi ini terdiri dari beberapa bab, yaitu:

1. Bab Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, metodologi penelitian, dan sistematika pembahasan.

2. Bab Dasar Teori

Bab ini berisi dasar-dasar teori dasar mengenai *captive portal*, *.NET framework*, *Universal Windows Platform* (UWP), dokumentasi kelas WebBrowser dan dokumentasi objek PasswordVault.

3. Bab Analisis

Bab ini berisi analisis kebutuhan untuk perancangan dan pembuatan perangkat lunak login otomatis untuk proteksi WiFi berbasis web.

4. Bab Perancangan

Bab ini berisi perancangan perangkat lunak login otomatis untuk proteksi WiFi berbasis web.

5. Bab Implementasi dan Pengujian

Bab ini berisi implementasi perangkat lunak login otomatis untuk proteksi WiFi berbasis web beserta pengujian dan hasil perbaikannya.

6. Bab Kesimpulan dan Saran

Bab ini berisi kesimpulan dari penelitian ini dan saran yang diberikan untuk penelitian selanjutnya.

BAB 2

DASAR TEORI

Bab ini menjelaskan mengenai teori-teori yang digunakan dalam penelitian ini, antara lain penjelasan mengenai *captive portal*, *.NET framework*, *Common Laguage Infrastructure*, *Universal Windows Platform*, kelas *WebView*, dan kelas *PasswordVault*.

2.1 *Captive Portal*

Captive portal adalah *router*¹ atau *gateway*² yang akan menutup koneksi eksternal sampai klien yang bersangkutan sudah terotentikasi[2]. Cara kerja *captive portal* secara umum adalah sebagai berikut:

1. Memberikan alamat IP melalui DHCP pada perangkat yang baru terhubung.
2. Menutup seluruh akses kecuali ke *captive portal server*.
3. Mengarahkan seluruh *request* HTTP ke *captive portal*.
4. Menampilkan aturan penggunaan, informasi pembayaran, dan/atau halaman *login*.
5. Membuka akses jika pengguna telah menyetujui aturan penggunaan atau telah melakukan *login*.
6. Opsional: Menutup akses saat pengguna telah melewati batas waktu tertentu.

Akan tetapi, pada prakteknya, implementasi *captive portal* sangat beragam dan bersifat *ad-hoc*[3]. Beberapa perilaku *captive portal* lain yang teramati adalah sebagai berikut:

- Memaksa pengguna untuk tetap membuka satu *browser window*. Teknik ini membantu mencegah pencurian koneksi pengguna dengan duplikasi alamat MAC.
- Menggunakan otorisasi yang terbatas oleh waktu. Pengguna harus berinteraksi kembali dengan portal setelah waktu tertentu.

¹Alat yang meneruskan paket data ke bagian dari jaringan yang dituju.

²Alat yang digunakan untuk menghubungkan dua jaringan yang berbeda, biasanya berupa hubungan ke internet.

Kode Status HTTP 511

Kode status HTTP adalah bilangan bulat positif yang terdiri dari 3 digit[4]. Kode status ini dikirimkan sebagai hasil dari usaha untuk memahami *request*. Kode status HTTP bersifat *extensible*. Secara garis besar, kode status HTTP dibagi menjadi 5 bagian, yaitu:

- 1xx (Informatif): *Request* telah diterima dan proses dapat dilanjutkan.
- 2xx (Berhasil): *Request* telah diterima dan dimengerti.
- 3xx (*Redirection*): Aksi selanjutnya perlu dilakukan untuk memenuhi *request*.
- 4xx (Kesalahan Klien): *Request* mengandung sintaks yang buruk.
- 5xx (Kesalahan *Server*): *Server* tidak dapat memenuhi *request* yang valid.

Kode status HTTP 511 menandakan bahwa klien perlu melakukan otentikasi untuk mendapatkan akses pada jaringan yang bersangkutan[5]. Respon dengan kode status ini harus menyertakan *link* ke sumber yang memungkinkan pengguna untuk memasukkan kredensial. Selain itu, respon dengan kode status ini tidak boleh diberikan oleh server tujuan. Respon ini dimaksudkan sebagai kontrol akses pada jaringan yang akan diberikan oleh komponen perantara dalam jaringan. Respon dengan kode status 511 tidak boleh disimpan oleh *cache*.

Kode Status HTTP 511 dan *Captive Portal*

Kode status 511 diciptakan untuk mengurangi masalah yang ditimbulkan oleh *captive portal* kepada perangkat lunak yang mengharapkan respon dari server tujuan, bukan dari komponen perantara dalam jaringan. Sebagai contoh, perangkat lunak yang bersangkutan mengirimkan *request* HTTP pada port TCP 80 seperti pada *listing 1* dan server akan mengirimkan respon dengan kode status 511 seperti pada *listing 2*.

```
1 GET /index.htm HTTP/1.1
2 Host: www.example.com
```

Listing 1: *Request* HTTP

Respon ini memungkinkan klien untuk mendeteksi bahwa respon tersebut bukan berasal dari server tujuan. Selain itu, elemen meta pada HTML yang disajikan memungkinkan klien untuk melakukan login pada *link* yang diberikan.

2.2 .NET Framework

.NET adalah platform yang bersifat *general purpose* untuk membangun aplikasi[6]. .NET memberikan lingkungan untuk melakukan pemrograman *high-level* dan tetap memberikan akses *low-level* ke memori dan beberapa API. .NET memiliki beberapa implementasi berdasarkan standar *open*. .NET yang mendefinisikan dasar-dasar yang harus dimiliki oleh platform ini. Implementasi-implementasi ini memberikan dukungan bagi beberapa *chip* (seperti x86/x64 dan ARM) dan beberapa sistem operasi (seperti Windows, Linux, iOS, Android, dan macOS).

```
1 HTTP/1.1 511 Network Authentication Required
2 Content-Type: text/html
3
4 <html>
5     <head>
6         <title>Network Authentication Required</title>
7         <meta http-equiv="refresh"
8             content="0; url=https://login.example.net/">
9     </head>
10    <body>
11        <p>You need to <a href="https://login.example.net/">
12            authenticate with the local network</a> in order to gain
13            access.</p>
14    </body>
15 </html>
```

Listing 2: Respon HTTP 511

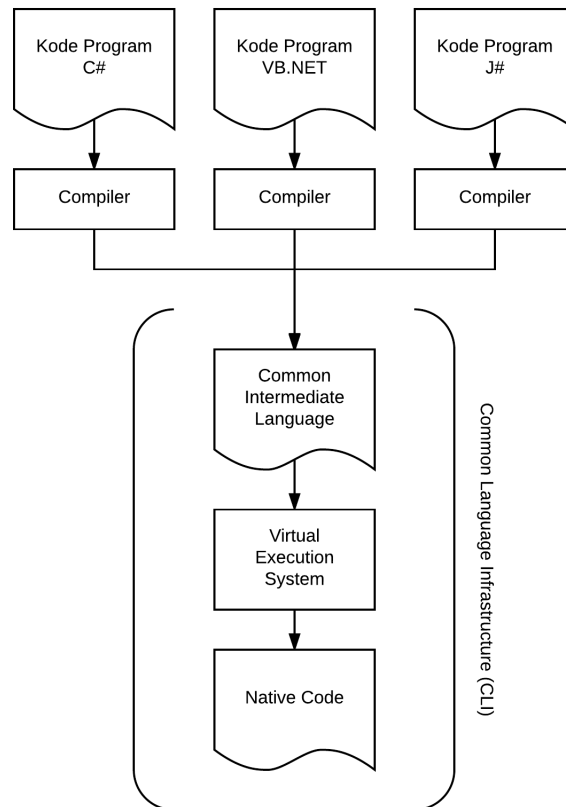
2.3 Common Language Infrastructure (CLI)

.NET memberikan kebebasan kepada *developer* untuk memilih bahasa yang ingin digunakan. Hal ini dapat dilakukan selama bahasa tersebut mendukung .NET. Kebebasan memilih bahasa ini dimungkinkan karena .NET menggunakan spesifikasi *Common Language Infrastructure* (CLI).

Komponen-komponen penting dalam CLI di antaranya adalah[7]:

- *Common Type System* (CTS)
CTS memberikan *type system* yang kaya dan mendukung tipe dan operasi yang ditemukan pada banyak bahasa pemrograman.
- *Metadata*
CLI menggunakan *metadata* untuk menjelaskan dan mereferensi tipe-tipe yang didefinisikan oleh CTS.
- *Common Language Specification* (CLS)
CLS adalah perjanjian desainer bahasa pemrograman dan desainer *framework*. Perjanjian ini menentukan bagian minimal dari CTS yang harus diimplementasikan oleh bahasa pemrograman dan *framework* yang bersangkutan.
- *Virtual Execution System* (VES)
VES mengimplementasikan model CTS dan memastikan hal tersebut berjalan sebagaimana mestinya. VES berfungsi untuk menjalankan program yang ditulis untuk CLI.

Seperti yang dijelaskan pada Gambar 2.1, beberapa bahasa pemrograman yang kompatibel dengan .NET adalah C#, VB.NET, dan J#[6]. Setelah kode program dari bahasa-bahasa tersebut diterjemahkan oleh *compiler*nya masing-masing, maka akan terbentuk *Common Intermediate Language* (CIL) yang kemudian akan dibaca dan dieksekusi oleh VES[7]. Implementasi VES pada .NET bernama *Common Language Runtime* (CLR).



Gambar 2.1: Diagram alir proses yang dilalui oleh kode program dalam CLI.

2.4 *Universal Windows Platform (UWP)*

Windows 8 memperkenalkan *Windows Runtime* (WinRT) yang merupakan arsitektur aplikasi umum untuk Windows[8]. Saat Windows Phone 8.1 keluar, Windows Runtime pada Windows 8 dan Windows Phone 8.1 disejajarkan agar *developer* dapat membangun satu aplikasi yang dapat dijalankan pada Windows 8 dan Windows Phone 8.1. *Universal Windows Platform* (UWP) pertama kali diperkenalkan pada Windows 10 sebagai perubahan dari model *Windows Runtime*. UWP tidak hanya dapat memanggil API dari WinRT, namun juga API spesifik dari device yang bersangkutan (seperti Win32 dan .NET).

2.5 Kelas WebView Pada UWP

Kelas WebView pada UWP memungkinkan *developer* untuk menampung konten HTML pada suatu aplikasi[9]. WebView tidak mendukung masukan 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.

Beberapa *property* yang dimiliki oleh WebView adalah:

- DocumentTitle

Menyimpan *title* dari dokumen yang sedang ditampilkan dalam bentuk String.

- BaseUri

Menyimpan *uri* dari dokumen yang sedang ditampilkan dalam bentuk Uri.

Beberapa *method* yang dimiliki oleh WebView adalah:

- InvokeScriptAsync(String scriptName, String[] arguments)

Method ini digunakan untuk melakukan eksekusi script tertentu pada HTML dengan argumen yang diberikan dalam bentuk *array of string*.

- Navigate(Uri source)

Method ini digunakan untuk membuka URI tertentu.

Salah satu *event* yang dimiliki oleh kelas WebView adalah *event* ScriptNotify. *Event* ini berguna untuk menangkap hasil dari fungsi javascript window.external.notify.

2.6 *Interface* IBackgroundTask Pada UWP

Interface IBackgroundTask pada UWP memiliki metode yang dapat dijalankan di belakang layar[9]. *Interface* ini memiliki satu metode yang harus diimplementasikan oleh kelas yang mengimplementasikan *interface* ini, yaitu:

- Run(IBackgroundTaskInstance taskInstance)

Metode ini adalah metode yang dijalankan berdasarkan *trigger* yang diberikan oleh sistem operasi. Metode ini memiliki satu parameter yaitu taskInstance yang merupakan *instance* dari *task* yang diciptakan oleh sistem operasi.

2.7 Kelas PasswordVault Pada Windows

Kelas PasswordVault merupakan komponen dari Windows Runtime API, dan bukan .NET[9]. Kelas ini merepresentasikan pengunci kredensial. Kredensial yang disimpan menggunakan kelas ini hanya dapat diakses oleh aplikasi atau *service* yang bersangkutan. Beberapa *method* yang dimiliki oleh kelas PasswordVault adalah:

- Add(PasswordCredential credential)

Method ini berfungsi untuk memasukkan kredensial.

- Retrieve(String resource, String userName)

Method ini berfungsi untuk mengambil kredensial yang tersimpan di dalam objek PasswordVault.

- Remove(PasswordCredential credential)

Method ini berfungsi untuk menghapus kredensial yang tersimpan di dalam objek PasswordVault.

BAB 3

ANALISIS

Bab ini menjelaskan mengenai analisis metode penyimpanan informasi login, analisis metode rekam dan kirim informasi login, analisis metode deteksi *captive portal*, analisis perangkat lunak, serta analisis perangkat lunak sejenis.

3.1 Analisis 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*, username, password, dan langkah-langkah login seperti menekan tombol. *Connection fingerprint* dibentuk dari SSID WiFi, url halaman login *captive portal*, dan potongan unik dokumen html halaman tersebut (bisa menggunakan konten tag head). Ketiga hal tersebut diperlukan untuk mengidentifikasi halaman *captive portal* karena halaman yang sama pada WiFi yang berbeda memiliki kredensial yang berbeda, dan halaman yang berbeda pada WiFi yang sama juga memiliki kredensial yang berbeda. 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 batas-

an 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`.

3.2 Analisis Metode Rekam dan Kirim Informasi Login

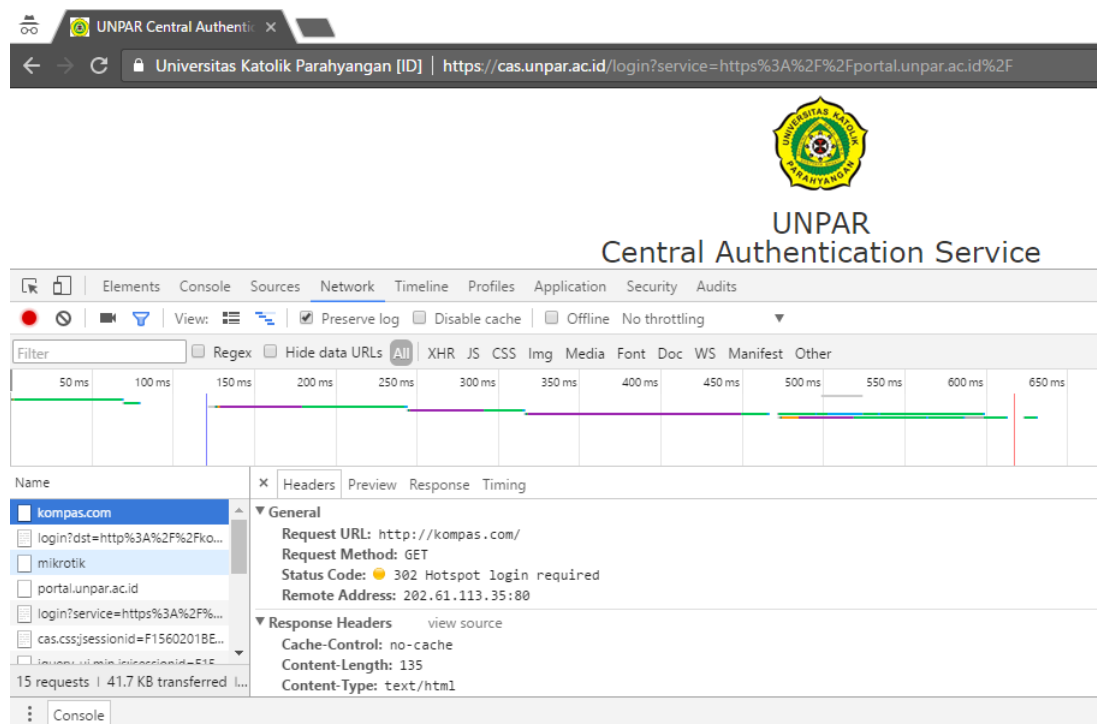
Kelas `WebView` pada *Universal Windows Platform* (UWP) hanya dapat dihubungkan dengan kode C# menggunakan javascript. *Method* yang digunakan untuk melakukan eksekusi javascript pada `WebView` adalah `InvokeScriptAsync`. Metode ini memiliki parameter string berupa nama fungsi javascript yang ingin dipanggil 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[10]`) 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.

3.3 Analisis Metode Deteksi *Captive Portal*

Berdasarkan teori *captive portal* pada bab 2, dijelaskan bahwa 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.

Gambar 3.1 menunjukkan *captive portal* Unpar yang menggunakan kode status HTTP 302 untuk memberitahu klien bahwa diperlukan otentikasi jaringan. Berdasarkan teori mengenai kode



Gambar 3.1: *Screenshot HTTP header yang dikirimkan oleh captive portal milik Unpar.*

status HTTP yang dijabarkan pada bab 2, kode status HTTP 3xx adalah kode status yang menyatakan *redirection*. Oleh karena adanya perbedaan implementasi seperti ini, deteksi *captive portal* menggunakan kode status HTTP tidak dapat dilakukan.

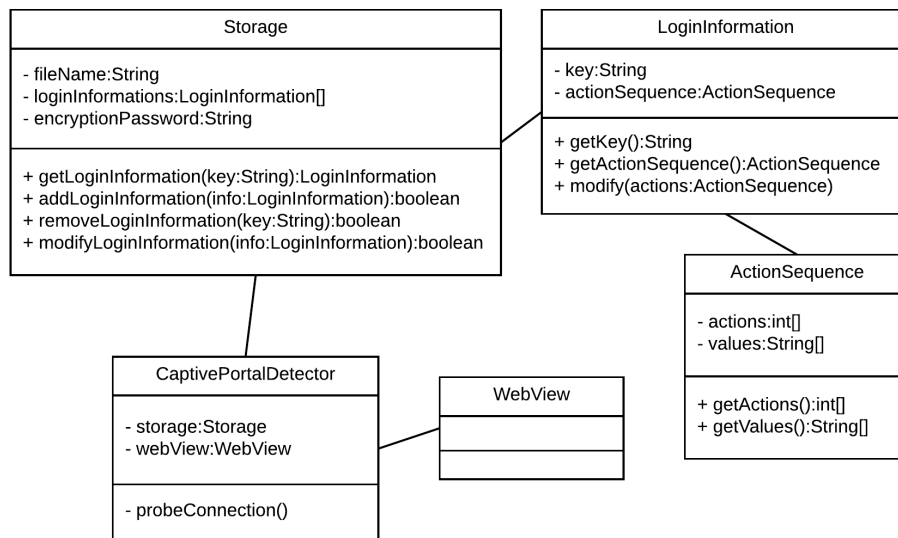
Persamaan implementasi yang dimiliki oleh setiap *captive portal* adalah dibutuhkankannya *redirection* yang dapat dikenali oleh *web browser* agar klien selalu diarahkan ke halaman *captive portal* tersebut sebelum melakukan otentikasi. Oleh karena itu, untuk setiap HTTP *request* yang dilakukan oleh 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.

3.4 Analisis Perangkat Lunak

Bagian ini menjelaskan mengenai diagram *use case* dan diagram kelas perangkat lunak yang dibangun.

3.4.1 Analisis Kelas



Gambar 3.2: Diagram kelas untuk perangkat lunak yang dibangun.

Diagram kelas untuk perangkat lunak yang dibangun dapat dilihat pada gambar 3.2. Seperti pada gambar 3.2, 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.

Atribut:

- `storage`: Kelas yang menangani penyimpanan informasi login.
- `webView`: Kelas bawaan UWP yang berfungsi sebagai pengganti *web browser*.

Metode:

- `probeConnection()`: Metode untuk mendeteksi *captive portal*.

- **Storage**

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

Atribut:

- `fileName`: Nama file yang digunakan untuk menyimpan informasi login.
- `loginInformations`: Representasi kelas dari informasi login yang ingin disimpan.
- `encryptionPassword`: *Password* yang digunakan untuk melakukan enkripsi terhadap file yang digunakan untuk menyimpan informasi login.

Metode:

- `getInformationLogin(key:String)`: Metode untuk mengambil informasi login berdasarkan suatu *unique key*.
 - `addInformationLogin(info:LoginInformation)`: Metode untuk menambahkan informasi login baru.
 - `removeInformationLogin(key:String)`: Metode untuk menghapus informasi login yang sudah disimpan berdasarkan suatu *unique key*.
- **LoginInformation**
Kelas ini digunakan untuk menyimpan informasi login dalam bentuk key atau fingerprint, serta `ActionSequence`
Atribut:
 - `key`: *Unique key* yang digunakan untuk mengidentifikasi *instance* `LoginInformation` ini.
 - `actionSequence`: Kelas yang berisi langkah-langkah login.Metode:
 - `getKey()`: Metode untuk mengambil key.
 - `getActionSequence()`: Metode untuk mengambil `actionSequence`.
 - **ActionSequence**
Kelas ini digunakan untuk menyimpan langkah-langkah login dalam bentuk urutan aksi dan nilai-nilai yang berkaitan dengan aksi tersebut.
Atribut:
 - `actions`: Urutan aksi yang perlu dilakukan selama login.
 - `values`: Daftar nilai-nilai yang berkaitan dengan aksi-aksi tersebut.Metode:
 - `getActions()`: Metode untuk mengambil `actions`.
 - `getValues()`: Metode untuk mengambil `values`.

3.4.2 Analisis *Use Case*

Diagram *use case* untuk perangkat lunak yang dibangun dapat dilihat pada gambar [3.3](#).

Skenario merekam informasi login

Nama: Merekam informasi login

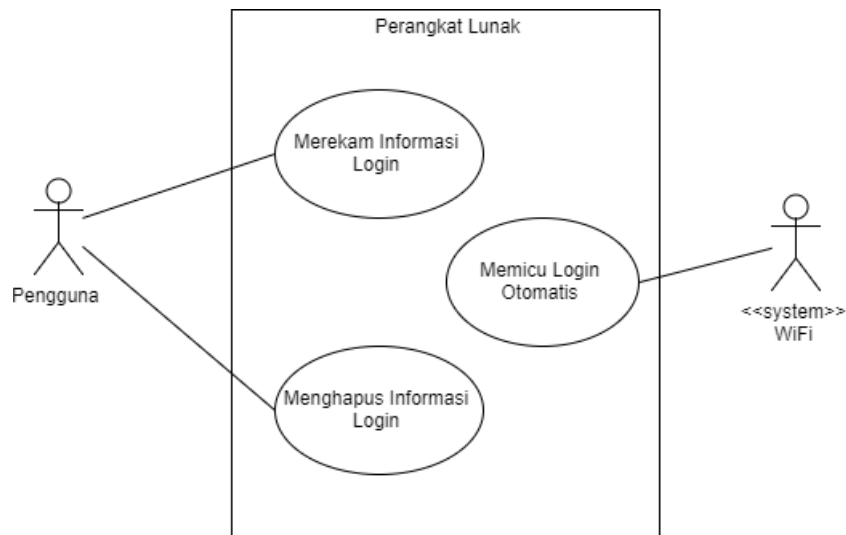
Aktor: Pengguna

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

Deskripsi: Pengguna menyimpan informasi login.

Kondisi Akhir: Informasi login tersimpan di dalam perangkat lunak.

Skenario:



Gambar 3.3: Diagram *use case* untuk perangkat lunak yang dibangun.

1. Pengguna memasukkan informasi login ke dalam form HTML.
2. 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:

1. Pengguna memilih untuk menghapus informasi login.
2. Sistem menghapus informasi login.

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.

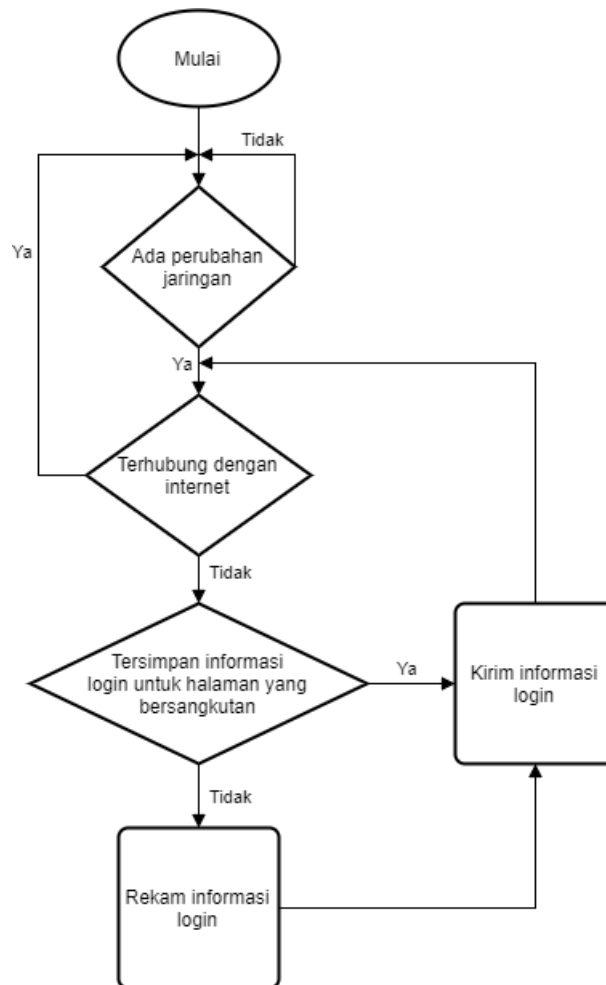
Kondisi Akhir: Klien terotentikasi pada *captive portal*.

Skenario:

1. Sistem mendeteksi adanya koneksi WiFi yang terjadi.
2. Sistem mendeteksi adanya *captive portal*.

3. Sistem mendeteksi adanya informasi login untuk *captive portal* tersebut.
4. Sistem mengirimkan informasi login kepada *captive portal*.
5. Sistem mendeteksi koneksi dengan internet dan klien sudah terotentikasi pada *captive portal* tersebut.

3.4.3 Analisis Alir



Gambar 3.4: Diagram alir untuk perangkat lunak yang dibangun.

Alir perangkat lunak yang dibangun dapat dilihat pada gambar 3.4. Perangkat lunak akan memeriksa apakah terjadi perubahan jaringan dari tidak terhubung menjadi terhubung. Jika kondisi tersebut terpenuhi, maka perangkat lunak akan memeriksa apakah terhubung dengan internet. Jika tidak terhubung dengan internet, maka diasumsikan bahwa terdapat *captive portal* dan perangkat lunak akan memeriksa apakah sudah ada informasi login terkait halaman yang termuat. Jika ada, maka perangkat lunak akan melakukan login otomatis. Akan tetapi jika tidak, perangkat lunak akan merekam terlebih dahulu informasi login, baru melakukan login otomatis.

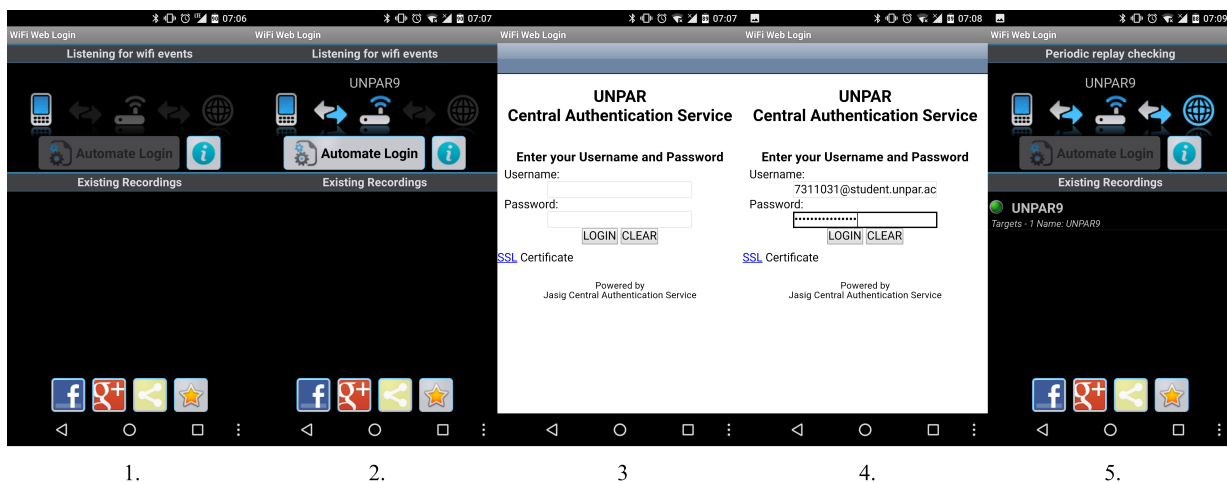
3.5 Analisis Perangkat Lunak Sejenis

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*¹.

Tampilan *WiFi Web Login*

Pada gambar 3.5 diperlihatkan langkah-langkah login *captive portal* Unpar sebagai berikut:

1. Menunggu koneksi WiFi.
2. Mendeteksi koneksi WiFi UNPAR9.
3. Menampilkan halaman *captive portal* setelah pengguna menekan tombol *Automate Login*.
4. Pengguna memasukkan username dan password lalu menekan tombol login.
5. Sambungan ke internet terdeteksi dan dilakukan pemeriksaan sambungan berkala.



Gambar 3.5: Tampilan aplikasi *WiFi Web Login*.

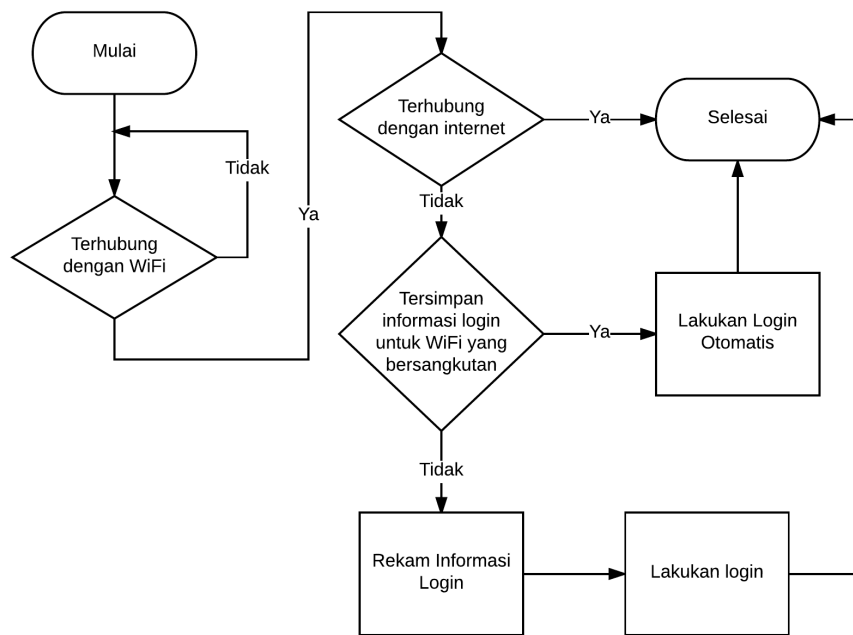
Diagram Alir *WiFi Web Login*

Langkah-langkah yang perlu ditempuh oleh aplikasi *WiFi Web Login* dapat digambarkan oleh diagram alir.

Berdasarkan gambar 3.6, langkah-langkah yang harus ditempuh untuk melakukan login wifi berbasis web pada aplikasi ini adalah:

1. Deteksi sambungan dengan wifi yang bersangkutan.
2. Deteksi hubungan dengan internet.

¹<https://play.google.com/store/apps/details?id=co.uk.syslynx.wifiwebloginapp>



Gambar 3.6: Diagram alir proses yang perlu dilalui oleh aplikasi *WiFi Web Login*.

3. Jika tidak terjadi hubungan dengan internet, deteksi apakah tersimpan informasi login untuk wifi yang bersangkutan.
4. Jika terdapat informasi login untuk wifi yang bersangkutan maka lakukan login otomatis.
5. Jika tidak terdapat informasi login untuk wifi yang bersangkutan maka rekam informasi login dan lakukan login.

Setelah pengguna sudah pernah melakukan login pertama kali menggunakan aplikasi tersebut, maka aplikasi akan melakukan login otomatis setiap kali terhubung dengan wifi yang bersangkutan. Alir *WiFi Web Login* memiliki prinsip yang sama dengan alir perangkat lunak yang dibangun sesuai dengan gambar 3.4.

BAB 4

PERANCANGAN

Bab ini menjelaskan mengenai perancangan yang disusun dari analisis yang dilakukan pada bab 3. 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.

4.1 Perancangan Kelas

Gambar 4.1 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`, dan `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:

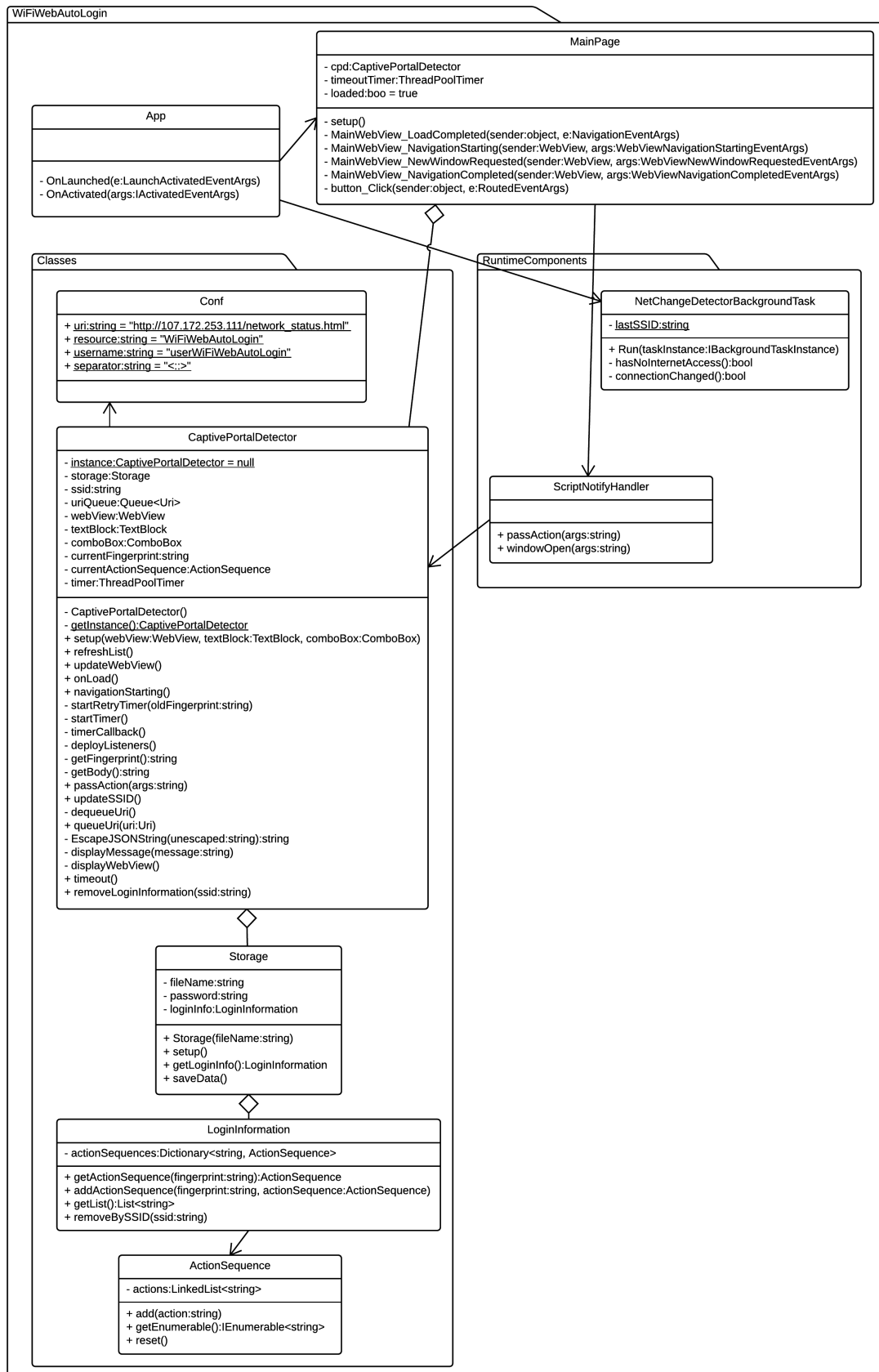
- **passAction**

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:



Gambar 4.1: Diagram Kelas Rinci.

- `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.

4.2 Perancangan Algoritma dan Struktur Data

Sub-bab ini menjelaskan mengenai perancangan algoritma untuk melakukan deteksi *captive portal*, struktur data dan format *fingerprint*, serta struktur data untuk menyimpan informasi login.

4.2.1 Algoritma Deteksi *Captive Portal*

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

```

Data: Network Status
Result: Captive portal detected / Not captive portal
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;
        else
            Not captive portal;
        end
    end
end

```

Algorithm 1: Algoritma Deteksi *Captive Portal*

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. Perangkat lunak akan mencoba untuk mengakses halaman panicingan yang beralamatkan pada `http://107.172.253.111/network_status.html` pada saat perangkat lunak pertama kali dijalankan. 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.

4.2.2 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>`.

4.2.3 Struktur Data LoginInformation

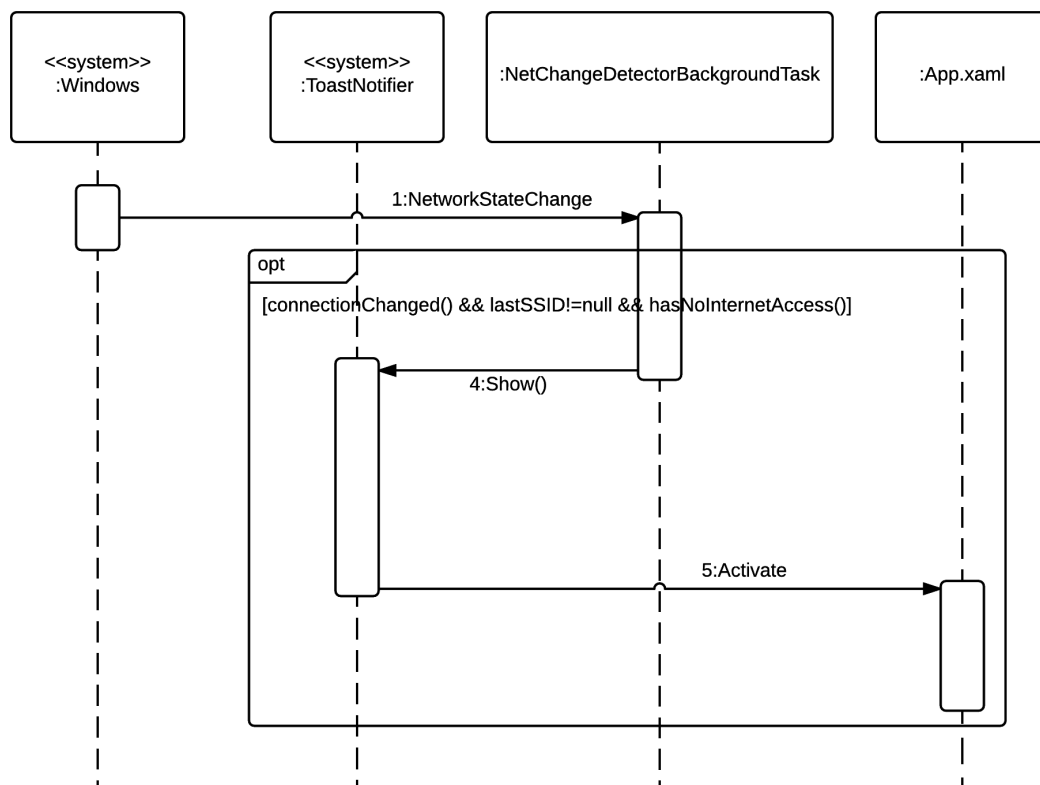
Kelas LoginInformation memiliki daftar objek bertipe ActionSequence yang disimpan pada properti actionSequences bertipe Dictionary. Kelas Dictionary dapat menyimpan data yang berupa pasangan *key-value*. *Key* yang digunakan bertipe string dan merupakan *fingerprint* suatu halaman. Value yang disimpan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

4.3 Perancangan Interaksi Perangkat Lunak

Beberapa interaksi yang dimodelkan menggunakan diagram interaksi adalah interaksi deteksi jaringan, interaksi penciptaan password, dan interaksi penyimpanan informasi login.

4.3.1 Perancangan Interaksi Deteksi Jaringan

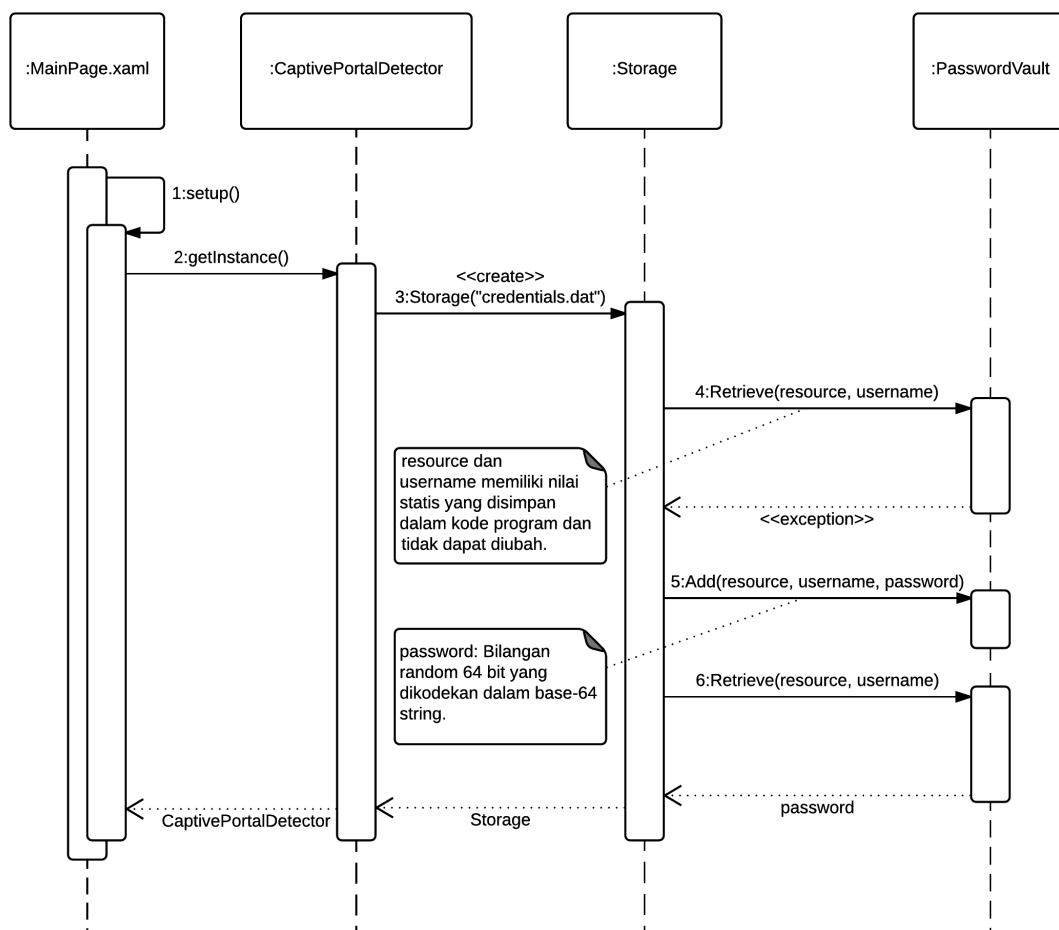


Gambar 4.2: Diagram Interaksi Deteksi Jaringan.

Gambar 4.2 menjelaskan mengenai interaksi antar objek dalam perangkat lunak untuk melakukan deteksi perubahan jaringan. Interaksi yang terjadi adalah sebagai berikut:

1. Saat komputer mengalami perubahan jaringan (tidak terhubung menjadi terhubung dan sebaliknya, atau terjadi perubahan *cost* jaringan), *trigger* `NetworkStateChange` akan diaktifkan oleh Windows, dan objek `NetChangeDetectorBackgroundTask` yang sudah didaftarkan akan menerima *trigger* tersebut.
2. Jika kondisi `connectionChanged()`, `lastSSID!=null`, dan `hasNoInternetAccess()` terpenuhi, maka:
 - (a) `NetChangeDetectorBackgroundTask` memerintahkan `ToastNotifier` untuk memunculkan notifikasi menggunakan method `Show()`.
 - (b) Saat user menekan tombol "Yes" pada notifikasi, `App.xaml` diaktivasi.

4.3.2 Perancangan Interaksi Penciptaan Password



Gambar 4.3: Diagram Interaksi Penciptaan Password.

Gambar 4.3 menjelaskan mengenai interaksi antar objek dalam perangkat lunak untuk menciptakan password random saat perangkat lunak pertama kali dijalankan. Interaksi yang terjadi adalah

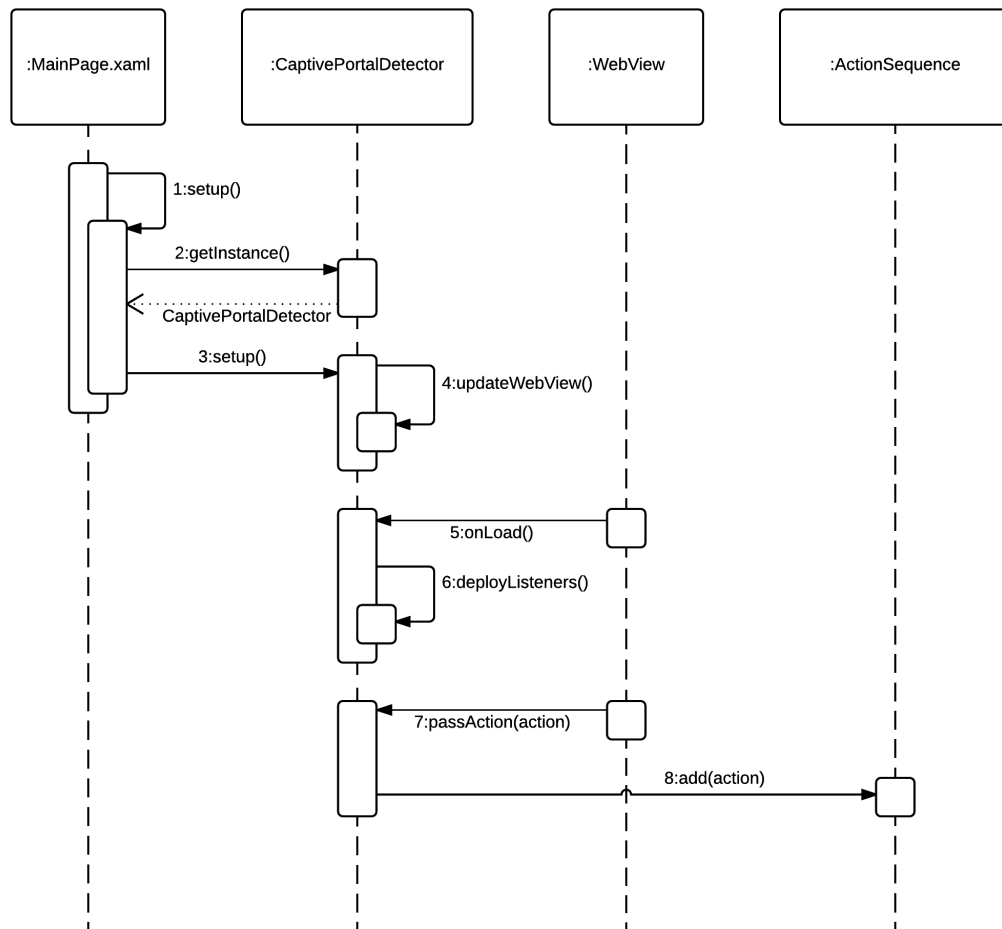
sebagai berikut:

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

4.3.3 Perancangan Interaksi Penyimpanan Informasi Login

Gambar 4.4 menjelaskan mengenai interaksi antar objek dalam perangkat lunak untuk menyimpan informasi login. Interaksi yang terjadi adalah sebagai berikut:

1. MainPage.xaml melakukan `setup()`.
2. MainPage.xaml memanggil metode `getInstance()` pada kelas `CaptivePortalDetector` untuk mendapatkan *instance* `CaptivePortalDetector`. MainPage.xaml mendapatkan *instance* `CaptivePortalDetector`.
3. MainPage.xaml memanggil metode `setup()` pada objek `CaptivePortalDetector`.
4. `CaptivePortalDetector` melakukan `updateWebView()` untuk mengarahkan `WebView` ke URI yang digunakan untuk melakukan deteksi koneksi internet.
5. `WebView` memanggil metode `onLoad()` pada objek `CaptivePortalDetector` saat halaman selesai dimuat.
6. Jika halaman tidak berisi teks "connected" (tanpa tanda petik), `CaptivePortalDetector` melakukan `deployListeners()` untuk menangkap semua *event* yang mungkin dilakukan oleh pengguna pada halaman tersebut.
7. Metode `passAction()` dipanggil pada objek `CaptivePortalDetector` saat pengguna melakukan klik atau input teks untuk mengirimkan aksi yang baru saja dilakukan oleh pengguna.
8. `CaptivePortalDetector` memanggil metode `add()` pada objek `ActionSequence` untuk menyimpan aksi tersebut.



Gambar 4.4: Diagram Interaksi Penyimpanan Informasi Login.

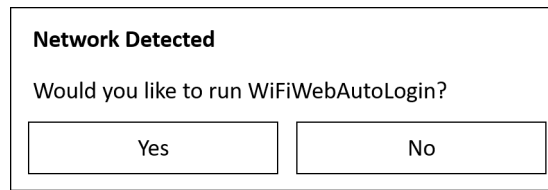
4.4 Perancangan Antarmuka

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

- Antarmuka notifikasi yang muncul setiap kali terhubung dengan WiFi yang menggunakan *captive portal*.
- Antarmuka untuk menampilkan halaman web.
- Antarmuka untuk menampilkan pesan-pesan seperti pesan "Connected." atau "Check your network connection.".

4.4.1 Antarmuka Notifikasi

Gambar 4.5 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



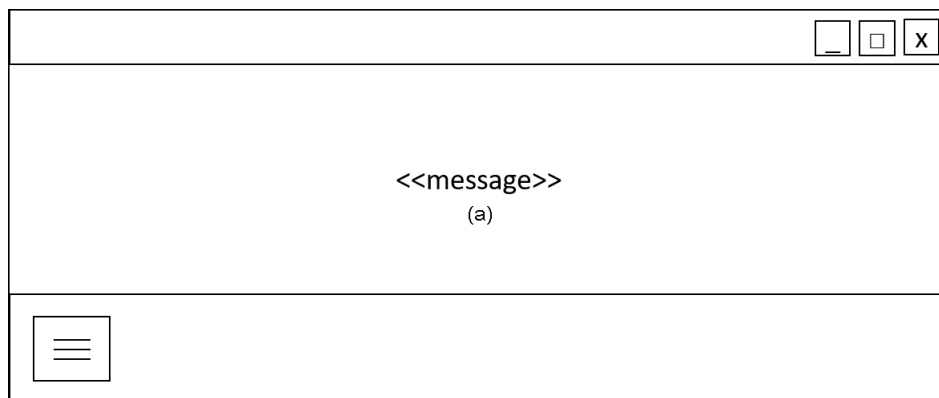
Gambar 4.5: Rancangan Antarmuka Notifikasi.

notifikasi akan hilang. Antarmuka notifikasi adalah antarmuka yang pertama kali akan muncul dalam siklus aplikasi karena kelas `NetChangeDetectorBackgroundTask` didaftarkan pada sistem untuk mendeteksi perubahan jaringan.

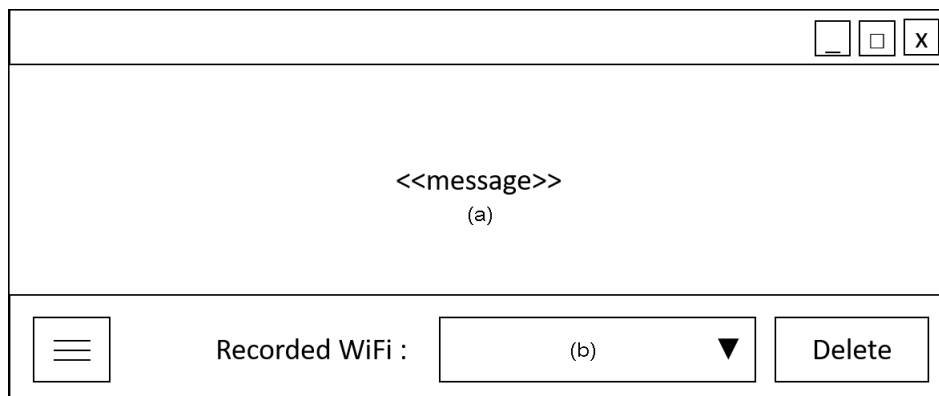
4.4.2 Antarmuka Aplikasi

Antarmuka ini digunakan untuk menampilkan halaman web dan menampilkan pesan dapat disatukan menjadi antarmuka aplikasi yang halaman kontennya dapat diubah menjadi `WebView` saat berada dalam mode web browser, dan menjadi `Label` saat berada dalam mode message box.

Antarmuka Message Box



Gambar 4.6: Rancangan Antarmuka Message Box.



Gambar 4.7: Rancangan Antarmuka Message Box Saat Menu Delete Informasi Login Sedang Diakses.

Gambar 4.6 menampilkan desain antarmuka message box dan gambar 4.7 menampilkan desain antarmuka yang sama setelah pengguna menekan tombol kiri bawah. Antar muka ini memiliki komponen label (a) untuk menaruh pesan serta komponen (b) untuk dapat menghapus WiFi yang sudah terekam. Dengan menghapus WiFi yang terdapat pada komponen ini, pengguna dapat merekam ulang informasi login pada WiFi tersebut.

Antarmuka Web Browser

The diagram illustrates a web browser window. At the top is a title bar with three control buttons: minimize, maximize, and close. The main body of the window is labeled with the identifier '(a)'. Centered within this body is a 'Login' section. This section includes two text input fields, one labeled 'username' and one labeled 'password', stacked vertically. Below these fields is a 'Submit' button. At the bottom of the window, there is a horizontal bar containing a hamburger menu icon on the left side.

Gambar 4.8: Rancangan Antarmuka Web Browser.

Gambar 4.8 menampilkan desain antarmuka *web browser* dan gambar 4.9 menampilkan antarmuka yang sama setelah pengguna menekan tombol kiri bawah. Komponen (a) digunakan untuk menampilkan halaman web yang berkaitan dengan login *captive portal*. Aksi pengguna akan direkam secara otomatis pada komponen ini. Selain itu, pada antarmuka ini terdapat komponen yang sama dengan antarmuka message box, yaitu komponen (b) untuk menghapus SSID WiFi yang sudah terekam.

X

(a)
Login

username

password

Submit

≡

Recorded WiFi :

(b) ▼

Delete

Gambar 4.9: Rancangan Antarmuka Web Browser Saat Menu Delete Informasi Login Sedang Diakses.

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan mengenai lingkungan yang digunakan untuk melakukan implementasi dan pengujian, masalah-masalah yang ditemui pada saat implementasi dan solusi yang dijalankan, pengujian yang dilakukan, baik pengujian fungsional maupun pengujian eksperimental, beserta hasilnya.

5.1 Lingkungan Implementasi dan Pengujian

Implementasi dan pengujian dilakukan pada laptop Asus N46VM dengan spesifikasi sebagai berikut:

- Sistem operasi: Windows 10
- Prosesor: Intel(R) Core(TM) i7-3610QM
- RAM: 12GB
- *Network adapter*: Qualcomm Atheros AR9485WB-EG Wireless Network Adapter

Implementasi dilakukan menggunakan bahasa pemrograman C# dan IDE¹ Microsoft Visual Studio Community 2015.

5.2 Masalah Implementasi dan Solusinya

Terdapat beberapa masalah implementasi 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 4.1.
- 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`

¹Integrated Development Environment

pada saat halaman selesai dimuat, dan menghubungkannya 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 tertulis pada bab 4 sudah merupakan hasil revisi dari analisis masalah implementasi ini.

5.3 Pengujian Fungsional

Pengujian fungsional dilakukan pada jaringan WiFi di kost di jalan Ciumbuleuit nomor 149, Bandung, dengan SSID "C149Net". Pengujian fungsional dilakukan pada tanggal 4 April 2017.

5.3.1 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.

5.3.2 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.

5.4 Pengujian Eksperimental

Pengujian eksperimental dilakukan untuk memeriksa apakah perangkat lunak dapat berjalan pada beragam *captive portal*. Pengujian eksperimental dilakukan pada tanggal 5 April 2017.

5.4.1 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.

5.4.2 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 CAS². 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

² *Central Authorization Service*

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 dan penyebab terjadinya hal ini juga sama dengan yang terjadi pada WiFi UNPAR9.

BAB 6

KESIMPULAN DAN SARAN

Bab ini memaparkan kesimpulan beserta dengan saran akan hal-hal yang dapat dilakukan untuk mengembangkan penelitian ini.

6.1 Kesimpulan

Beberapa hal yang dapat disimpulkan dari penelitian ini antara lain:

- 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*.

6.2 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.

DAFTAR REFERENSI

- [1] A.S.Rumale and D. D. N. Chaudhari, “Ieee 802.11x, and wep, eap,wpa / wpa2,” *International Journal of Computer Technology and Applications*, vol. 2, pp. 1945–1950, 2011.
- [2] B. Potter and B. Fleck, *802.11 Security*. O’Reilly, 2002.
- [3] HTTP Working Group, “Captive Portals.” <https://github.com/httpwg/wiki/wiki/Captive-Portals>, 2016. [Online; diakses 11-September-2016].
- [4] Internet Engineering Task Force, “Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.” <https://tools.ietf.org/html/rfc7231>, 2016. [Online; diakses 24-September-2016].
- [5] Internet Engineering Task Force, “Additional HTTP Status Codes.” <https://tools.ietf.org/html/rfc6585>, 2016. [Online; diakses 24-September-2016].
- [6] R. Lander, “.NET Primer.” <https://docs.microsoft.com/en-us/dotnet/articles/standard/index>, 2016. [Online; diakses 24-September-2016].
- [7] Ecma International, “Common Language Infrastructure (CLI) Partitions I to IV.” <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf>, 2016. [Online; diakses 24-September-2016].
- [8] Microsoft, “Intro to the Universal Windows Platform.” <https://msdn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide>, 2016. [Online; diakses 24-September-2016].
- [9] Microsoft, “Windows API Reference.” <https://msdn.microsoft.com/en-us/library/windows/apps/bg124285.aspx>, 2016. [Online; diakses 24-September-2016].
- [10] Internet Engineering Task Force, “The JavaScript Object Notation (JSON) Data Interchange Format.” <https://tools.ietf.org/html/rfc7159>, 2017. [Online; diakses 6-Maret-2017].

LAMPIRAN A

KODE SUMBER

A.1 Namespace: WiFiWebAutoLogin

```
WiFiWebAutoLogin/Package.appxmanifest
1 <?xml version="1.0" encoding="utf-8"?>
2 <Package xmlns="http://schemas.microsoft.com/appx/manifest/foundation/windows10"
   ↳ xmlns:mp="http://schemas.microsoft.com/appx/2014/phone/manifest"
   ↳ xmlns:uap="http://schemas.microsoft.com/appx/manifest/uap/windows10"
   ↳ IgnorableNamespaces="uap mp">
3   <Identity Name="com.yohanesmario.WiFiWebAutoLogin" Publisher="CN=Yohanes
   ↳ Mario" Version="1.0.1.0" />
4   <mp:PhoneIdentity PhoneProductId="24a4f351-3e45-4b12-946e-bfc3593444b4"
   ↳ PhonePublisherId="00000000-0000-0000-0000-000000000000" />
5   <Properties>
6     <DisplayName>WiFiWebAutoLogin</DisplayName>
7     <PublisherDisplayName>Yohanes Mario</PublisherDisplayName>
8     <Logo>Assets\StoreLogo.png</Logo>
9   </Properties>
10  <Dependencies>
11    <TargetDeviceFamily Name="Windows.Universal" MinVersion="10.0.0.0"
   ↳ MaxVersionTested="10.0.0.0" />
12  </Dependencies>
13  <Resources>
14    <Resource Language="x-generate" />
15  </Resources>
16  <Applications>
17    <Application Id="App" Executable="$targetnametoken$.exe"
   ↳ EntryPoint="WiFiWebAutoLogin.App">
18      <uap:VisualElements DisplayName="WiFiWebAutoLogin"
   ↳ Square150x150Logo="Assets\Square150x150Logo.png"
   ↳ Square44x44Logo="Assets\Square44x44Logo.png"
   ↳ Description="WiFiWebAutoLogin" BackgroundColor="transparent">
19        <uap:DefaultTile Wide310x150Logo="Assets\Wide310x150Logo.png">
20        </uap:DefaultTile>
```

```

21     <uap:SplashScreen Image="Assets\SplashScreen.png" />
22 </uap:VisualElements>
23 <Extensions>
24     <Extension Category="windows.backgroundTasks" EntryPoint="WiFiWebAutoLog_
    ↪ in.RuntimeComponents.NetChangeDetectorBackgroundTask">
25         <BackgroundTasks>
26             <Task Type="systemEvent" />
27         </BackgroundTasks>
28     </Extension>
29 </Extensions>
30 </Application>
31 </Applications>
32 <Capabilities>
33     <Capability Name="internetClient" />
34 </Capabilities>
35 </Package>

```

WiFiWebAutoLogin/App.xaml

```

1 <Application
2     x:Class="WiFiWebAutoLogin.App"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:WiFiWebAutoLogin"
6     RequestedTheme="Light">
7
8 </Application>

```

WiFiWebAutoLogin/App.xaml.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.Diagnostics;
4 using System.IO;
5 using System.Linq;
6 using System.Runtime.InteropServices.WindowsRuntime;
7 using Windows.ApplicationModel;
8 using Windows.ApplicationModel.Activation;
9 using Windows.ApplicationModel.Background;
10 using Windows.ApplicationModel.Core;
11 using Windows.Foundation;
12 using Windows.Foundation.Collections;
13 using Windows.UI.Notifications;
14 using Windows.UI.Xaml;

```

```
15 using Windows.UI.Xaml.Controls;
16 using Windows.UI.Xaml.Controls.Primitives;
17 using Windows.UI.Xaml.Data;
18 using Windows.UI.Xaml.Input;
19 using Windows.UI.Xaml.Media;
20 using Windows.UI.Xaml.Navigation;
21
22 namespace WiFiWebAutoLogin
23 {
24     /// <summary>
25     /// Provides application-specific behavior to supplement the default
26         ↪ Application class.
27     /// </summary>
28     sealed partial class App : Application
29     {
30         /// <summary>
31         /// Initializes the singleton application object. This is the first
32             ↪ line of authored code
33         /// executed, and as such is the logical equivalent of main() or
34             ↪ WinMain().
35         /// </summary>
36         public App()
37         {
38             this.InitializeComponent();
39             this.Suspending += OnSuspending;
40         }
41
42         /// <summary>
43         /// Invoked when the application is launched normally by the end user.
44             ↪ Other entry points
45         /// will be used such as when the application is launched to open a
46             ↪ specific file.
47         /// </summary>
48         /// <param name="e">Details about the launch request and process.</param>
49         protected override void OnLaunched(LaunchActivatedEventArgs e)
50         {
51             // Initialize background task
52             bool taskRegistered = false;
53             string taskName = "CustomBackgroundTask";
54
55             foreach (var task in BackgroundTaskRegistration.AllTasks) {
56                 if (task.Value.Name == taskName) {
```

```

52         task.Value.Unregister(true);
53     }
54 }
55
56 taskName = "NetChangeDetectorBackgroundTask";
57
58 foreach (var task in BackgroundTaskRegistration.AllTasks) {
59     if (task.Value.Name == taskName) {
60         task.Value.Unregister(true);
61     }
62 }
63
64 if (!taskRegistered) {
65     Debug.WriteLine("Register Task");
66
67     // Register task
68     var builder = new BackgroundTaskBuilder();
69
70     builder.Name = taskName;
71     builder.TaskEntryPoint = "WiFiWebAutoLogin.RuntimeComponents.Net_
    ↪ ChangeDetectorBackgroundTask";
72     builder.SetTrigger(new
    ↪ SystemTrigger(SystemTriggerType.NetworkStateChange, false));
73     builder.AddCondition(new
    ↪ SystemCondition(SystemConditionType.UserPresent));
74     BackgroundTaskRegistration task = builder.Register();
75 }
76 #if DEBUG
77 if (System.Diagnostics.Debugger.IsAttached) {
78     //this.DebugSettings.EnableFrameRateCounter = true;
79     this.DebugSettings.EnableFrameRateCounter = false;
80 }
81 #endif
82 Frame rootFrame = Window.Current.Content as Frame;
83
84 // Do not repeat app initialization when the Window already has
    ↪ content,
85 // just ensure that the window is active
86 if (rootFrame == null) {
87     // Create a Frame to act as the navigation context and navigate
    ↪ to the first page
88     rootFrame = new Frame();

```

```
89
90     rootFrame.NavigationFailed += OnNavigationFailed;
91
92     //if (e.PreviousExecutionState ==
93     ↪ ApplicationExecutionState.Terminated) {
94         //TODO: Load state from previously suspended application
95     //}
96
97     // Place the frame in the current Window
98     Window.Current.Content = rootFrame;
99
100 }
101
102 //if (e.PrelaunchActivated == false) {
103     //if (rootFrame.Content == null) {
104         // When the navigation stack isn't restored navigate to the
105         ↪ first page,
106         // configuring the new page by passing required information
107         ↪ as a navigation
108         // parameter
109         rootFrame.Navigate(typeof(MainPage), e.Arguments);
110     //}
111     // Ensure the current window is active
112     Window.Current.Activate();
113 //}
114
115 }
116
117 protected override void OnActivated(IActivatedEventArgs args) {
118     if (args.Kind == ActivationKind.ToastNotification) {
119         var toastArgs = args as ToastNotificationActivatedEventArgs;
120         var arguments = toastArgs.Argument;
121
122         if (arguments == "Yes") {
123             Frame rootFrame = Window.Current.Content as Frame;
124             if (rootFrame == null) {
125                 rootFrame = new Frame();
126                 Window.Current.Content = rootFrame;
127             }
128             rootFrame.Navigate(typeof(MainPage));
129             Window.Current.Activate();
130         }
131         else {
132             CoreApplication.Exit();
133         }
134     }
135 }
```

```

128         }
129
130         ToastNotificationManager.History.Remove("WWAL_TOAST");
131     }
132 }
133
134     /// <summary>
135     /// Invoked when Navigation to a certain page fails
136     /// </summary>
137     /// <param name="sender">The Frame which failed navigation</param>
138     /// <param name="e">Details about the navigation failure</param>
139     void OnNavigationFailed(object sender, NavigationFailedEventArgs e)
140     {
141         throw new Exception("Failed to load Page " +
142             ↪ e.SourcePageType.FullName);
143     }
144
145     /// <summary>
146     /// Invoked when application execution is being suspended. Application
147     ↪ state is saved
148     /// without knowing whether the application will be terminated or
149     ↪ resumed with the contents
150     /// of memory still intact.
151     /// </summary>
152     /// <param name="sender">The source of the suspend request.</param>
153     /// <param name="e">Details about the suspend request.</param>
154     private void OnSuspending(object sender, SuspendingEventArgs e)
155     {
156         var deferral = e.SuspendingOperation.GetDeferral();
157         //TODO: Save application state and stop any background activity
158         deferral.Complete();
159     }
160 }

```

WiFiWebAutoLogin/MainPage.xaml

```

1 <Page
2     x:Class="WiFiWebAutoLogin.MainPage"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:WiFiWebAutoLogin"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

```

```

7      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8      mc:Ignorable="d">
9
10     <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
11         <Grid Margin="0,0,0,52">
12             <TextBlock x:Name="textBlock" TextWrapping="Wrap"
13                 ↪ TextAlignment="Center" Text="" VerticalAlignment="Center"
14                 ↪ HorizontalAlignment="Center" FontSize="16"/>
15             <WebView Name="MainWebView"
16                 ↪ NewWindowRequested="MainWebView_NewWindowRequested"
17                 ↪ DOMContentLoaded="MainWebView_DOMContentLoaded"
18                 ↪ ContentLoading="MainWebView_ContentLoading" LongRunningScriptDetected="MainWebView_LongRunningScriptDetected"
19                 ↪ PermissionRequested="MainWebView_PermissionRequested"
20                 ↪ LoadCompleted="MainWebView_LoadCompleted"
21                 ↪ NavigationStarting="MainWebView_NavigationStarting"
22                 ↪ NavigationCompleted="MainWebView_NavigationCompleted"
23                 ↪ Margin="0,768,0,-768" />
24         </Grid>
25         <Grid Height="52" VerticalAlignment="Bottom" RequestedTheme="Dark"
26             ↪ Background="Black">
27             <Button x:Name="button" Content="Delete" HorizontalAlignment="Right"
28                 ↪ Margin="0,0,10,10" VerticalAlignment="Bottom"
29                 ↪ Click="button_Click" RequestedTheme="Dark"/>
30             <ComboBox x:Name="comboBox" HorizontalAlignment="Right"
31                 ↪ Margin="0,0,78,10" VerticalAlignment="Bottom" Width="200"
32                 ↪ RequestedTheme="Dark"/>
33             <TextBlock x:Name="textBlock1" HorizontalAlignment="Right"
34                 ↪ Margin="0,0,283,16" TextWrapping="Wrap" Text="Recorded WiFi :"
35                 ↪ VerticalAlignment="Bottom" RequestedTheme="Dark"/>
36         </Grid>
37     </Grid>
38 </Page>

```

WiFiWebAutoLogin/MainPage.xaml.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.IO;
4 using System.Linq;
5 using System.Runtime.InteropServices.WindowsRuntime;
6 using System.Threading.Tasks;
7 using Windows.Foundation;

```

```
8 using Windows.Foundation.Collections;
9 using Windows.UI.Xaml;
10 using Windows.UI.Xaml.Controls;
11 using Windows.UI.Xaml.Controls.Primitives;
12 using Windows.UI.Xaml.Data;
13 using Windows.UI.Xaml.Input;
14 using Windows.UI.Xaml.Media;
15 using Windows.UI.Xaml.Navigation;
16 using Windows.Storage;
17 using System.Reflection;
18 using WiFiWebAutoLogin.RuntimeComponents;
19 using Windows.UI.ViewManagement;
20 using WiFiWebAutoLogin.Classes;
21 using System.Diagnostics;
22 using Windows.System.Threading;
23
24 // The Blank Page item template is documented at
25 ↪ http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409
26 namespace WiFiWebAutoLogin
27 {
28     /// <summary>
29     /// An empty page that can be used on its own or navigated to within a Frame.
30     /// </summary>
31     public sealed partial class MainPage : Page
32     {
33         private CaptivePortalDetector cpd = null;
34         private ThreadPoolTimer timeoutTimer = null;
35         private bool loaded = true;
36
37         public MainPage()
38         {
39             this.InitializeComponent();
40             ApplicationView.GetForCurrentView().SetPreferredMinSize(new Size {
41                 ↪ Width = 600, Height = 150 });
42             ApplicationView.PreferredLaunchViewSize = new Size(600, 150);
43             ApplicationView.PreferredLaunchWindowingMode =
44                 ↪ ApplicationViewWindowingMode.PreferredLaunchViewSize;
45             MainWebView.Margin = new Thickness(0, int.MaxValue, 0, int.MinValue);
46             textBlock.Text = "Initializing...";
47             this.setup();
48         }
49     }
50 }
```



```

47
48     private async void setup() {
49         cpd = await CaptivePortalDetector.GetInstance();
50         cpd.setup(MainWebView, textBlock, comboBox);
51         Debug.WriteLine("TEST SETUP");
52     }
53
54     private void MainWebView_LoadCompleted(object sender,
55     ↪ NavigationEventArgs e) {
56         if (cpd != null) {
57             this.loaded = true;
58             cpd.onLoad();
59         }
60     }
61
62     private void MainWebView_NavigationStarting(Webview sender,
63     ↪ WebViewNavigationStartingEventArgs args) {
64         Debug.WriteLine(args.Uri);
65         if (cpd != null) {
66             this.cpd.navigationStarting();
67
68             if (this.timeoutTimer!=null) {
69                 this.timeoutTimer.Cancel();
70                 this.timeoutTimer = null;
71             }
72
73             ScriptNotifyHandler scriptNotify = new ScriptNotifyHandler();
74             MainWebView.AddWebAllowedObject("ScriptNotifyHandler",
75             ↪ scriptNotify);
76
77             // Handle Timeout
78             this.loaded = false;
79             this.timeoutTimer = ThreadPoolTimer.CreateTimer(async (source)
80             ↪ => {
81                 await Windows.ApplicationModel.Core.CoreApplication.MainView
82                 ↪ .CoreWindow.Dispatcher.RunAsync(Windows.UI.Core.CoreDisp
83                 ↪ atcherPriority.Normal, () =>
84                 ↪ {
85                     this.timeoutTimer = null;
86                     if (!this.loaded) {
87                         this.cpd.timeout();
88                     }
89                 }

```

```
82         });
83     }, TimeSpan.FromSeconds(20));
84 }
85 }
86
87 private void MainWebView_NewWindowRequested(WebView sender,
88     ↪ WebViewNewWindowRequestedEventArgs args) {
89     args.Handled = true;
90     cpd.queueUri(args.Uri);
91 }
92
93 private async void MainWebView_NavigationCompleted(WebView sender,
94     ↪ WebViewNavigationCompletedEventArgs args) {
95     if (cpd != null) {
96         await sender.InvokeScriptAsync("eval", new string[] {
97             "window.open =
98             ↪ function(url){ScriptNotifyHandler.windowOpen(url)};" +
99             "var open = window.open;" +
100             "document.open = window.open;"
101         });
102     }
103 }
104
105 private void button_Click(object sender, RoutedEventArgs e) {
106     cpd.removeLoginInformation((string)comboBox.SelectedItem);
107 }
108
109 private void MainWebView_DOMContentLoaded(WebView sender,
110     ↪ WebViewDOMContentLoadedEventArgs args) {
111     // DISABLED
112 }
113
114 private void MainWebView_ContentLoading(WebView sender,
115     ↪ WebViewContentLoadingEventArgs args) {
116     // DISABLED
117 }
118
119 private void MainWebView_LongRunningScriptDetected(WebView sender,
120     ↪ WebViewLongRunningScriptDetectedEventArgs args) {
121     // DISABLED
122 }
```

```

118     private void MainWebView_PermissionRequested(Webview sender,
119         ↪ WebViewPermissionRequestedEventArgs args) {
120         // DISABLED
121     }
122 }

```

```

----- WiFiWebAutoLogin/JavaScript/DeployListeners.js -----
1  var inputs = document.getElementsByTagName("input");
2  var ahrefs = document.getElementsByTagName("a");
3  var buttons = document.getElementsByTagName("button");
4
5  function addClickListener(el, tagName, idx) {
6      el.addEventListener("click", function () {
7          ScriptNotifyHandler.passAction("document.getElementsByTagName(\"" +
8          ↪ tagName + "\")[\" + idx + \"].click();");
9      });
10 }
11 function addChangeListener(el, tagName, idx) {
12     el.addEventListener("change", function () {
13         ScriptNotifyHandler.passAction("document.getElementsByTagName(\"" +
14         ↪ tagName + "\")[\" + idx + \"].value = \" +
15         ↪ JSON.stringify(document.getElementsByTagName(tagName)[idx].value) +
16         ↪ \";");
17     });
18 }
19
20 for (var i = 0; i < inputs.length; i++) {
21     addClickListener(inputs[i], "input", i);
22     addChangeListener(inputs[i], "input", i);
23 }
24
25 for (var i = 0; i < ahrefs.length; i++) {
26     addClickListener(ahrefs[i], "a", i);
27 }
28
29 for (var i = 0; i < buttons.length; i++) {
30     addClickListener(buttons[i], "button", i);
31 }

```

A.2 Namespace: WiFiWebAutoLogin.Classes

WiFiWebAutoLogin.Classes/ActionSequence.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Runtime.Serialization;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace WiFiWebAutoLogin.Classes {
9     [DataContract]
10    class ActionSequence {
11        [DataMember]
12        private LinkedList<string> actions;
13
14        public ActionSequence() {
15            this.actions = new LinkedList<string>();
16        }
17
18        public void add(string action) {
19            this.actions.AddLast(action);
20        }
21
22        public IEnumerable<string> getEnumerable() {
23            return this.actions.AsEnumerable();
24        }
25
26        public void reset() {
27            this.actions.Clear();
28        }
29    }
30 }
```

WiFiWebAutoLogin.Classes/CaptivePortalDetector.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Diagnostics;
4 using System.IO;
5 using System.Linq;
6 using System.Net;
7 using System.Net.Http;
8 using System.Net.NetworkInformation;
```

```
9 using System.Runtime.Serialization.Json;
10 using System.Text;
11 using System.Text.RegularExpressions;
12 using System.Threading;
13 using System.Threading.Tasks;
14 using Windows.Foundation;
15 using Windows.Foundation.Metadata;
16 using Windows.Networking.Connectivity;
17 using Windows.Storage;
18 using Windows.System.Threading;
19 using Windows.UI.ViewManagement;
20 using Windows.UI.Xaml;
21 using Windows.UI.Xaml.Controls;
22
23 namespace WiFiWebAutoLogin.Classes {
24     public class CaptivePortalDetector {
25         private static CaptivePortalDetector instance = null;
26         private Storage storage;
27         private string ssid;
28         private Queue<Uri> uriQueue;
29
30         private WebView webView;
31         private TextBlock textBlock;
32         private ComboBox comboBox;
33
34         private string currentFingerprint;
35         private ActionSequence currentActionSequence;
36         private ThreadPoolTimer timer;
37
38         private CaptivePortalDetector() {
39             this.webView = null;
40             this.storage = new Storage("credentials.dat");
41             this.uriQueue = new Queue<Uri>();
42         }
43
44         public bool isSetup() {
45             return this.webView != null;
46         }
47
48         public void setup(WebView webView, TextBlock textBlock, ComboBox
49             ↪ comboBox) {
50             this.webView = webView;
```

```
50         this.textBlock = textBlock;
51         this.comboBox = comboBox;
52
53         this.refreshList();
54
55         this.updateSSID();
56         this.updateWebView();
57     }
58
59     public void refreshList() {
60         comboBox.ItemsSource = this.storage.getLoginInfo().getList();
61     }
62
63     public WebView getWebView() {
64         return this.webView;
65     }
66
67     public void updateWebView() {
68         if (this.ssid != null) {
69             // CONNECTED
70             this.webView.Navigate(new Uri(Config.uri));
71         }
72         else {
73             // DISCONNECTED
74             this.displayMessage("Check your network connection.");
75         }
76     }
77
78     public static async Task<CaptivePortalDetector> getInstance() {
79         if (instance==null) {
80             instance = new CaptivePortalDetector();
81             await instance.storage.setup();
82         }
83         return instance;
84     }
85
86     public async void onLoad() {
87         if (this.ssid!=null) {
88             // GET FINGERPRINT
89             this.currentFingerprint = await this.getFingerprint();
90             string body = await this.getBody();
91         }
```

```

92         this.currentActionSequence = this.storage.getLoginInfo().getActi_
          ↪ onSequence(this.currentFingerprint);
93         bool hasActionSequence = true;
94         if (this.currentActionSequence == null) {
95             hasActionSequence = false;
96             this.currentActionSequence = new ActionSequence();
97             this.storage.getLoginInfo().addActionSequence(this.currentFi_
          ↪ ngerprint,
          ↪ this.currentActionSequence);
98             this.storage.saveData();
99             this.refreshList();
100         }
101
102         if (!body.Trim().Equals("connected")) {
103             // Not Connected
104
105             if (hasActionSequence) {
106                 this.displayMessage("Executing recorded
          ↪ actions...\r\n\r\n" + "(" +
          ↪ this.currentFingerprint.Split(new string[] {
          ↪ Conf.separator },
          ↪ StringSplitOptions.RemoveEmptyEntries)[1] + ")");
107             }
108
109             IEnumerable<string> actions =
          ↪ this.currentActionSequence.getEnumerable();
110             string compiledActions = "";
111             foreach (string action in actions) {
112                 compiledActions += action;
113             }
114             await this.webView.InvokeScriptAsync("eval", new string[] {
          ↪ compiledActions });
115
116             // Deploy Listeners
117             this.deployListeners();
118             if (this.uriQueue.Count > 0) {
119                 this.startTimer();
120             }
121
122             if (!hasActionSequence) {
123                 this.displayWebView();
124             }

```

```

125         else {
126             this.startRetryTimer(this.currentFingerprint);
127         }
128     }
129     else {
130         // Connected
131         this.displayMessage("Connected.");
132         this.uriQueue.Clear();
133     }
134 }
135 }
136
137 public void navigationStarting() {
138     if (this.timer!=null) {
139         this.timer.Cancel();
140         this.timer = null;
141     }
142 }
143
144 private void startRetryTimer(string oldFingerprint) {
145     ThreadPoolTimer.CreateTimer(async (source) => {
146         await Windows.ApplicationModel.Core.CoreApplication.MainView.CoreWin_
147         ↪ eWindow.Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPr_
148         ↪ iority.Normal, () =>
149         ↪ {
150             if (this.currentFingerprint.Equals(oldFingerprint)) {
151                 this.displayWebView();
152             }
153         });
154     }, TimeSpan.FromSeconds(5));
155 }
156
157 private async void timerCallback() {
158     await Windows.ApplicationModel.Core.CoreApplication.MainView.CoreWin_
159     ↪ dow.Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.N_
160     ↪ ormal, () =>
161     ↪ {
162         if (this.timer!=null) {
163             this.timer = null;
164         }
165         this.dequeueUri();
166     });

```



```
161     }
162
163     private async void deployListeners() {
164         StorageFolder InstallationFolder =
165             ↪ Windows.ApplicationModel.Package.Current.InstalledLocation;
166         StorageFile file = await InstallationFolder.GetFilesAsync(@"JavaScript\
167             ↪ t\DeployListeners.js");
168         string js = await FileIO.ReadTextAsync(file);
169         await this.webView.InvokeScriptAsync("eval", new string[] { js });
170     }
171
172     private async Task<string> getFingerprint() {
173         string uri = "";
174         string title = "";
175         try {
176             uri = await this.webView.InvokeScriptAsync("eval", new string[]
177                 ↪ { "window.location.href;" });
178             title = await this.webView.InvokeScriptAsync("eval", new
179                 ↪ string[] { "document.getElementsByTagName(\"title\")[0].inne
180                 ↪ rHTML.trim();"
181                 ↪ });
182         } catch (Exception e) {
183         }
184         return this.ssid + Conf.separator + uri + Conf.separator + title;
185     }
186
187     private async Task<string> getUri() {
188         string uri = "";
189         try {
190             uri = await this.webView.InvokeScriptAsync("eval", new string[]
191                 ↪ { "window.location.href;" });
192         }
193         catch (Exception e) {
194         }
195         return uri;
196     }
197
198     private async Task<string> getBody() {
199         return await this.webView.InvokeScriptAsync("eval", new string[] {
200             ↪ "document.body.innerHTML;" });
201     }
```

```

195     private async Task<string> getScripts() {
196         return await this.webView.InvokeScriptAsync("eval", new string[] {
197             ↪ "document.body.innerHTML;" });
198     }
199
200     public void passAction(string args) {
201         if (this.currentActionSequence!=null) {
202             this.currentActionSequence.add(args);
203             this.storage.saveData();
204         }
205     }
206
207     public void updateSSID() {
208         ConnectionProfile connectionProfile =
209             ↪ NetworkInformation.GetInternetConnectionProfile();
210
211         string data = "";
212         if (connectionProfile != null) {
213             Debug.WriteLine("[NETWORK]: "+connectionProfile.GetNetworkConnec_
214                 ↪ tivityLevel().ToString());
215
216             IEnumerable<string> enumerable =
217                 ↪ connectionProfile.GetNetworkNames().AsEnumerable();
218             foreach (string v in enumerable) {
219                 if (data.Equals("")) {
220                     data += v;
221                 }
222                 else {
223                     data += " | " + v;
224                 }
225             }
226             if (data.Equals("")) {
227                 this.ssid = null;
228             }
229             else {
230                 this.ssid = data;
231                 Debug.WriteLine("[SSID]: " + data);
232             }
233         }
234         else {
235             this.ssid = null;
236         }
237     }

```

```
233     }
234
235     public string getSSID() {
236         return this.ssid;
237     }
238
239     private void dequeueUri() {
240         Uri uri;
241         try {
242             uri = this.uriQueue.Dequeue();
243         } catch (Exception e) {
244             uri = null;
245         }
246
247         if (uri!=null) {
248
249             this.webView.Navigate(uri);
250         }
251     }
252
253     private async Task<string> EscapeJSONString(string unescaped) {
254         DataContractJsonSerializer serializer = new
255             ↪ DataContractJsonSerializer(typeof(LoginInformation));
256         MemoryStream stream = new MemoryStream(); ;
257         serializer.WriteObject(stream, unescaped);
258         stream.Position = 0;
259         return await (new StreamReader(stream)).ReadToEndAsync();
260     }
261
262     public void queueUri(Uri uri) {
263         this.uriQueue.Enqueue(uri);
264         if (this.uriQueue.Count == 1) {
265             this.startTimer();
266         }
267     }
268
269     private void startTimer() {
270         this.timer = ThreadPoolTimer.CreateTimer((source) => {
271             this.timerCallback();
272         }, TimeSpan.FromSeconds(1));
273     }
```

```

274     private void displayMessage(string message) {
275         ApplicationView.GetForCurrentView().TryResizeView(new Size { Width =
           ↳ 600, Height = 150 });
276         this.textBlock.Text = message;
277         this.webView.Margin = new Thickness(0, int.MaxValue, 0,
           ↳ int.MinValue);
278     }
279
280     private void displayWebView() {
281         ApplicationView.GetForCurrentView().TryResizeView(new Size { Width =
           ↳ 800, Height = 500 });
282         this.textBlock.Text = "";
283         this.webView.Margin = new Thickness(0, 0, 0, 0);
284     }
285
286     public void timeout() {
287         this.displayMessage("Operation timeout.\r\nCheck your network
           ↳ connection.");
288     }
289
290     public void removeLoginInformation(string ssid) {
291         if (ssid != null) {
292             this.storage.getLoginInfo().removeBySSID(ssid);
293             this.storage.saveData();
294             this.refreshList();
295         }
296     }
297 }
298 }

```

WiFiWebAutoLogin.Classes/Conf.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace WiFiWebAutoLogin.Classes {
8      class Conf {
9          public static readonly string uri =
           ↳ "http://107.172.253.111/network_status.html";
10         public static readonly string resource = "WiFiWebAutoLogin";

```

```

11         public static readonly string username = "userWiFiWebAutoLogin";
12         public static readonly string separator = "<.:>";
13     }
14 }

```

```

_____ WiFiWebAutoLogin.Classes/LoginInformation.cs _____
1  using System;
2  using System.Collections.Generic;
3  using System.Collections.ObjectModel;
4  using System.Linq;
5  using System.Runtime.Serialization;
6  using System.Text;
7  using System.Threading.Tasks;
8  using Windows.ApplicationModel.Contacts;
9
10 namespace WiFiWebAutoLogin.Classes {
11     [DataContract]
12     class LoginInformation {
13         [DataMember]
14         private Dictionary<string, ActionSequence> actionSequences;
15
16         public LoginInformation() {
17             this.actionSequences = new Dictionary<string, ActionSequence>();
18             ActionSequence actionSequence = new ActionSequence();
19         }
20
21         public ActionSequence getActionSequence(string fingerprint) {
22             try {
23                 return this.actionSequences[fingerprint];
24             } catch (Exception e) {
25                 return null;
26             }
27         }
28
29         public void addActionSequence(string fingerprint, ActionSequence
↵ actionSequence) {
30             this.actionSequences.Add(fingerprint, actionSequence);
31         }
32
33         public List<string> getList() {
34             Dictionary<string, ActionSequence>.KeyCollection.Enumerator
↵ loginInfoEnumerator = actionSequences.Keys.GetEnumerator();

```

```

35     List<string> list = new List<string>();
36     while (loginInfoEnumerator.MoveNext()) {
37         string ssid = loginInfoEnumerator.Current.Split(new string[] {
38             ↪ Conf.separator }, StringSplitOptions.RemoveEmptyEntries)[0];
39         if (!list.Contains(ssid)) {
40             list.Add(ssid);
41         }
42     }
43     return list;
44 }
45
46 public void removeBySSID(string ssid) {
47     Dictionary<string, ActionSequence>.KeyCollection.Enumerator
48     ↪ loginInfoEnumerator = actionSequences.Keys.GetEnumerator();
49     List<string> removalList = new List<string>();
50     while (loginInfoEnumerator.MoveNext()) {
51         string enumSSID = loginInfoEnumerator.Current.Split(new string[]
52             ↪ { Conf.separator },
53             ↪ StringSplitOptions.RemoveEmptyEntries)[0];
54         if (enumSSID.Equals(ssid)) {
55             removalList.Add(loginInfoEnumerator.Current);
56         }
57     }
58     List<string>.Enumerator removalListEnumerator =
59     ↪ removalList.GetEnumerator();
60     while (removalListEnumerator.MoveNext()) {
61         actionSequences.Remove(removalListEnumerator.Current);
62     }
63 }
64 }
65 }
66 }

```

WiFiWebAutoLogin.Classes/Storage.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.IO;
4 using System.Linq;
5 using System.Runtime.Serialization.Json;
6 using System.Text;
7 using System.Threading.Tasks;
8 using System.Xml.Serialization;
9 using Windows.Security.Credentials;

```

```
10 using Windows.Security.Cryptography;
11 using Windows.Security.Cryptography.Core;
12 using Windows.Storage;
13 using Windows.Storage.Streams;
14
15 namespace WiFiWebAutoLogin.Classes {
16     class Storage {
17         private string fileName;
18         private string password;
19         private LoginInformation loginInfo;
20
21         public Storage(string fileName) {
22             this.fileName = new String(fileName.ToCharArray());
23             PasswordVault vault = new PasswordVault();
24
25             try {
26                 this.password = vault.Retrieve(Config.resource,
27                     ↪ Config.username).Password;
28             }
29             catch (Exception e) {
30                 vault.Add(new PasswordCredential(Config.resource, Config.username,
31                     ↪ CryptographicBuffer.EncodeToBase64String(CryptographicBuffer.
32                     ↪ .GenerateRandom(64))));
33                 this.password = vault.Retrieve(Config.resource,
34                     ↪ Config.username).Password;
35             }
36         }
37
38         public async Task setup() {
39             StorageFolder folder = ApplicationData.Current.LocalFolder;
40             StorageFile file;
41             try {
42                 file = await folder.GetFilesAsync(this.fileName);
43             } catch (Exception e) {
44                 file = await folder.CreateFileAsync(this.fileName);
45             }
46
47             IBuffer encryptedJson = await FileIO.ReadBufferAsync(file);
48             SymmetricKeyAlgorithmProvider algorithmProvider = SymmetricKeyAlgori
49                 ↪ thmProvider.OpenAlgorithm(SymmetricAlgorithmNames.AesCbcPkcs7);
```

```

45     IBuffer bufferedPassword =
        ↳ CryptographicBuffer.ConvertStringToBinary(password,
        ↳ BinaryStringEncoding.Utf8);
46     IBuffer decryptedJson = CryptographicEngine.Decrypt(algorithmProvide_
        ↳ r.CreateSymmetricKey(bufferedPassword), encryptedJson,
        ↳ bufferedPassword);
47     DataReader dataReader =
        ↳ Windows.Storage.Streams.DataReader.FromBuffer(decryptedJson);
48     string json = dataReader.ReadString(decryptedJson.Length);
49
50     if (json.Trim().Equals("")) {
51         loginInfo = new LoginInformation();
52         this.saveData();
53     }
54     else {
55         DataContractJsonSerializer serializer = new
        ↳ DataContractJsonSerializer(typeof(LoginInformation));
56         loginInfo = (LoginInformation)serializer.ReadObject(new
        ↳ MemoryStream(Encoding.Unicode.GetBytes(json)));
57     }
58 }
59
60 public LoginInformation getLoginInfo() {
61     return this.loginInfo;
62 }
63
64 public async void saveData() {
65     DataContractJsonSerializer serializer = new
        ↳ DataContractJsonSerializer(typeof(LoginInformation));
66     MemoryStream stream = new MemoryStream();
67     serializer.WriteObject(stream, loginInfo);
68     stream.Position = 0;
69     StreamReader sr = new StreamReader(stream);
70     string data = sr.ReadToEnd();
71
72     SymmetricKeyAlgorithmProvider algorithmProvider = SymmetricKeyAlgori_
        ↳ thmProvider.OpenAlgorithm(SymmetricAlgorithmNames.AesCbcPkcs7);
73     IBuffer keyMaterial =
        ↳ CryptographicBuffer.ConvertStringToBinary(password,
        ↳ BinaryStringEncoding.Utf8);
74     IBuffer bufferedData = CryptographicBuffer.CreateFromByteArray(Encod_
        ↳ ing.UTF8.GetBytes(data));

```



```
75     CryptographicKey key =  
76         ↪ algorithmProvider.CreateSymmetricKey(keyMaterial);  
77     IBuffer encryptedData = CryptographicEngine.Encrypt(key,  
78         ↪ bufferedData, keyMaterial);  
79  
80     StorageFolder folder = ApplicationData.Current.LocalFolder;  
81     StorageFile file;  
82     try {  
83         file = await folder.GetFilesAsync(this.fileName);  
84     }  
85     catch (Exception e) {  
86         file = await folder.CreateFileAsync(this.fileName);  
87     }  
88     await FileIO.WriteBufferAsync(file, encryptedData);  
89 }  
90 }
```

A.3 Namespace: WiFiWebAutoLogin.RuntimeComponents

```
WiFiWebAutoLogin.RuntimeComponents/NetChangeDetectorBackgroundTask.cs  
1 using System;  
2 using System.Collections.Generic;  
3 using System.Diagnostics;  
4 using System.Linq;  
5 using System.Text;  
6 using System.Threading.Tasks;  
7 using Windows.ApplicationModel.Background;  
8 using Windows.Networking.Connectivity;  
9 using WiFiWebAutoLogin.Classes;  
10 using Windows.UI.Notifications;  
11 using Windows.Data.Xml.Dom;  
12 using System.IO;  
13  
14 namespace WiFiWebAutoLogin.RuntimeComponents {  
15     public sealed class NetChangeDetectorBackgroundTask : IBackgroundTask {  
16         private static string lastSSID = "";  
17         private static Boolean lastConnectionChanged = false;  
18  
19         public void Run(IBackgroundTaskInstance taskInstance) {
```

```

20     var mDeferral = taskInstance.GetDeferral();
21
22     Debug.WriteLine("Result:");
23     Debug.WriteLine(this.connectionChanged());
24     Debug.WriteLine(lastSSID != null);
25     Debug.WriteLine(this.hasNoInternetAccess());
26     if ((lastConnectionChanged || this.connectionChanged()) &&
    ↪ lastSSID!=null && this.hasNoInternetAccess()) {
27
28         string xmlText = "<?xml version=\"1.0\" encoding=\"utf-8\" ?>" +
29             "<toast launch=\"app-defined-string\">" +
30                 "<visual>" +
31                     "<binding template=\"ToastGeneric\">" +
32                         "<text>Network Detected</text>" +
33                         "<text>Would you like to run
    ↪ WiFiWebAutoLogin?</text>" +
34                     "</binding>" +
35                 "</visual>" +
36                 "<actions>" +
37                     "<action content=\"Yes\" arguments=\"Yes\" />" +
38                     "<action content=\"No\" arguments=\"No\"
    ↪ activationType=\"background\" />" +
39                 "</actions>" +
40                 "<audio
    ↪ src=\"ms-winsoundevent:Notification.Reminder\"/>" +
41             "</toast>";
42
43     XmlDocument xmlContent = new XmlDocument();
44     xmlContent.LoadXml(xmlText);
45
46     ToastNotification notification = new
    ↪ ToastNotification(xmlContent);
47     notification.Tag = "WWAL_TOAST";
48     notification.Dismissed += (ToastNotification n,
    ↪ ToastDismissedEventArgs args) => {
49         ToastNotificationManager.History.Remove("WWAL_TOAST");
50     };
51     ToastNotificationManager.CreateToastNotifier().Show(notification
    ↪ );
52 }
53
54 mDeferral.Complete();

```

```
55     }
56
57     private bool hasNoInternetAccess() {
58         ConnectionProfile connectionProfile =
59             ↪ NetworkInformation.GetInternetConnectionProfile();
60
61         if (connectionProfile != null) {
62             if (connectionProfile.GetNetworkConnectivityLevel().ToString().Trim().Equals("InternetAccess"))
63                 ↪ {
64                     return false;
65                 }
66         }
67
68         return true;
69     }
70
71     private void testNotification() {
72         string xmlText = "<?xml version=\"1.0\" encoding=\"utf-8\" ?>" +
73             "<toast launch=\"app-defined-string\">" +
74             "<visual>" +
75                 "<binding template=\"ToastGeneric\">" +
76                     "<text>WiFiWebAutoLogin</text>" +
77                     "</binding>" +
78                 "</visual>" +
79                 "<audio"
80                 ↪ src=\"ms-winsoundevent:Notification.Reminder\"/>" +
81             "</toast>";
82
83         XmlDocument xmlContent = new XmlDocument();
84         xmlContent.LoadXml(xmlText);
85
86         ToastNotification notification = new ToastNotification(xmlContent);
87         notification.Tag = "WWAL_TOAST_TEST";
88         notification.Dismissed += (ToastNotification n,
89             ↪ ToastDismissedEventArgs args) => {
90             ToastNotificationManager.History.Remove("WWAL_TOAST_TEST");
91         };
92         ToastNotificationManager.CreateToastNotifier().Show(notification);
93     }
94
95     private bool connectionChanged() {
```

```
92     string ssid;
93     ConnectionProfile connectionProfile =
94         ↪ NetworkInformation.GetInternetConnectionProfile();
95     string data = "";
96     if (connectionProfile != null) {
97         IEnumerable<string> enumerable =
98             ↪ connectionProfile.GetNetworkNames().AsEnumerable();
99         foreach (string v in enumerable) {
100             if (data.Equals("")) {
101                 data += v;
102             }
103             else {
104                 data += " | " + v;
105             }
106             if (data.Equals("")) {
107                 ssid = null;
108             }
109             else {
110                 ssid = data;
111             }
112         }
113     }
114     else {
115         ssid = null;
116     }
117
118     if (lastSSID != null) {
119         if (lastSSID.Equals(ssid)) {
120             lastSSID = ssid;
121             lastConnectionChanged = false;
122             return false;
123         }
124         else {
125             lastSSID = ssid;
126             lastConnectionChanged = true;
127             return true;
128         }
129     }
130     else {
131         if (ssid==null) {
132             lastSSID = ssid;
133             lastConnectionChanged = false;
```

```
132         return false;
133     }
134     else {
135         lastSSID = ssid;
136         lastConnectionChanged = true;
137         return true;
138     }
139 }
140 }
141 }
142 }
```

WiFiWebAutoLogin.RuntimeComponents/ScriptNotifyHandler.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using Windows.Foundation.Metadata;
7 using WiFiWebAutoLogin.Classes;
8 using System.Diagnostics;
9
10 namespace WiFiWebAutoLogin.RuntimeComponents
11 {
12     [AllowForWeb]
13     public sealed class ScriptNotifyHandler
14     {
15         public async void passAction(string args) {
16             CaptivePortalDetector cpd = await
17                 ↪ CaptivePortalDetector.GetInstance();
18             cpd.passAction(args);
19         }
20
21         public async void windowOpen(string args) {
22             CaptivePortalDetector cpd = await
23                 ↪ CaptivePortalDetector.GetInstance();
24             cpd.queueUri(new Uri(args));
25             Debug.WriteLine(args);
26         }
27
28         public void testDebug(string args) {
29             Debug.WriteLine(args);
30         }
31     }
32 }
```

28	}
29	}
30	}