SKRIPSI

PERANGKAT LUNAK LOGIN OTOMATIS UNTUK *CAPTIVE PORTAL* WI-FI

YOHANES MARIO CHANDRA

NPM: 2011730031

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2017

UNDERGRADUATE THESIS

AUTOMATED LOGIN SOFTWARE FOR WI-FI CAPTIVE PORTAL

YOHANES MARIO CHANDRA

NPM: 2011730031

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2017

ABSTRAK

Captive portal yang banyak digunakan pada Wi-Fi publik membutuhkan login berulang-ulang setiap kali pengguna ingin menggunakan Wi-Fi tersebut. Oleh karena itu, dibutuhkan otomasi untuk mempermudah penggunaan Wi-Fi 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 Wi-Fi require repeated login every time a user wants to use the Wi-Fi. Therefore, it requires automation to facilitate the use of Wi-Fi 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

DAFTAR ISI

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 Wi-Fi. Wi-Fi 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, 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.

Berdasarkan pengamatan peneliti, *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.

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 Wi-Fi 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 Wi-Fi 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 Wi-Fi berbasis web.

4. Bab Perancangan
   Bab ini berisi perancangan perangkat lunak login otomatis untuk proteksi Wi-Fi berbasis web.

5. Bab Implementasi dan Pengujian
   Bab ini berisi implementasi perangkat lunak login otomatis untuk proteksi Wi-Fi 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[1]. 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, danatau 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*[2]. 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.

Kode Status HTTP 511

Kode status HTTP adalah bilangan bulat positif yang terdiri dari 3 digit[3]. 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*.

[1] Alat yang meneruskan paket data ke bagian dari jaringan yang dituju.
[2] Alat yang digunakan untuk menghubungkan dua jaringan yang berbeda, biasanya berupa hubungan ke internet.

- 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[4]. 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 sebagai berikut:

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

Saat menerima *request* tersebut, server login akan mengirimkan respon dengan kode status 511:

```
HTTP/1.1 511 Network Authentication Required
Content-Type: text/html

<html>
    <head>
        <title>Network Authentication Required</title>
        <meta http-equiv="refresh"
            content="0; url=https://login.example.net/">
    </head>
    <body>
        <p>You need to <a href="https://login.example.net/">
        authenticate with the local network</a> in order to gain
        access.</p>
    </body>
</html>
```

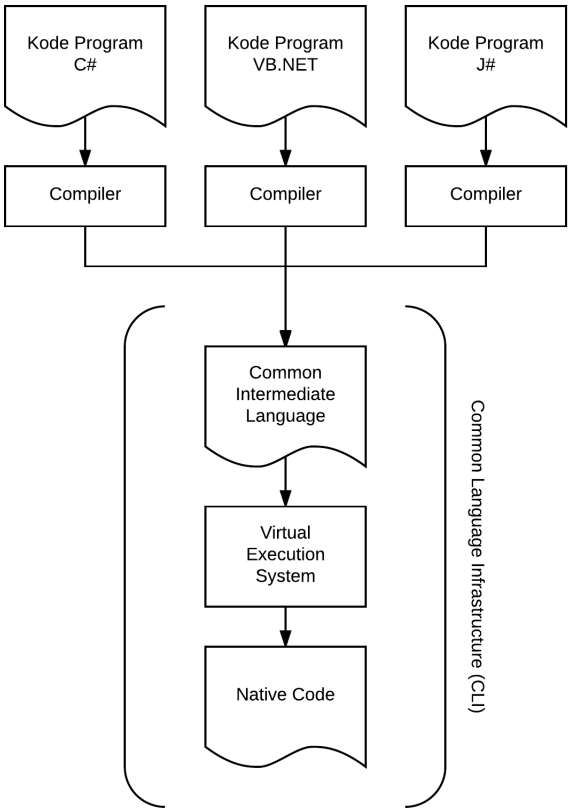
Respon ini memungkinkan klien untuk mendeteksi bahwa respon tersebut bukan berasal dari server tujuan. Selain itu, elemen meta pada HTML yang disajikan memungkinkan klien untuk melakukan login pada *link* yang diberikan.

2.2 *.NET Framework*

.NET adalah platform yang bersifat *general purpose* untuk membangun aplikasi[5]. .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).

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 Inrastructure* (CLI).



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

Komponen-komponen penting dalam CLI di antaranya adalah[6]:

- *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 Spesification* (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#[5]. 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[6]. Implementasi VES pada .NET bernama *Common Language Runtime* (CLR).

2.4 *Universal Windows Platform* (UWP)

Windows 8 memperkenalkan *Windows Runtime* (WinRT) yang merupakan arsitektur aplikasi umum untuk Windows[7]. 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[8]. WebView tidak mendukung masukkan pengguna seperti *key-down*, *key-up*, dan *pointer-pressed*. Oleh karena itu, dibutuhkan metode lain yang melibatkan InvokeScriptAsync dengan fungsi *eval* javascript untuk menggunakan HTML *event handler* dan fungsi window.external.notify untuk menangani event dari HTML pada aplikasi.

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[8]. 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[8]. 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

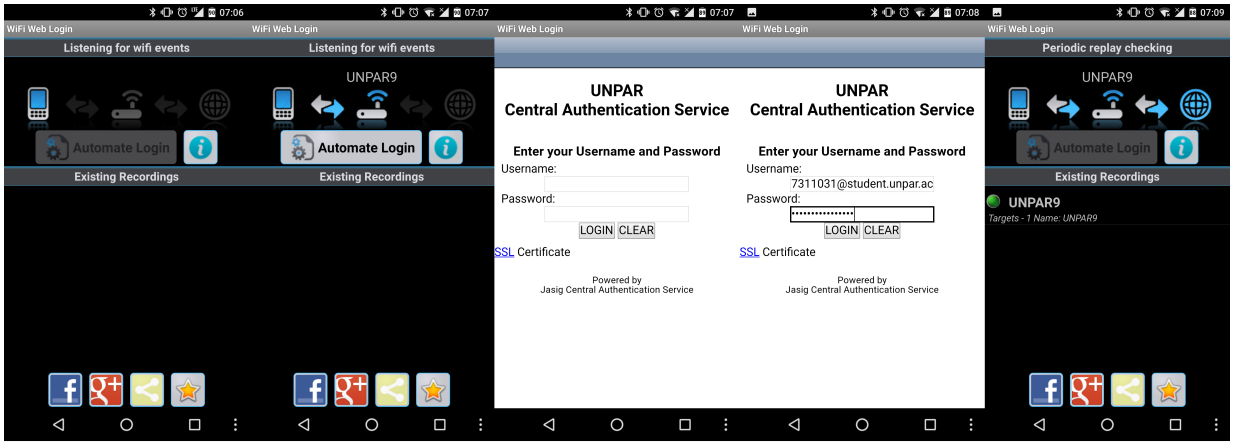
ANALISIS

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

3.1 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*



1. 2. 3 4. 5.

Gambar 3.1: Tampilan aplikasi *WiFi Web Login*.

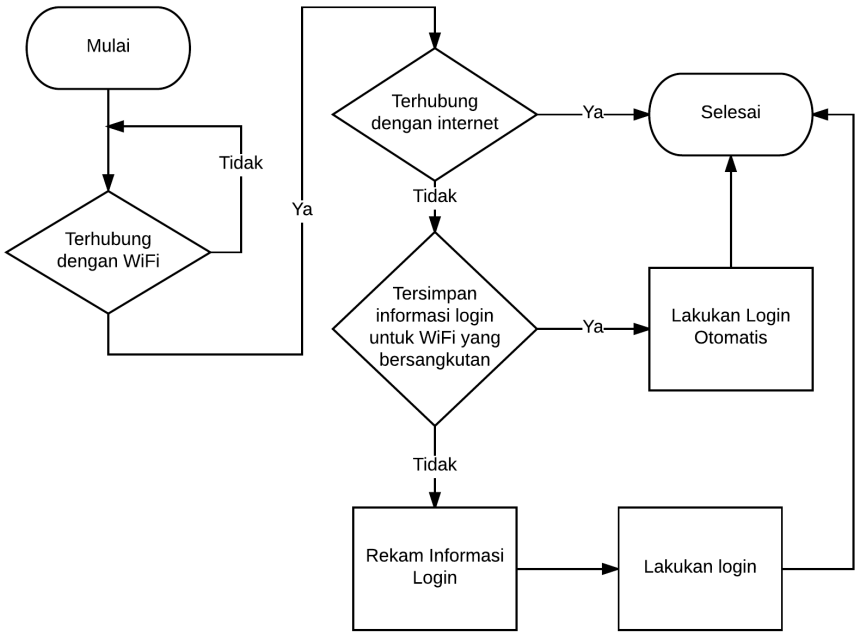
Pada gambar 3.1 diperlihatkan langkah-langkah login *captive portal* Unpar sebagai berikut:

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.

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

Diagram Alir *WiFi Web Login*

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



Gambar 3.2: Diagram alir proses yang perlu dilalui oleh aplikasi *WiFi Web Login*.

Berdasarkan gambar 3.2, 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.
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 melalui sudah pernah melakukan login pertama kali menggunakan aplikasi tersebut, maka aplikasi akan melakukan login otomatis setiap kali terhubung dengan wifi yang bersangkutan.

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

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

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

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

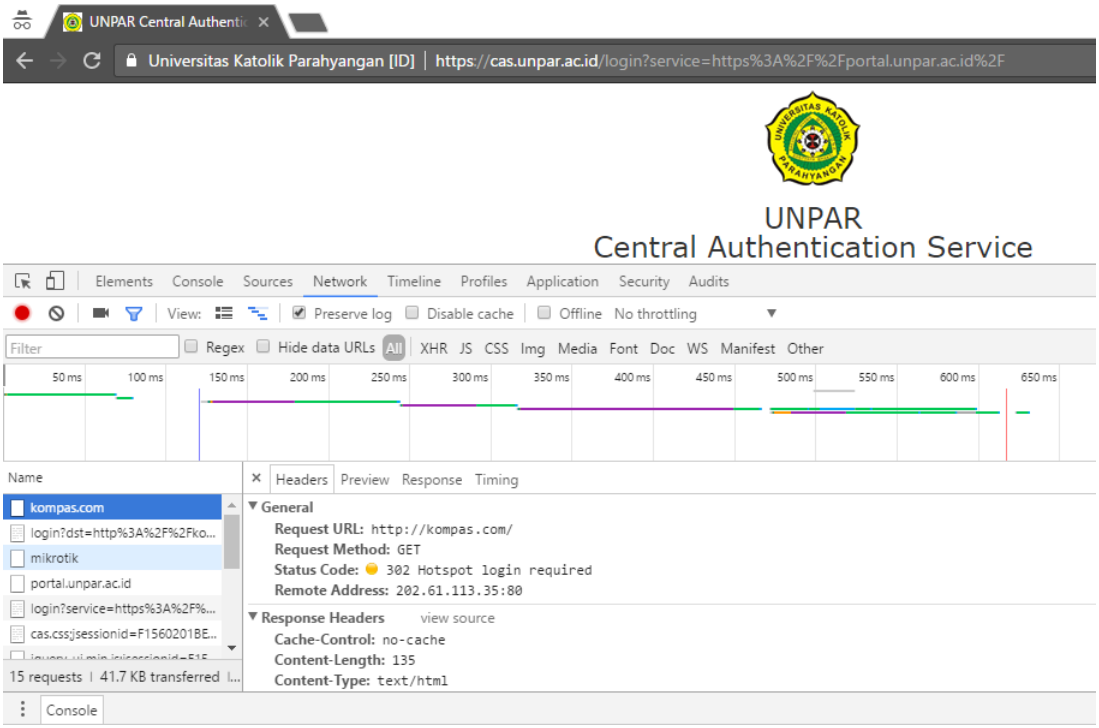
3.3 Analisis Metode Rekam dan Kirim Informasi Login

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[9]) 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.

3.4 Analisis Metode Deteksi *Captive Portal*



Gambar 3.3: *Screenshot* HTTP *header* yang dikirimkan oleh *captive portal* milik Unpar.

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 3.3 menunjukkan *captive portal* Unpar yang menggunakan kode status HTTP 302 untuk memberitahu klien bahwa diperlukan otentikasi jaringan. Berdasarkan teori mengenai kode 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 dibutuhkannya *redirection* yang dapat dikenali oleh *web browser* agar klien selalu diarahkan ke halaman *captive portal* tersebut sebelum melakukan otentikasi. Oleh karena itu, untuk setiap HTTP *request* yang dilakukan oleh klien sebelum melakukan otentikasi, HTTP *response* yang diberikan bukanlah berasal dari server tujuan. Sifat ini dapat dimaanfaatkan 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.

3.5 Analisis Perangkat Lunak

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

3.5.1 Analisis Kelas

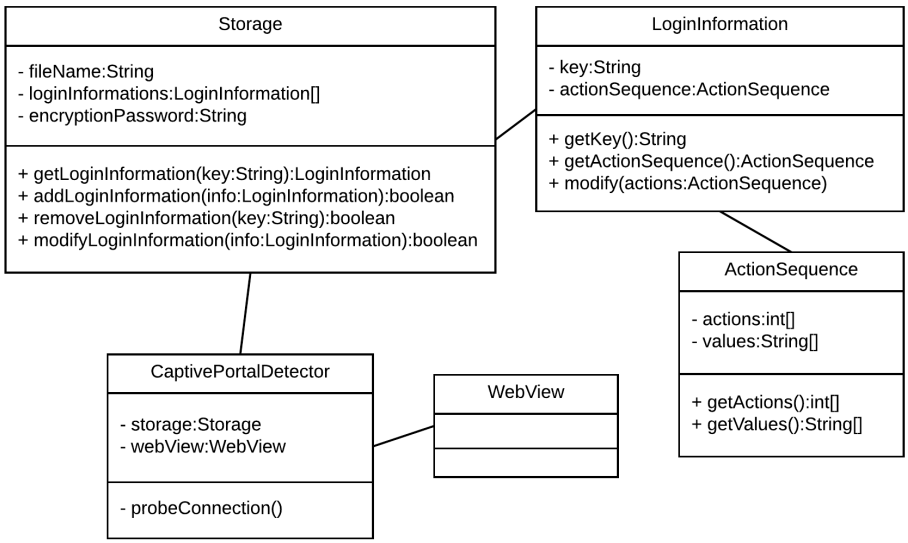
Diagram kelas untuk perangkat lunak yang dibangun dapat dilihat pada gambar 3.4. Seperti pada gambar 3.4, kelas-kelas yang dibutuhkan pada perangkat lunak ini adalah:

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

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

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

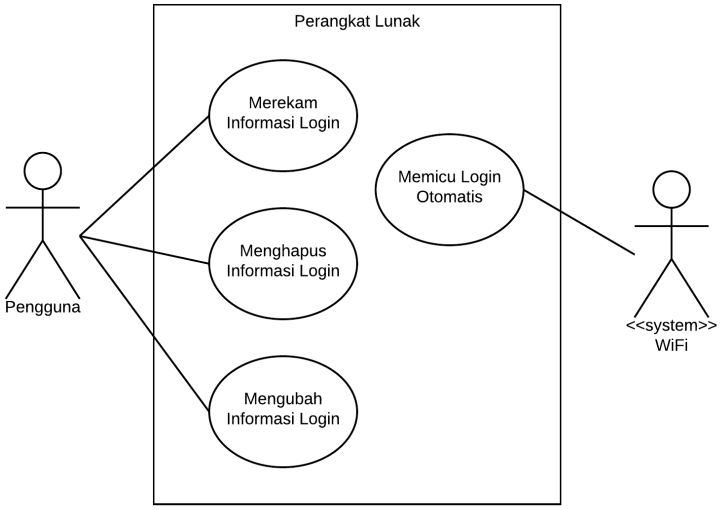
- ActionSequence
  Kelas ini digunakan untuk menyimpan langkah-langkah login dalam bentuk urutan aksi dan nilai-nilai yang berkaitan dengan aksi tersebut.



Gambar 3.4: Diagram kelas untuk perangkat lunak yang dibangun.

3.5.2 Analisis *Use Case*

Diagram *use case* untuk perangkat lunak yang dibangun dapat dilihat pada gambar 3.5.



Gambar 3.5: Diagram *use case* untuk perangkat lunak yang dibangun.

Skenario merekam informasi login

Nama: Merekam informasi login
Aktor: Pengguna
Kondisi awal: Perangkat lunak mendeteksi adanya *captive portal*.
Deskripsi: Pengguna menyimpan informasi login.
Kondisi Akhir: Informasi login tersimpan di dalam perangkat lunak.

Skenario:

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 mengubah informasi login

Nama: Mengubah informasi login
Aktor: Pengguna
Kondisi awal: Perangkat lunak sudah dijalankan.
Deskripsi: Pengguna mengubah informasi login.
Kondisi Akhir: Informasi login diubah dari perangkat lunak.

Skenario:

1. Pengguna memilih untuk mengubah informasi login.
2. Sistem menampilkan form ubah informasi login.
3. Pengguna memasukkan informasi login baru.
4. Sistem menghapus informasi login lama dan menyimpan informasi login baru.

Skenario memicu login otomatis

Nama: Memicu login otomatis
Aktor: WiFi
Kondisi awal: Perangkat lunak sudah dijalankan.
Deskripsi: Sistem memicu login otomatis berdasarkan perubahan status WiFi.
Kondisi Akhir: Klien terotentikasi pada *captive portal*.

Skenario:

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.

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:

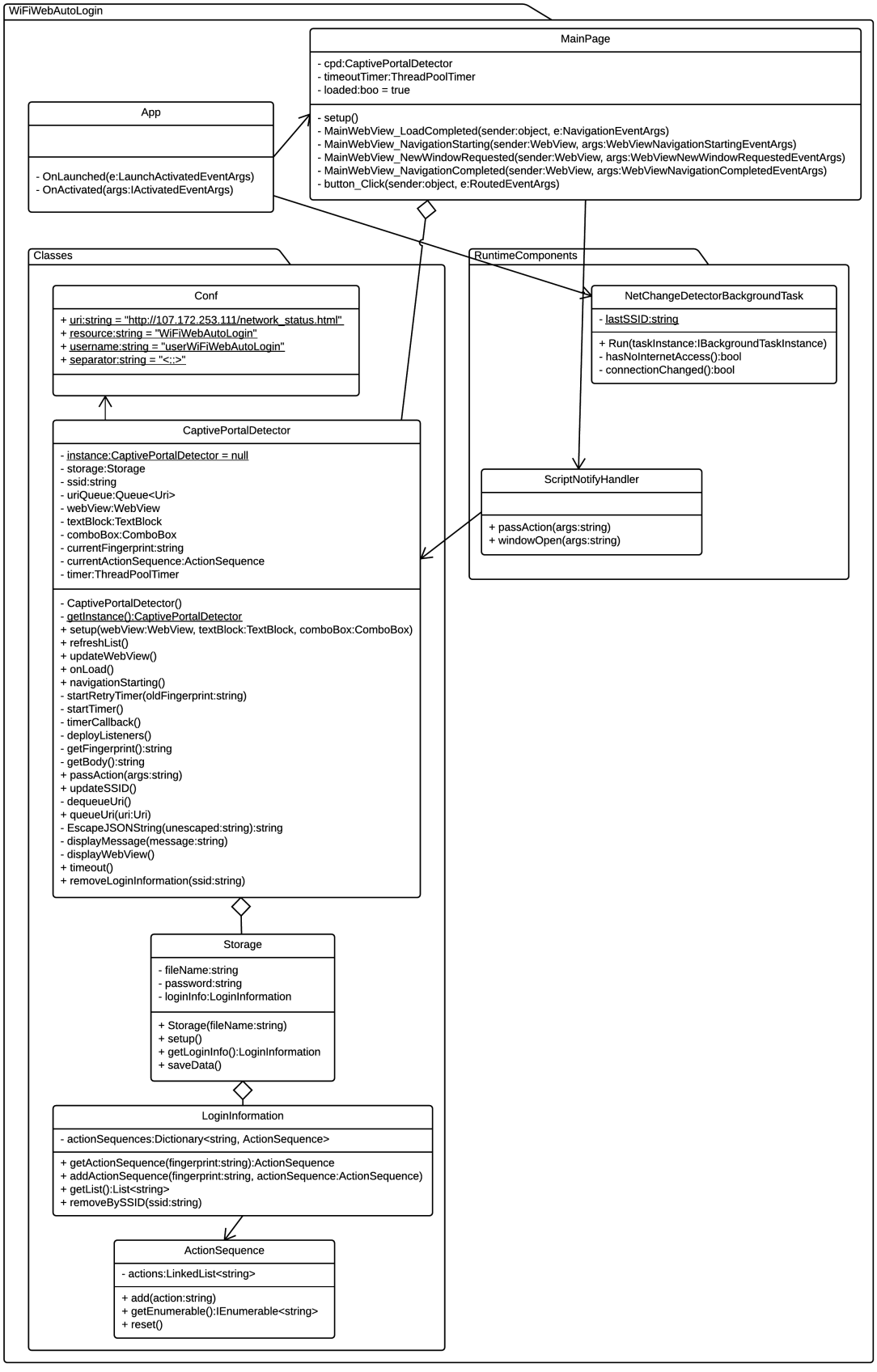
MainPage : Kelas ini merupakan kelas yang berperan sebagai tampilan utama aplikasi. Atribut-atribut yang dimiliki oleh kelas ini adalah:

- 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.



Gambar 4.1: Diagram Kelas Rinci.

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 medeteksi tidak adanya akses internet menggunakan API yang diberkan oleh UWP.

- conectionChanged
  Metode ini digunakan untuk medeteksi perubahan SSID.

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

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

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

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

- instance
  Atribut ini menyimpan *instance* CaptivePortalDetector.

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

- ssid
  Atribut ini menyimpan SSID saat ini.

- uriQueue
  Atribut ini menyimpan queue Uri yang perlu diakses.

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

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

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

- currentFingerprint
  Atribut ini menyimpan fingerprint saat ini.

- currentActionSequence
  Atribut ini menyimpan ActionSequence yang terkait dengan fingerprint saat ini.

- timer
  Atribut ini menyimpan timer yang digunakan untuk mengatur waktu akses Uri dalam uriQueue.

Metode-metode yang dimiliki oleh kelas ini adalah:

- getInstance
  Metode ini digunakan untuk mendapatkan *instance* dari CaptivePortalDetector.

- setup
  Metode ini digunakan untuk melakukan *setup* awal yang menyimpan WebView, TextBlock, dan ComboBox ke dalam *instance* CaptivePortalDetector.

- refreshList
  Metode ini digunakan untuk melakukan *refresh* tampilan ComboBox.

- updateWebView
  Metode ini digunakan untuk menentukan melakukan deteksi *captive portal* jika terhubung dengan koneksi WiFi, atau menampilkan pesan kepada pengguna juka 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:

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

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

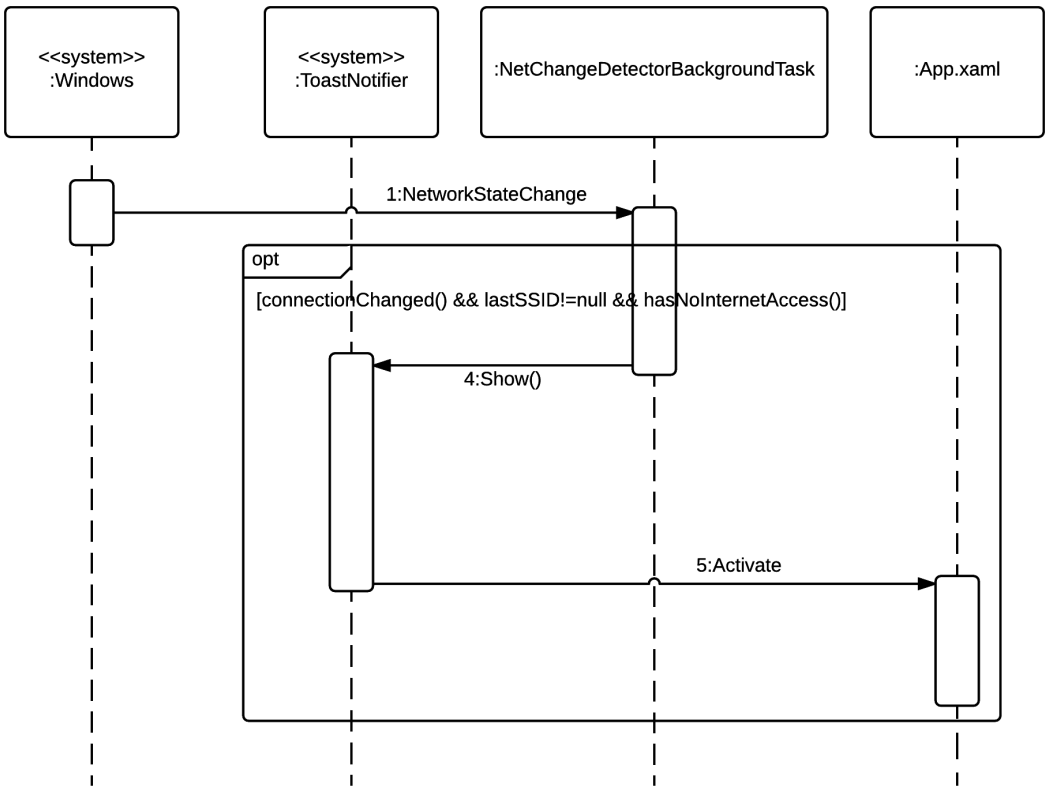
Kelas LoginInformation memiliki daftar objek bertipe ActionSequence yang disimpan pada properti actionSequences bertipe Dictionary. Kelas Dictionary dapat menyimpan data yang berupa pasangan *key-value*. *Key* yang digunakan bertipe string dan merupakan *fingerprint* suatu halaman. Value yang disimpan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

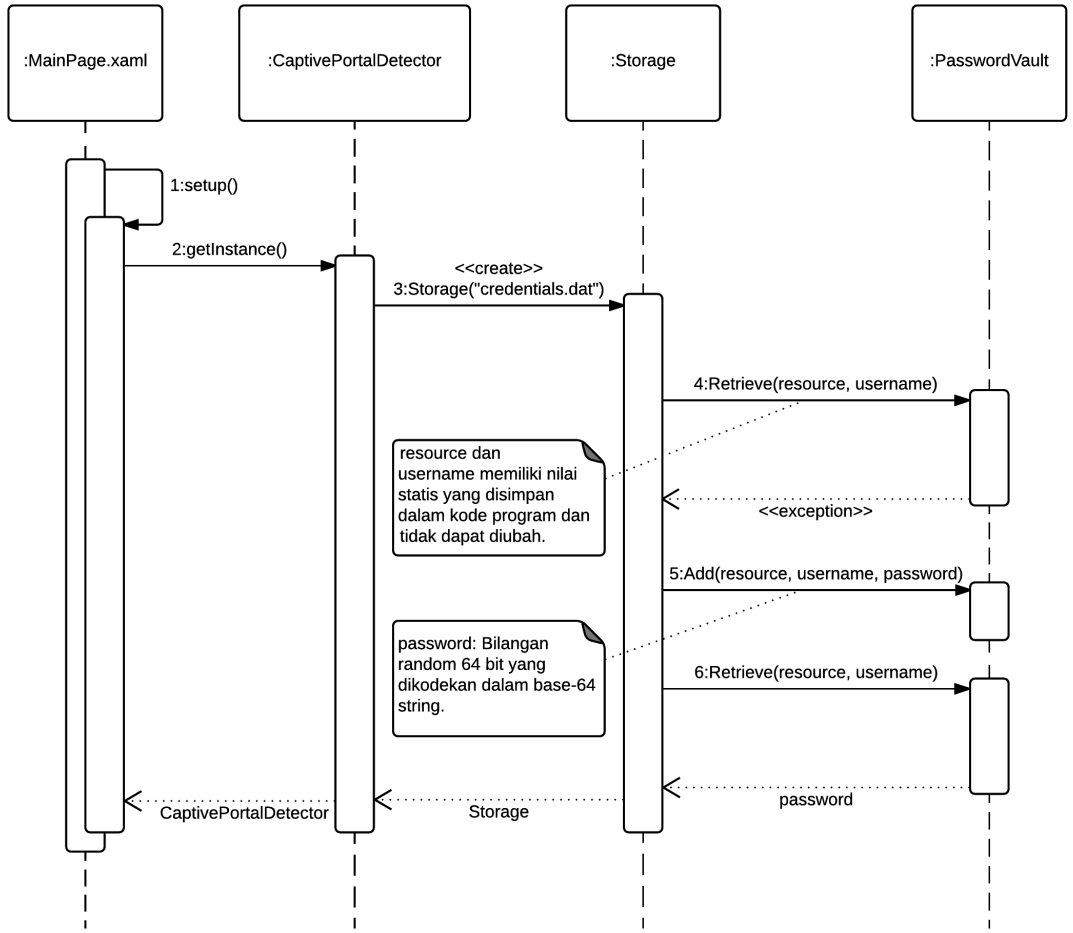
akan diaktifkan oleh Windows, dan objek NetChangeDetectorBackgroundTask yang sudah didaftarkan akan menerima *trigger* tersebut.

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.

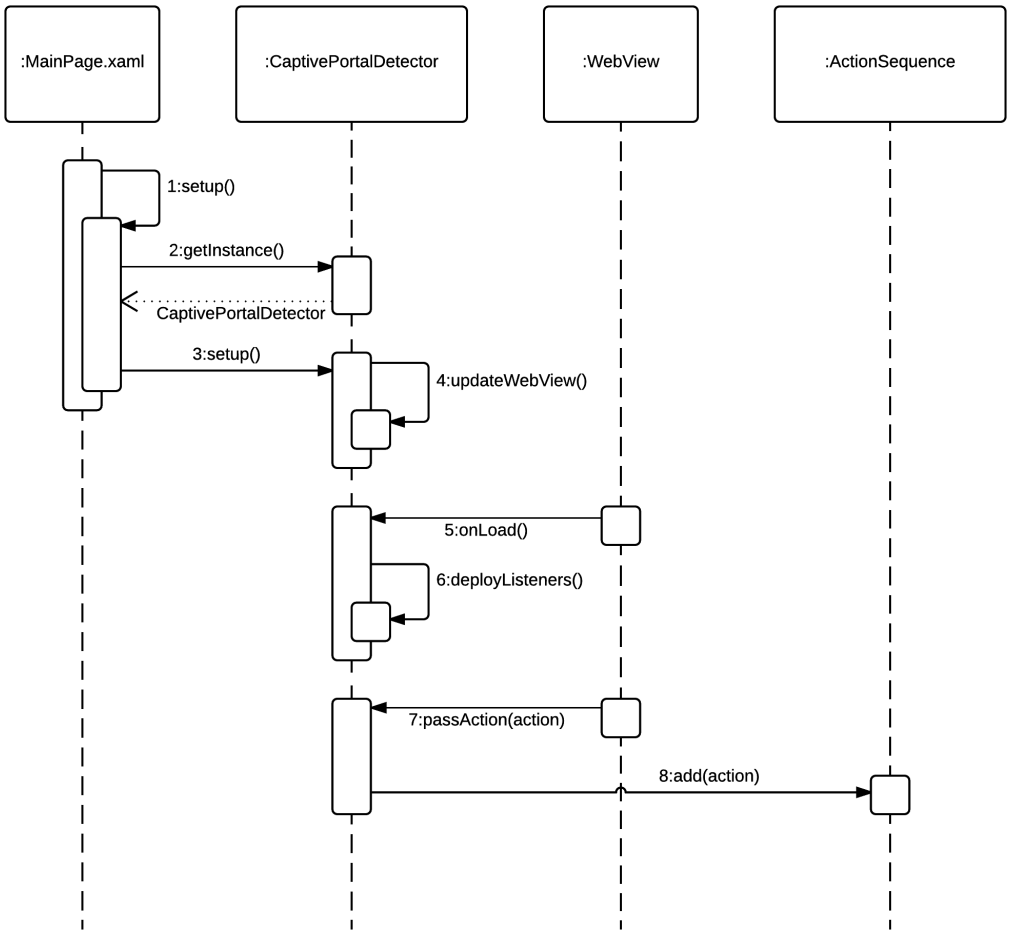
3. CaptivePortalDetector menciptakan objek Storage baru pada *constructor*-nya.

4. Objek Storage berusaha untuk mendapatkan password dengan memangil 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: Diagram 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.

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.".

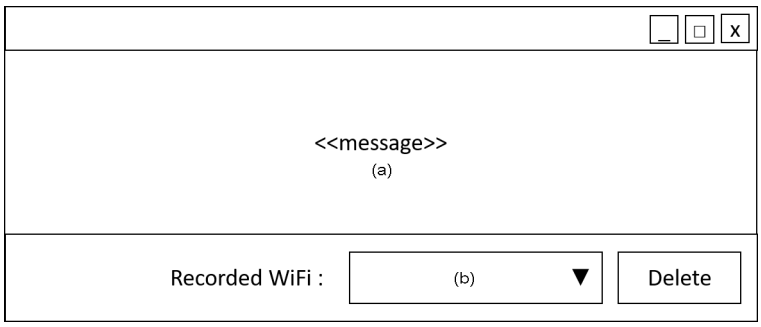
4.4.1 Antarmuka Notifikasi

Gambar 4.5: Rancangan 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 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.

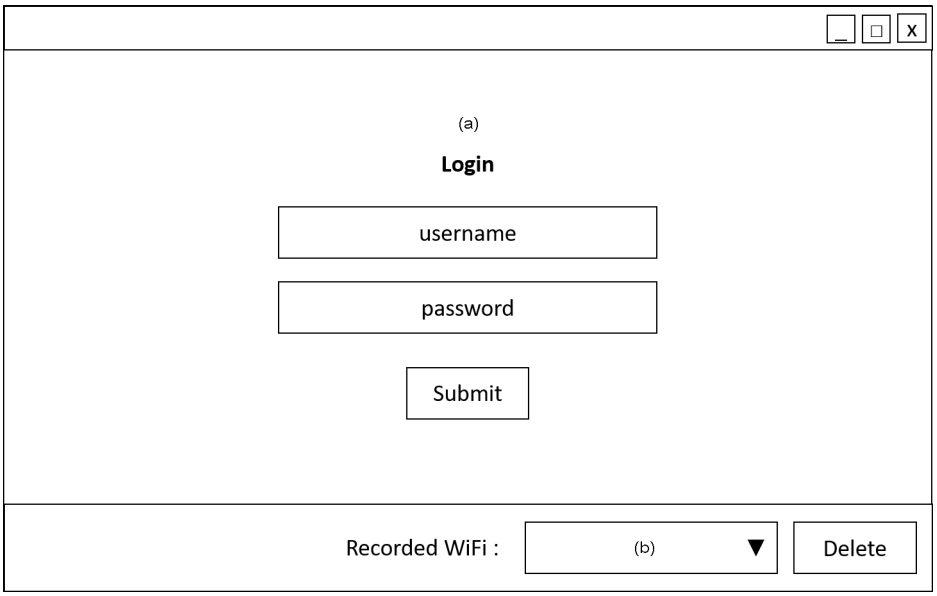


Gambar 4.6: Rancangan Antarmuka Message Box.

Antarmuka Message Box

Gambar 4.6 menampilkan desain antarmuka message box. Selain label (a) untuk menaruh pesan, antarmuka ini juga memiliki 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



Gambar 4.7: Rancangan Antarmuka Web Browser.

Gambar 4.7 menampilkan desain antarmuka web browser. 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.

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 IDE[1] 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 pada saat halaman selesai dimuat, dan mengubungkannya dengan kelas ScriptNotifyHandler. Akan tetapi, metode ini masih kurang memadai karena kode JavaScript yang memanggil fungsi-fungsi tersebut pada saat halaman dimuat akan dieksekusi sebelum *override* terjadi.

[1] Intergrated Develompent Environment

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 CAS[2]. Selain itu, CAS ini menggunakan *pop-up* untuk menampilkan halaman `https://wireless.unpar.ac.id/status` dan halaman tersebut adalah halaman yang membuka popup yang menuju ke halaman tujuan. WebView pada UWP tidak memperbolehkan pemanggilan fungsi open(), window.open(), el.click() dan form.submit() yang bukan merupakan aksi langsung oleh pengguna. *Override* tiap fungsi tersebut dimungkinkan setelah halaman selesai dimuat, namun itu berarti fungsi hasil *override* tidak akan digunakan jika fungsi tersebut dipanggil pada saat halaman pertama kali dimuat. Hal ini mencakup *onload event* dan script yang dipanggil langsung pada *script tag* baik pada *body* maupun *head*.

Hasil Pengujian WiFi FTIS.cisco

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

[2] *Central Authorization Service*

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] B. Potter and B. Fleck, *802.11 Security*. O'Reilly, 2002.

[2] HTTP Working Group, "Captive Portals." https://github.com/httpwg/wiki/wiki/Captive-Portals, 2016. [Online; diakses 11-September-2016].

[3] 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].

[4] Internet Engineering Task Force, "Additional HTTP Status Codes." https://tools.ietf.org/html/rfc6585, 2016. [Online; diakses 24-September-2016].

[5] R. Lander, ".NET Primer." https://docs.microsoft.com/en-us/dotnet/articles/standard/index, 2016. [Online; diakses 24-September-2016].

[6] 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].

[7] 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].

[8] Microsoft, "Windows API Reference." https://msdn.microsoft.com/en-us/library/windows/apps/bg124285.aspx, 2016. [Online; diakses 24-September-2016].

[9] 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

```
<?xml version="1.0" encoding="utf-8"?>
<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">
  <Identity Name="24a4f351-3e45-4b12-946e-bfc3593444b4"
    Publisher="CN=Yohanes Mario" Version="1.0.0.0" />
  <mp:PhoneIdentity PhoneProductId="24a4f351-3e45-4b12-946e-bfc3593444b4"
    PhonePublisherId="00000000-0000-0000-0000-000000000000" />
  <Properties>
    <DisplayName>WiFiWebAutoLogin</DisplayName>
    <PublisherDisplayName>Yohanes Mario</PublisherDisplayName>
    <Logo>Assets\StoreLogo.png</Logo>
  </Properties>
  <Dependencies>
    <TargetDeviceFamily Name="Windows.Universal" MinVersion="10.0.0.0"
      MaxVersionTested="10.0.0.0" />
  </Dependencies>
  <Resources>
    <Resource Language="x-generate" />
  </Resources>
  <Applications>
    <Application Id="App" Executable="$targetnametoken$.exe"
      EntryPoint="WiFiWebAutoLogin.App">
      <uap:VisualElements DisplayName="WiFiWebAutoLogin"
        Square150x150Logo="Assets\Square150x150Logo.png"
        Square44x44Logo="Assets\Square44x44Logo.png"
        Description="WiFiWebAutoLogin" BackgroundColor="transparent">
        <uap:DefaultTile Wide310x150Logo="Assets\Wide310x150Logo.png">
        </uap:DefaultTile>
        <uap:SplashScreen Image="Assets\SplashScreen.png" />
      </uap:VisualElements>
      <Extensions>
        <Extension Category="windows.backgroundTasks" EntryPoint="WiFiWebAutoLogin.RuntimeComponents.CustomBackgroundTask">
          <BackgroundTasks>
            <Task Type="systemEvent" />
          </BackgroundTasks>
        </Extension>
```

```
      </Extensions>
    </Application>
  </Applications>
  <Capabilities>
    <Capability Name="internetClient" />
  </Capabilities>
</Package>
```

1

WiFiWebAutoLogin/App.xaml

```
<Application
    x:Class="WiFiWebAutoLogin.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:WiFiWebAutoLogin"
    RequestedTheme="Light">

</Application>
```

2

WiFiWebAutoLogin/App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.ApplicationModel;
using Windows.ApplicationModel.Activation;
using Windows.ApplicationModel.Background;
using Windows.ApplicationModel.Core;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Notifications;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

namespace WiFiWebAutoLogin
{
    /// <summary>
    /// Provides application-specific behavior to supplement the default
    ↪ Application class.
    /// </summary>
    sealed partial class App : Application
    {
        /// <summary>
        /// Initializes the singleton application object.  This is the
        ↪ first line of authored code
        /// executed, and as such is the logical equivalent of main() or
        ↪ WinMain().
```

```
31          /// </summary>
32          public App()
33          {
34              this.InitializeComponent();
35              this.Suspending += OnSuspending;
36          }
37
38          /// <summary>
39          /// Invoked when the application is launched normally by the end
            ↪ user.  Other entry points
40          /// will be used such as when the application is launched to open
            ↪ a specific file.
41          /// </summary>
42          /// <param name="e">Details about the launch request and
            ↪ process.</param>
43          protected override void OnLaunched(LaunchActivatedEventArgs e)
44          {
45              // Initialize background task
46              bool taskRegistered = false;
47              string taskName = "CustomBackgroundTask";
48
49              foreach (var task in BackgroundTaskRegistration.AllTasks) {
50                  if (task.Value.Name == taskName) {
51                      task.Value.Unregister(true);
52                  }
53              }
54
55              if (!taskRegistered) {
56                  // Register task
57                  var builder = new BackgroundTaskBuilder();
58
59                  builder.Name = taskName;
60                  builder.TaskEntryPoint = "WiFiWebAutoLogin.RuntimeComponen
                    ↪ ts.CustomBackgroundTask";
61                  builder.SetTrigger(new
                    ↪ SystemTrigger(SystemTriggerType.NetworkStateChange,
                    ↪ false));
62                  builder.AddCondition(new
                    ↪ SystemCondition(SystemConditionType.UserPresent));
63                  BackgroundTaskRegistration task = builder.Register();
64              }
65  #if DEBUG
66              if (System.Diagnostics.Debugger.IsAttached) {
67                  //this.DebugSettings.EnableFrameRateCounter = true;
68                  this.DebugSettings.EnableFrameRateCounter = false;
69              }
70  #endif
71              Frame rootFrame = Window.Current.Content as Frame;
72
73              // Do not repeat app initialization when the Window already
                ↪ has content,
74              // just ensure that the window is active
75              if (rootFrame == null) {
```

```
76              // Create a Frame to act as the navigation context and
                ↪  navigate to the first page
77              rootFrame = new Frame();
78
79              rootFrame.NavigationFailed += OnNavigationFailed;
80
81              //if (e.PreviousExecutionState ==
                ↪  ApplicationExecutionState.Terminated) {
82                  //TODO: Load state from previously suspended
                    ↪  application
83              //}
84
85              // Place the frame in the current Window
86              Window.Current.Content = rootFrame;
87          }
88
89          //if (e.PrelaunchActivated == false) {
90              //if (rootFrame.Content == null) {
91                  // When the navigation stack isn't restored navigate
                    ↪  to the first page,
92                  // configuring the new page by passing required
                    ↪  information as a navigation
93                  // parameter
94                  rootFrame.Navigate(typeof(MainPage), e.Arguments);
95              //}
96              // Ensure the current window is active
97              Window.Current.Activate();
98          //}
99      }
100
101     protected override void OnActivated(IActivatedEventArgs args) {
102         if (args.Kind == ActivationKind.ToastNotification) {
103             var toastArgs = args as
                ↪  ToastNotificationActivatedEventArgs;
104             var arguments = toastArgs.Argument;
105
106             if (arguments == "Yes") {
107                 Frame rootFrame = Window.Current.Content as Frame;
108                 if (rootFrame == null) {
109                     rootFrame = new Frame();
110                     Window.Current.Content = rootFrame;
111                 }
112                 rootFrame.Navigate(typeof(MainPage));
113                 Window.Current.Activate();
114             }
115             else {
116                 CoreApplication.Exit();
117             }
118
119             ToastNotificationManager.History.Remove("WWAL_TOAST");
120         }
121     }
122
```

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

1

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"
                ↪  TextAlignment="Center" Text="" VerticalAlignment="Center"
                ↪  HorizontalAlignment="Center" FontSize="16"/>
```

```
13              <WebView Name="MainWebView"
                ↪  NewWindowRequested="MainWebView_NewWindowRequested"
                ↪  DOMContentLoaded="MainWebView_DOMContentLoaded"
                ↪  ContentLoading="MainWebView_ContentLoading" LongRunningScr
                ↪  iptDetected="MainWebView_LongRunningScriptDetected"
                ↪  PermissionRequested="MainWebView_PermissionRequested"
                ↪  LoadCompleted="MainWebView_LoadCompleted"
                ↪  NavigationStarting="MainWebView_NavigationStarting"
                ↪  NavigationCompleted="MainWebView_NavigationCompleted"
                ↪  Margin="0,768,0,-768" />
14          </Grid>
15          <Grid Height="52" VerticalAlignment="Bottom" RequestedTheme="Dark"
            ↪  Background="Black">
16              <Button x:Name="button" Content="Delete"
                ↪  HorizontalAlignment="Right" Margin="0,0,10,10"
                ↪  VerticalAlignment="Bottom" Click="button_Click"
                ↪  RequestedTheme="Dark"/>
17              <ComboBox x:Name="comboBox" HorizontalAlignment="Right"
                ↪  Margin="0,0,78,10" VerticalAlignment="Bottom" Width="200"
                ↪  RequestedTheme="Dark"/>
18              <TextBlock x:Name="textBlock1" HorizontalAlignment="Right"
                ↪  Margin="0,0,283,16" TextWrapping="Wrap" Text="Recorded
                ↪  WiFi :" VerticalAlignment="Bottom" RequestedTheme="Dark"/>
19          </Grid>
20      </Grid>
21  </Page>
```

1

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
    ↪  http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409
```

```
25
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 { Width = 600, Height = 150 });
41              ApplicationView.PreferredLaunchViewSize = new Size(600, 150);
42              ApplicationView.PreferredLaunchWindowingMode =
                ↪  ApplicationViewWindowingMode.PreferredLaunchViewSize;
43              MainWebView.Margin = new Thickness(0, int.MaxValue, 0,
                ↪  int.MinValue);
44              textBlock.Text = "Initializing...";
45              this.setup();
46          }
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,
            ↪  NavigationEventArgs e) {
55              if (cpd != null) {
56                  this.loaded = true;
57                  cpd.onLoad();
58              }
59          }
60
61          private void MainWebView_NavigationStarting(WebView sender,
            ↪  WebViewNavigationStartingEventArgs args) {
62              Debug.WriteLine(args.Uri);
63              if (cpd != null) {
64                  this.cpd.navigationStarting();
65
66                  if (this.timeoutTimer!=null) {
67                      this.timeoutTimer.Cancel();
68                      this.timeoutTimer = null;
69                  }
70
71                  ScriptNotifyHandler scriptNotify = new
                    ↪  ScriptNotifyHandler();
```

```
72              MainWebView.AddWebAllowedObject("ScriptNotifyHandler",
                ↪ scriptNotify);
73
74              // Handle Timeout
75              this.loaded = false;
76              this.timeoutTimer = ThreadPoolTimer.CreateTimer(async
                ↪ (source) => {
77                  await Windows.ApplicationModel.Core.CoreApplication.Ma
                    ↪ inView.CoreWindow.Dispatcher.RunAsync(Windows.UI.C
                    ↪ ore.CoreDispatcherPriority.Normal, () =>
                    ↪ {
78                      this.timeoutTimer = null;
79                      if (!this.loaded) {
80                          this.cpd.timeout();
81                      }
82                  });
83              }, TimeSpan.FromSeconds(20));
84          }
85      }
86
87      private void MainWebView_NewWindowRequested(WebView sender,
        ↪ WebViewNewWindowRequestedEventArgs args) {
88          args.Handled = true;
89          cpd.queueUri(args.Uri);
90      }
91
92      private async void MainWebView_NavigationCompleted(WebView sender,
        ↪ WebViewNavigationCompletedEventArgs args) {
93          if (cpd != null) {
94              await sender.InvokeScriptAsync("eval", new string[] {
95                  "window.open = function(url){ScriptNotifyHandler.windo
                    ↪ wOpen(url)};"
                    ↪ +
96                  "var open = window.open;" +
97                  "document.open = window.open;"
98              });
99          }
100     }
101
102     private void button_Click(object sender, RoutedEventArgs e) {
103         cpd.removeLoginInformation((string)comboBox.SelectedItem);
104     }
105
106     private void MainWebView_DOMContentLoaded(WebView sender,
        ↪ WebViewDOMContentLoadedEventArgs args) {
107         // DISABLED
108     }
109
110     private void MainWebView_ContentLoading(WebView sender,
        ↪ WebViewContentLoadingEventArgs args) {
111         // DISABLED
112     }
113
```

```
        private void MainWebView_LongRunningScriptDetected(WebView sender,
        ↪ WebViewLongRunningScriptDetectedEventArgs args) {
            // DISABLED
        }

        private void MainWebView_PermissionRequested(WebView sender,
        ↪ WebViewPermissionRequestedEventArgs args) {
            // DISABLED
        }
    }
}
```

WiFiWebAutoLogin/JavaScript/DeployListeners.js

```
var inputs = document.getElementsByTagName("input");
var ahrefs = document.getElementsByTagName("a");
var buttons = document.getElementsByTagName("button");

function addClickListener(el, tagName, idx) {
    el.addEventListener("click", function () {
        ScriptNotifyHandler.passAction("document.getElementsByTagName(\""
        ↪ + tagName + "\")[" + idx + "].click();");
    });
}
function addChangeListener(el, tagName, idx) {
    el.addEventListener("change", function () {
        ScriptNotifyHandler.passAction("document.getElementsByTagName(\""
        ↪ + tagName + "\")[" + idx + "].value = " + JSON.stringify(docum
        ↪ ent.getElementsByTagName(tagName)[idx].value) +
        ↪ ";");
    });
}

for (var i = 0; i < inputs.length; i++) {
    addClickListener(inputs[i], "input", i);
    addChangeListener(inputs[i], "input", i);
}

for (var i = 0; i < ahrefs.length; i++) {
    addClickListener(ahrefs[i], "a", i);
}

for (var i = 0; i < buttons.length; i++) {
    addClickListener(buttons[i], "button", i);
}
```

A.2 Namespace: WiFiWebAutoLogin.Classes

WiFiWebAutoLogin.Classes/ActionSequence.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace WiFiWebAutoLogin.Classes {
    [DataContract]
    class ActionSequence {
        [DataMember]
        private LinkedList<string> actions;

        public ActionSequence() {
            this.actions = new LinkedList<string>();
        }

        public void add(string action) {
            this.actions.AddLast(action);
        }

        public IEnumerable<string> getEnumerable() {
            return this.actions.AsEnumerable();
        }

        public void reset() {
            this.actions.Clear();
        }
    }
}
```

1

WiFiWebAutoLogin.Classes/CaptivePortalDetector.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Net.NetworkInformation;
using System.Runtime.Serialization.Json;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
using System.Threading.Tasks;
using Windows.Foundation;
using Windows.Foundation.Metadata;
using Windows.Networking.Connectivity;
using Windows.Storage;
using Windows.System.Threading;
using Windows.UI.ViewManagement;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace WiFiWebAutoLogin.Classes {
    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
            ↪ comboBox) {
49              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(Conf.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().g
                    ↪  etActionSequence(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.cur
                        ↪  rentFingerprint,
                        ↪  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.MainVi⌋
                    ↪ ew.CoreWindow.Dispatcher.RunAsync(Windows.UI.Core.Core⌋
                    ↪ DispatcherPriority.Normal, () =>
                    ↪ {
147                     if (this.currentFingerprint.Equals(oldFingerprint)) {
148                         this.displayWebView();
149                     }
150                 });
151             }, TimeSpan.FromSeconds(5));
152         }
153
154         private async void timerCallback() {
155             await Windows.ApplicationModel.Core.CoreApplication.MainView.C⌋
                ↪ oreWindow.Dispatcher.RunAsync(Windows.UI.Core.CoreDispatch⌋
                ↪ erPriority.Normal, () =>
                ↪ {
156                 if (this.timer!=null) {
157                     this.timer = null;
158                 }
159                 this.dequeueUri();
160             });
161         }
162
163         private async void deployListeners() {
164             StorageFolder InstallationFolder =
                ↪ Windows.ApplicationModel.Package.Current.InstalledLocation;
165             StorageFile file = await InstallationFolder.GetFileAsync(@"Jav⌋
                ↪ aScript\DeployListeners.js");
```

```
166             string js = await FileIO.ReadTextAsync(file);
167             await this.webView.InvokeScriptAsync("eval", new string[] { js
                    ↪ });
168         }
169
170         private async Task<string> getFingerprint() {
171             string uri = "";
172             string title = "";
173             try {
174                 uri = await this.webView.InvokeScriptAsync("eval", new
                        ↪ string[] { "window.location.href;" });
175                 title = await this.webView.InvokeScriptAsync("eval", new
                        ↪ string[] { "document.getElementsByTagName(\"title\")[0
                        ↪ ].innerHTML.trim();"
                        ↪ });
176             } catch (Exception e) {
177             }
178             return this.ssid + Conf.separator + uri + Conf.separator +
                    ↪ title;
179         }
180
181         private async Task<string> getUri() {
182             string uri = "";
183             try {
184                 uri = await this.webView.InvokeScriptAsync("eval", new
                        ↪ string[] { "window.location.href;" });
185             }
186             catch (Exception e) {
187             }
188             return uri;
189         }
190
191         private async Task<string> getBody() {
192             return await this.webView.InvokeScriptAsync("eval", new
                    ↪ string[] { "document.body.innerHTML;" });
193         }
194
195         private async Task<string> getScripts() {
196             return await this.webView.InvokeScriptAsync("eval", new
                    ↪ string[] { "document.body.innerHTML;" });
197         }
198
199         public void passAction(string args) {
200             if (this.currentActionSequence!=null) {
201                 this.currentActionSequence.add(args);
202                 this.storage.saveData();
203             }
204         }
205
206         public void updateSSID() {
207             ConnectionProfile connectionProfile =
                    ↪ NetworkInformation.GetInternetConnectionProfile();
208
```

```
209             string data = "";
210             if (connectionProfile != null) {
211                 Debug.WriteLine("[NETWORK]: "+connectionProfile.GetNetwork
                    ↪ ConnectivityLevel().ToString());
212 
213                 IEnumerable<string> enumerable =
                    ↪ connectionProfile.GetNetworkNames().AsEnumerable();
214                 foreach (string v in enumerable) {
215                     if (data.Equals("")) {
216                         data += v;
217                     }
218                     else {
219                         data += " | " + v;
220                     }
221                 }
222                 if (data.Equals("")) {
223                     this.ssid = null;
224                 }
225                 else {
226                     this.ssid = data;
227                 }
228             }
229             else {
230                 this.ssid = null;
231             }
232         }
233 
234         public string getSSID() {
235             return this.ssid;
236         }
237 
238         private void dequeueUri() {
239             Uri uri;
240             try {
241                 uri = this.uriQueue.Dequeue();
242             } catch (Exception e) {
243                 uri = null;
244             }
245 
246             if (uri!=null) {
247 
248                 this.webView.Navigate(uri);
249             }
250         }
251 
252         private async Task<string> EscapeJSONString(string unescaped) {
253             DataContractJsonSerializer serializer = new
                ↪ DataContractJsonSerializer(typeof(LoginInformation));
254             MemoryStream stream = new MemoryStream(); ;
255             serializer.WriteObject(stream, unescaped);
256             stream.Position = 0;
257             return await (new StreamReader(stream)).ReadToEndAsync();
258         }
```

```
259
260         public void queueUri(Uri uri) {
261             this.uriQueue.Enqueue(uri);
262             if (this.uriQueue.Count == 1) {
263                 this.startTimer();
264             }
265         }
266
267         private void startTimer() {
268             this.timer = ThreadPoolTimer.CreateTimer((source) => {
269                 this.timerCallback();
270             }, TimeSpan.FromSeconds(1));
271         }
272
273         private void displayMessage(string message) {
274             ApplicationView.GetForCurrentView().TryResizeView(new Size {
                ↪  Width = 600, Height = 150 });
275             this.textBlock.Text = message;
276             this.webView.Margin = new Thickness(0, int.MaxValue, 0,
                ↪  int.MinValue);
277         }
278
279         private void displayWebView() {
280             ApplicationView.GetForCurrentView().TryResizeView(new Size {
                ↪  Width = 800, Height = 500 });
281             this.textBlock.Text = "";
282             this.webView.Margin = new Thickness(0, 0, 0, 0);
283         }
284
285         public void timeout() {
286             this.displayMessage("Operation timeout.\r\nCheck your network
                ↪  connection.");
287         }
288
289         public void removeLoginInformation(string ssid) {
290             if (ssid != null) {
291                 this.storage.getLoginInfo().removeBySSID(ssid);
292                 this.storage.saveData();
293                 this.refreshList();
294             }
295         }
296     }
297 }
```

1

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  }
```

1

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[] { Conf.separator },
                    ↪ StringSplitOptions.RemoveEmptyEntries)[0];
38                  if (!list.Contains(ssid)) {
39                      list.Add(ssid);
```

```
                }
            }
            return list;
        }

        public void removeBySSID(string ssid) {
            Dictionary<string, ActionSequence>.KeyCollection.Enumerator
            ↪ loginInfoEnumerator = actionSequences.Keys.GetEnumerator();
            List<string> removalList = new List<string>();
            while (loginInfoEnumerator.MoveNext()) {
                string enumSSID = loginInfoEnumerator.Current.Split(new
                ↪ string[] { Conf.separator },
                ↪ StringSplitOptions.RemoveEmptyEntries)[0];
                if (enumSSID.Equals(ssid)) {
                    removalList.Add(loginInfoEnumerator.Current);
                }
            }
            List<string>.Enumerator removalListEnumerator =
            ↪ removalList.GetEnumerator();
            while (removalListEnumerator.MoveNext()) {
                actionSequences.Remove(removalListEnumerator.Current);
            }
        }
    }
}
```

1

WiFiWebAutoLogin.Classes/Storage.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Json;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Serialization;
using Windows.Security.Credentials;
using Windows.Security.Cryptography;
using Windows.Security.Cryptography.Core;
using Windows.Storage;
using Windows.Storage.Streams;

namespace WiFiWebAutoLogin.Classes {
    class Storage {
        private string fileName;
        private string password;
        private LoginInformation loginInfo;

        public Storage(string fileName) {
            this.fileName = new String(fileName.ToCharArray());
            PasswordVault vault = new PasswordVault();

            try {
```

```
26              this.password = vault.Retrieve(Conf.resource,
                ↪ Conf.username).Password;
27          }
28          catch (Exception e) {
29              vault.Add(new PasswordCredential(Conf.resource,
                ↪ Conf.username, CryptographicBuffer.EncodeToBase64Strin
                ↪ g(CryptographicBuffer.GenerateRandom(64))));
30              this.password = vault.Retrieve(Conf.resource,
                ↪ Conf.username).Password;
31          }
32      }
33
34      public async Task setup() {
35          StorageFolder folder = ApplicationData.Current.LocalFolder;
36          StorageFile file;
37          try {
38              file = await folder.GetFileAsync(this.fileName);
39          } catch (Exception e) {
40              file = await folder.CreateFileAsync(this.fileName);
41          }
42
43          IBuffer encryptedJson = await FileIO.ReadBufferAsync(file);
44          SymmetricKeyAlgorithmProvider algorithmProvider =
            ↪ SymmetricKeyAlgorithmProvider.OpenAlgorithm(SymmetricAlgor
            ↪ ithmNames.AesCbcPkcs7);
45          IBuffer bufferedPassword =
            ↪ CryptographicBuffer.ConvertStringToBinary(password,
            ↪ BinaryStringEncoding.Utf8);
46          IBuffer decryptedJson = CryptographicEngine.Decrypt(algorithmP
            ↪ rovider.CreateSymmetricKey(bufferedPassword),
            ↪ encryptedJson, bufferedPassword);
47          DataReader dataReader = Windows.Storage.Streams.DataReader.Fro
            ↪ mBuffer(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));
```

```
            MemoryStream stream = new MemoryStream();
            serializer.WriteObject(stream, loginInfo);
            stream.Position = 0;
            StreamReader sr = new StreamReader(stream);
            string data = sr.ReadToEnd();

            SymmetricKeyAlgorithmProvider algorithmProvider =
            ↪ SymmetricKeyAlgorithmProvider.OpenAlgorithm(SymmetricAlgor
            ↪ ithmNames.AesCbcPkcs7);
            IBuffer keyMaterial =
            ↪ CryptographicBuffer.ConvertStringToBinary(password,
            ↪ BinaryStringEncoding.Utf8);
            IBuffer bufferedData = CryptographicBuffer.CreateFromByteArray
            ↪ (Encoding.UTF8.GetBytes(data));
            CryptographicKey key =
            ↪ algorithmProvider.CreateSymmetricKey(keyMaterial);
            IBuffer encryptedData = CryptographicEngine.Encrypt(key,
            ↪ bufferedData, keyMaterial);

            StorageFolder folder = ApplicationData.Current.LocalFolder;
            StorageFile file;
            try {
                file = await folder.GetFileAsync(this.fileName);
            }
            catch (Exception e) {
                file = await folder.CreateFileAsync(this.fileName);
            }

            await FileIO.WriteBufferAsync(file, encryptedData);
        }
    }
}
```

A.3 Namespace: WiFiWebAutoLogin.RuntimeComponents

WiFiWebAutoLogin.RuntimeComponents/NetChangeDetectorBackgroundTask.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.ApplicationModel.Background;
using Windows.Networking.Connectivity;
using WiFiWebAutoLogin.Classes;
using Windows.UI.Notifications;
using Windows.Data.Xml.Dom;
using System.IO;

namespace WiFiWebAutoLogin.RuntimeComponents {
    public sealed class NetChangeDetectorBackgroundTask : IBackgroundTask {
        private static string lastSSID = "";
```

```
17
18          public void Run(IBackgroundTaskInstance taskInstance) {
19              if (this.connectionChanged() && lastSSID!=null &&
                ↪  this.hasNoInternetAccess()) {
20
21                  string xmlText = "<?xml version=\"1.0\" encoding=\"utf-8\"
                    ↪  ?>" +
22                      "<toast launch=\"app-defined-string\">" +
23                          "<visual>" +
24                              "<binding template=\"ToastGeneric\">" +
25                                  "<text>Network Detected</text>" +
26                                  "<text>Would you like to run
                                    ↪  WiFiWebAutoLogin?</text>" +
27                              "</binding>" +
28                          "</visual>" +
29                          "<actions>" +
30                              "<action content=\"Yes\" arguments=\"Yes\" />"
                                ↪  +
31                              "<action content=\"No\" arguments=\"No\"
                                ↪  activationType=\"background\" />" +
32                          "</actions>" +
33                          "<audio src=\"ms-winsoundevent:Notification.Remind
                            ↪  er\"/>"
                            ↪  +
34                      "</toast>";
35
36                  XmlDocument xmlContent = new XmlDocument();
37                  xmlContent.LoadXml(xmlText);
38
39                  ToastNotification notification = new
                    ↪  ToastNotification(xmlContent);
40                  notification.Tag = "WWAL_TOAST";
41                  notification.Dismissed += (ToastNotification n,
                    ↪  ToastDismissedEventArgs args) => {
42                      ToastNotificationManager.History.Remove("WWAL_TOAST");
43                  };
44                  ToastNotificationManager.CreateToastNotifier().Show(notifi
                    ↪  cation);
45              }
46          }
47
48          private bool hasNoInternetAccess() {
49              ConnectionProfile connectionProfile =
                ↪  NetworkInformation.GetInternetConnectionProfile();
50
51              if (connectionProfile != null) {
52                  if (connectionProfile.GetNetworkConnectivityLevel().ToStri
                    ↪  ng().Trim().Equals("InternetAccess"))
                    ↪  {
53                      return false;
54                  }
55              }
56
```

```
57              return true;
58          }
59
60          private bool connectionChanged() {
61              string ssid;
62              ConnectionProfile connectionProfile =
                ↪ NetworkInformation.GetInternetConnectionProfile();
63              string data = "";
64              if (connectionProfile != null) {
65                  IEnumerable<string> enumerable =
                    ↪ connectionProfile.GetNetworkNames().AsEnumerable();
66                  foreach (string v in enumerable) {
67                      if (data.Equals("")) {
68                          data += v;
69                      }
70                      else {
71                          data += " | " + v;
72                      }
73                  }
74                  if (data.Equals("")) {
75                      ssid = null;
76                  }
77                  else {
78                      ssid = data;
79                  }
80              }
81              else {
82                  ssid = null;
83              }
84
85              if (lastSSID != null) {
86                  if (lastSSID.Equals(ssid)) {
87                      lastSSID = ssid;
88                      return false;
89                  }
90                  else {
91                      lastSSID = ssid;
92                      return true;
93                  }
94              }
95              else {
96                  if (ssid==null) {
97                      lastSSID = ssid;
98                      return false;
99                  }
100                 else {
101                     lastSSID = ssid;
102                     return true;
103                 }
104             }
105         }
106     }
107 }
```

WiFiWebAutoLogin.RuntimeComponents/ScriptNotifyHandler.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.Foundation.Metadata;
using WiFiWebAutoLogin.Classes;
using System.Diagnostics;

namespace WiFiWebAutoLogin.RuntimeComponents
{
    [AllowForWeb]
    public sealed class ScriptNotifyHandler
    {
        public async void passAction(string args) {
            CaptivePortalDetector cpd = await
            ↪  CaptivePortalDetector.getInstance();
            cpd.passAction(args);
        }

        public async void windowOpen(string args) {
            CaptivePortalDetector cpd = await
            ↪  CaptivePortalDetector.getInstance();
            cpd.queueUri(new Uri(args));
            Debug.WriteLine(args);
        }

        public void testDebug(string args) {
            Debug.WriteLine(args);
        }
    }
}
```