

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
2016

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
2016**

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

DAFTAR ISI	ix
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Tujuan Penelitian	2
1.4 Batasan Masalah	2
1.5 Metodologi Penelitian	2
1.6 Sistematika Pembahasan	3
2 DASAR TEORI	5
2.1 <i>Captive Portal</i>	5
2.2 <i>.NET Framework</i>	6
2.3 <i>Common Language Infrastructure</i> (CLI)	7
2.4 <i>Universal Windows Platform</i> (UWP)	8
2.5 Kelas WebView Pada UWP	8
2.6 <i>Interface IBackgroundTask</i> Pada UWP	8
2.7 Kelas PasswordVault Pada Windows	9
3 ANALISIS	11
3.1 Analisis Perangkat Lunak Sejenis	11
3.2 Analisis Metode Penyimpanan Informasi Login	12
3.3 Analisis Metode Rekam dan Kirim Informasi Login	13
3.4 Analisis Metode Deteksi <i>Captive Portal</i>	14
3.5 Analisis Perangkat Lunak	15
3.5.1 Analisis Kelas	15
3.5.2 Analisis <i>Use Case</i>	15
4 PERANCANGAN	19
4.1 Perancangan Kelas	19
4.2 Perancangan Algoritma dan Struktur Data	24
4.2.1 Algoritma Deteksi <i>Captive Portal</i>	24
4.2.2 Struktur Data dan Format <i>Fingerprint</i>	24
4.2.3 Struktur Data LoginInformation	25
4.3 Perancangan Interaksi Perangkat Lunak	25
4.3.1 Perancangan Interaksi Deteksi Jaringan	25
4.3.2 Perancangan Interaksi Penciptaan Password	26
4.3.3 Perancangan Interaksi Penyimpanan Informasi Login	27
4.4 Perancangan Antarmuka	28
4.4.1 Antarmuka Notifikasi	28
4.4.2 Antarmuka Aplikasi	29
5 IMPLEMENTASI DAN PENGUJIAN	31

5.1	Lingkungan Implementasi dan Pengujian	31
5.2	Masalah Implementasi dan Solusinya	31
5.3	Pengujian Fungsional	32
5.3.1	Rencana Pengujian Fungsional	32
5.3.2	Hasil Pengujian Fungsional	32
5.4	Pengujian Eksperimental	33
5.4.1	Rencana Pengujian Eksperimental	33
5.4.2	Hasil Pengujian Eksperimental	33
6	KESIMPULAN DAN SARAN	35
6.1	Kesimpulan	35
6.2	Saran	35
	DAFTAR REFERENSI	37
A	KODE SUMBER	39
A.1	Namespace: WiFiWebAutoLogin	39
A.2	Namespace: WiFiWebAutoLogin.Classes	47
A.3	Namespace: WiFiWebAutoLogin.RuntimeComponents	58

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 Language 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, dan/atau halaman *login*.
5. Membuka akses jika pengguna telah menyetujui aturan penggunaan atau telah melakukan *login*.
6. Opsional: Menutup akses saat pengguna telah melewati batas waktu tertentu.

Akan tetapi, pada prakteknya, implementasi *captive portal* sangat beragam dan bersifat *ad-hoc*[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*.

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

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

- 1 • 4xx (Kesalahan Klien): *Request* mengandung sintaks yang buruk.
- 2 • 5xx (Kesalahan *Server*): *Server* tidak dapat memenuhi *request* yang valid.

3 Kode status HTTP 511 menandakan bahwa klien perlu melakukan otentikasi untuk men-
 4 dapatkan akses pada jaringan yang bersangkutan[4]. Respon dengan kode status ini harus
 5 menyertakan *link* ke sumber yang memungkinkan pengguna untuk memasukkan kredensial.
 6 Selain itu, respon dengan kode status ini tidak boleh diberikan oleh server tujuan. Respon
 7 ini dimaksudkan sebagai kontrol akses pada jaringan yang akan diberikan oleh komponen
 8 perantara dalam jaringan. Respon dengan kode status 511 tidak boleh disimpan oleh *cache*.

9 Kode Status HTTP 511 dan *Captive Portal*

10 Kode status 511 diciptakan untuk mengurangi masalah yang ditimbulkan oleh *captive portal*
 11 kepada perangkat lunak yang mengharapkan respon dari server tujuan, bukan dari komponen
 12 perantara dalam jaringan. Sebagai contoh, perangkat lunak yang bersangkutan mengirimkan
 13 *request* HTTP pada port TCP 80 sebagai berikut:

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

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

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

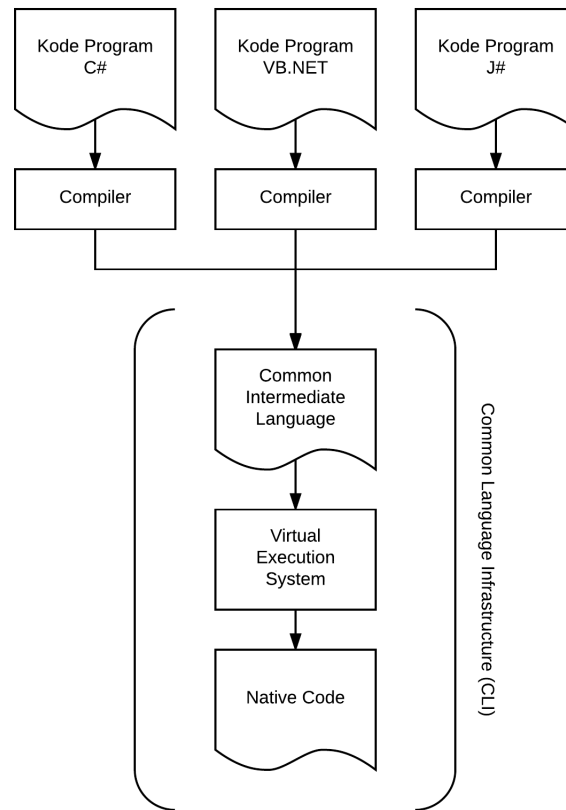
16 Respon ini memungkinkan klien untuk mendeteksi bahwa respon tersebut bukan berasal
 17 dari server tujuan. Selain itu, elemen meta pada HTML yang disajikan memungkinkan klien
 18 untuk melakukan login pada *link* yang diberikan.

19 2.2 .NET Framework

20 .NET adalah platform yang bersifat *general purpose* untuk membangun aplikasi[5]. .NET
 21 memberikan lingkungan untuk melakukan pemrograman *high-level* dan tetap memberikan
 22 akses *low-level* ke memori dan beberapa API. .NET memiliki beberapa implementasi ber-
 23 dasarkan standar *open* .NET yang mendefinisikan dasar-dasar yang harus dimiliki oleh pla-
 24 tform ini. Implementasi-implementasi ini memberikan dukungan bagi beberapa *chip* (seperti
 25 x86/x64 dan ARM) dan beberapa sistem operasi (seperti Windows, Linux, iOS, Android,
 26 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 Infrastructure* (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 Specification* (CLS)
CLS adalah perjanjian desainer bahasa pemrograman dan desainer *framework*. Perjanjian ini menentukan bagian minimal dari CTS yang harus diimplementasikan oleh bahasa pemrograman dan *framework* yang bersangkutan.
- *Virtual Execution System* (VES)
VES mengimplementasikan model CTS dan memastikan hal tersebut berjalan sebagaimana mestinya. VES berfungsi untuk menjalankan program yang ditulis untuk CLI.

1 Seperti yang dijelaskan pada Gambar 2.1, beberapa bahasa pemrograman yang kompa-
 2 tibel dengan .NET adalah C#, VB.NET, dan J#[5]. Setelah kode program dari bahasa-
 3 bahasa tersebut diterjemahkan oleh *compiler*nya masing-masing, maka akan terbentuk *Com-*
 4 *mon Intermediate Language* (CIL) yang kemudian akan dibaca dan dieksekusi oleh VES[6].
 5 Implementasi VES pada .NET bernama *Common Language Runtime* (CLR).

6 2.4 Universal Windows Platform (UWP)

7 Windows 8 memperkenalkan *Windows Runtime* (WinRT) yang merupakan arsitektur apli-
 8 kasi umum untuk Windows[7]. Saat Windows Phone 8.1 keluar, Windows Runtime pada
 9 Windows 8 dan Windows Phone 8.1 disejajarkan agar *developer* dapat membangun satu
 10 aplikasi yang dapat dijalankan pada Windows 8 dan Windows Phone 8.1. *Universal Windo-*
 11 *ws Platform* (UWP) pertama kali diperkenalkan pada Windows 10 sebagai perubahan dari
 12 model *Windows Runtime*. UWP tidak hanya dapat memanggil API dari WinRT, namun
 13 juga API spesifik dari device yang bersangkutan (seperti Win32 dan .NET).

14 2.5 Kelas WebView Pada UWP

15 Kelas WebView pada UWP memungkinkan *developer* untuk menampung konten HTML
 16 pada suatu aplikasi[8]. WebView tidak mendukung masukkan pengguna seperti *key-down*,
 17 *key-up*, dan *pointer-pressed*. Oleh karena itu, dibutuhkan metode lain yang melibatkan
 18 InvokeScriptAsync dengan fungsi *eval* javascript untuk menggunakan HTML *event handler*
 19 dan fungsi *window.external.notify* untuk menangani event dari HTML pada aplikasi.

20 Beberapa *property* yang dimiliki oleh WebView adalah:

- 21 • DocumentTitle
 22 Menyimpan *title* dari dokumen yang sedang ditampilkan dalam bentuk String.
- 23 • BaseUri
 24 Menyimpan *uri* dari dokumen yang sedang ditampilkan dalam bentuk Uri.

25 Beberapa *method* yang dimiliki oleh WebView adalah:

- 26 • InvokeScriptAsync(String scriptName, String[] arguments)
 27 Method ini digunakan untuk melakukan eksekusi script tertentu pada HTML dengan
 28 argumen yang diberikan dalam bentuk *array of string*.
- 29 • Navigate(Uri source)
 30 Method ini digunakan untuk membuka URI tertentu.

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

33 2.6 Interface IBackgroundTask Pada UWP

34 *Interface* IBackgroundTask pada UWP memiliki metode yang dapat dijalankan di belakang
 35 layar[8]. Interface ini memiliki satu metode yang harus diimplementasikan oleh kelas yang
 36 mengimplementasikan *interface* ini, yaitu:

- 37 • Run(IBackgroundTaskInstance taskInstance)
 38 Metode ini adalah metode yang dijalankan berdasarkan *trigger* yang diberikan oleh sis-
 39 tem operasi. Metode ini memiliki satu parameter yaitu *taskInstance* yang merupakan
 40 *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

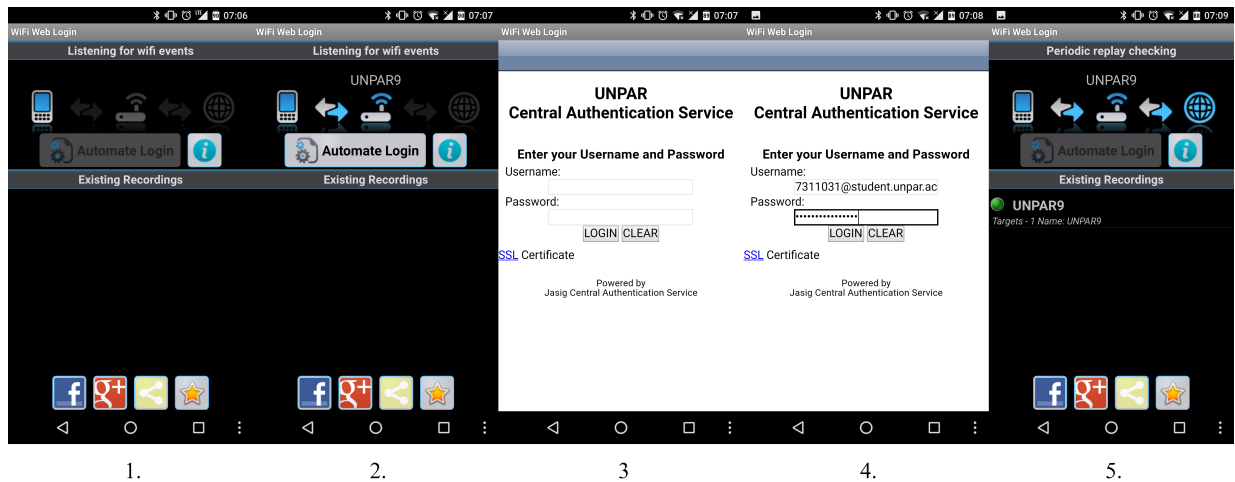
ANALISIS

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*



Gambar 3.1: Tampilan aplikasi *WiFi Web Login*.

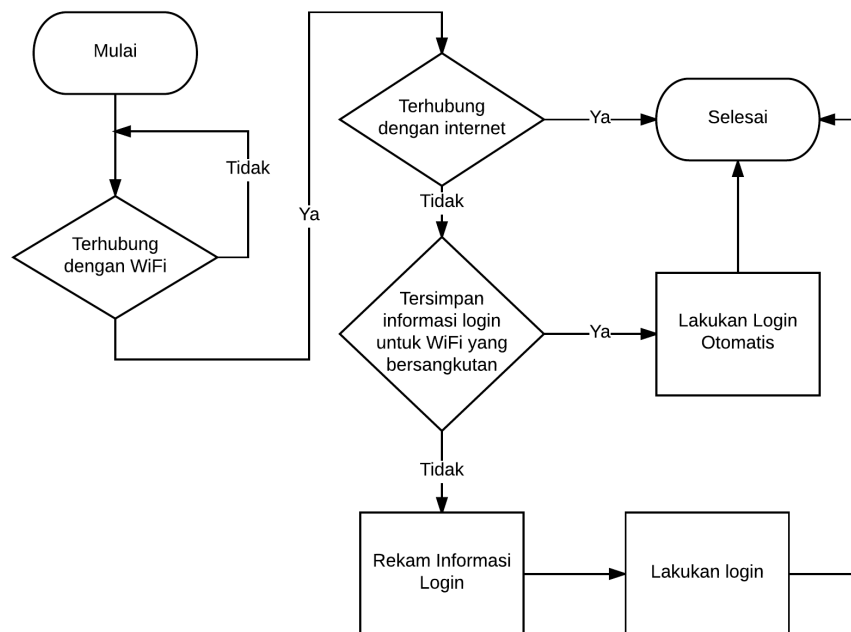
Pada gambar 3.1 diperlihatkan langkah-langkah login *captive portal* Unpar sebagai berikut:

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>

1 Diagram Alir *WiFi Web Login*

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



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

- 4 Berdasarkan gambar 3.2, langkah-langkah yang harus ditempuh untuk melakukan *login*
5 *wifi* berbasis web pada aplikasi ini adalah:

- 6 1. Deteksi sambungan dengan wifi yang bersangkutan.
- 7 2. Deteksi hubungan dengan internet.
- 8 3. Jika tidak terjadi hubungan dengan internet, deteksi apakah tersimpan informasi login
9 untuk wifi yang bersangkutan.
- 10 4. Jika terdapat informasi login untuk wifi yang bersangkutan maka lakukan login oto-
11 matis.
- 12 5. Jika tidak terdapat informasi login untuk wifi yang bersangkutan maka rekam infor-
13 masi login dan lakukan login.

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

17 3.2 Analisis Metode Penyimpanan Informasi Login

18 Penyimpanan informasi login dapat dilakukan dengan beberapa metode, diantaranya adalah
19 dengan menggunakan file teks atau menggunakan PasswordVault. Penyimpanan informasi
20 menggunakan file teks berarti informasi disimpan dalam bentuk *plaintext* dalam file yang
21 diberikan *access permission* tertentu. Sementara itu, penyimpanan informasi menggunakan

1 PasswordVault memanfaatkan kelas API yang terdapat pada *Universal Windows Platform*
2 (UWP).

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

8 PasswordVault dapat menyimpan informasi yang berisi *resource* (biasanya berupa na-
9 ma aplikasi atau string unik lainnya), username, dan password. Informasi yang perlu di-
10 simpan selain username dan password adalah *connection fingerprint* dan langkah-langkah
11 login. *Connection fingerprint* dapat disimpan pada resource karena sifatnya yang unik. Se-
12 mentara itu, langkah-langkah login dapat disimpan pada username atau password karena
13 langkah-langkah login dapat disimpan dalam bentuk String. Akan tetapi, langkah-langkah
14 login sebaiknya disimpan pada password, dipisahkan dengan karakter pemisah tertentu, agar
15 username dapat digunakan sebagai *identifier* unik. PasswordVault memiliki batasan 10 kre-
16 densial yang dapat disimpan per aplikasi. Jika aplikasi mencoba menyimpan lebih dari 10
17 kredensial maka akan terjadi exception. Oleh karena hal ini, PasswordVault menjadi pilihan
18 yang kurang baik untuk kebutuhan perangkat lunak pada penelitian ini.

19 Metode penyimpanan lainnya adalah dengan menggunakan file teks. Penyimpanan in-
20 formasi menggunakan file teks dapat dilakukan untuk informasi berbasis teks apapun dan
21 tidak ada batasan banyaknya informasi yang dapat disimpan (kecuali batasan perangkat
22 keras seperti kapasitas hard disk). Akan tetapi, file teks dapat dibaca oleh aplikasi mana-
23 pun, sehingga penyimpanan informasi sensitif tidak dapat dilakukan tanpa adanya metode
24 pengamanan tertentu. Salah satu metode pengamanan yang dapat dilakukan adalah de-
25 ngan mendeklarasikan *file access permission*. Akan tetapi, karena Windows memiliki *secu-*
26 *rity model* per pengguna dan bukan per aplikasi, maka aplikasi lain yang dijalankan oleh
27 pengguna tersebut memiliki akses yang sama kepada file yang bersangkutan. Metode pe-
28 ngamanan lainnya adalah dengan melakukan enkripsi pada file yang bersangkutan sehingga
29 hanya pemegang kunci yang dapat membaca file tersebut. Enkripsi file pada windows dapat
30 dilakukan menggunakan kelas CryptographicEngine. Kunci enkripsi dan dekripsi dapat di-
31 simpan menggunakan PasswordVault atau dengan meminta pengguna untuk memasukkan
32 kunci tersebut setiap kali aplikasi dijalankan.

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

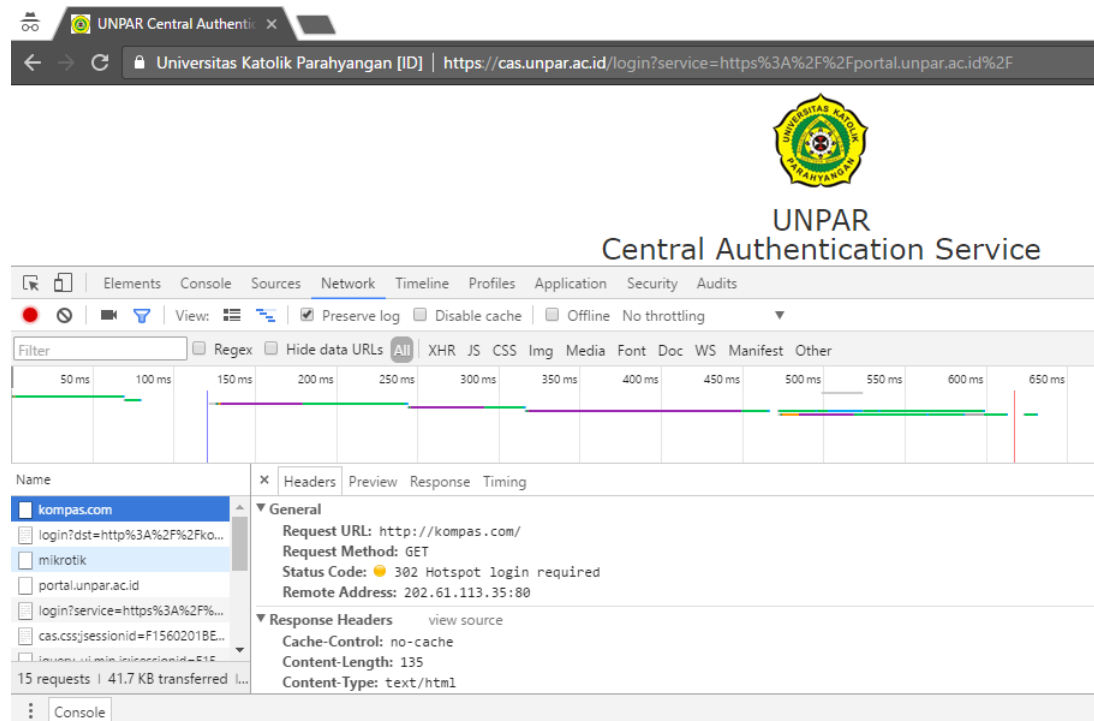
38 3.3 Analisis Metode Rekam dan Kirim Informasi Login

39 Kelas WebView pada *Universal Windows Platform* (UWP) hanya dapat dihubungkan de-
40 ngan kode C# menggunakan javascript. *Method* yang digunakan untuk melakukan eksekusi
41 javascript pada WebView adalah *InvokeScriptAsync*. Metode ini memiliki parameter string
42 berupa nama fungsi javascript yang ingin dipanggil dan array of string yang berisi argumen
43 yang ingin dikirimkan ke dalam fungsi tersebut. Salah satu fungsi yang dapat dipanggil
44 adalah *eval*. Dengan menggunakan *eval*, ekspresi javascript apapun dapat dijalankan pada
45 WebView. Untuk mengirimkan data dari javascript ke kode C#, dapat dijalankan fungsi
46 *window.external.notify* dengan parameter berupa string. Oleh karena itu, diperlukan *enco-*
47 *ding* tertentu (seperti JSON[9]) untuk memasukkan lebih dari satu argumen.

48 *InvokeScriptAsync* dapat digunakan untuk memanggil fungsi *eval* dengan parameter
49 berupa function yang dapat digunakan untuk menekan tombol atau memasukkan nilai pada
50 *text field* tertentu. Selain itu, dapat dimasukkan event listener yang dapat memanggil win-
51 *ding*. *external.notify* menggunakan cara ini. Fungsi *window.external.notify* dapat membantu

- 1 mengirimkan event-event seperti mouse click, keypress, atau perubahan nilai pada *text field*
- 2 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 dibutuhkan *redirection* yang dapat dikenali oleh *web browser* agar klien selalu diarahkan ke halaman *captive portal* tersebut sebelum melakukan otentikasi. Oleh karena itu, untuk setiap HTTP request yang dilakukan oleh klien sebelum melakukan otentikasi, HTTP response yang diberikan bukanlah berasal dari server tujuan. Sifat ini dapat dimanfaatkan untuk keperluan deteksi *captive portal* dengan mengirimkan request ke server yang sudah ditentukan sebelumnya, dan mendeteksi apakah response yang diberikan sesuai dengan harapan. Jika response yang diberikan tidak sesuai dengan harapan, maka dapat diasumsikan bahwa klien dibatasi oleh suatu *captive portal*.

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

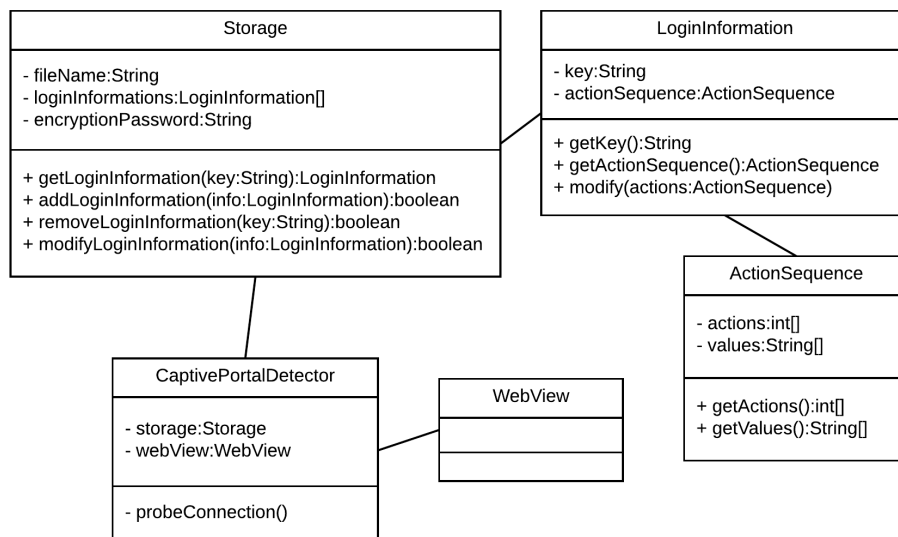
3.5 Analisis Perangkat Lunak

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:

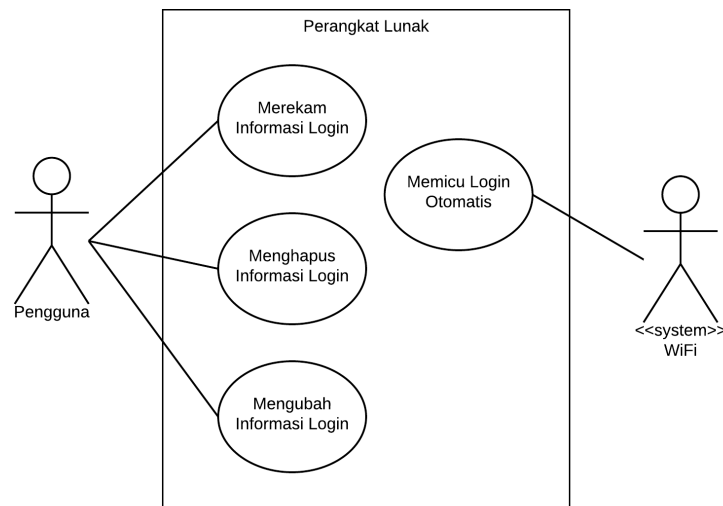
- **CaptivePortalDetector**
Kelas ini digunakan untuk mendeteksi keberadaan *captive portal*. Jika captive portal terdeteksi, maka informasi login yang tersimpan dalam Storage digunakan. Jika tidak terdapat informasi login dalam Storage, maka direkam informasi login baru.
- **Storage**
Kelas ini digunakan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

1 Skenario merekam informasi login

2 Nama: Merekam informasi login

3 Aktor: Pengguna

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

5 Deskripsi: Pengguna menyimpan informasi login.

6 Kondisi Akhir: Informasi login tersimpan di dalam perangkat lunak.

7

8 Skenario:

9 1. Pengguna memasukkan informasi login ke dalam form HTML.

10 2. Sistem menyimpan informasi login yang dimasukkan oleh pengguna.

11 Skenario menghapus informasi login

12 Nama: Menghapus informasi login

13 Aktor: Pengguna

14 Kondisi awal: Perangkat lunak sudah dijalankan.

15 Deskripsi: Pengguna menghapus informasi login.

16 Kondisi Akhir: Informasi login dihapus dari perangkat lunak.

17

18 Skenario:

19 1. Pengguna memilih untuk menghapus informasi login.

20 2. Sistem menghapus informasi login.

21 Skenario mengubah informasi login

22 Nama: Mengubah informasi login

23 Aktor: Pengguna

24 Kondisi awal: Perangkat lunak sudah dijalankan.

25 Deskripsi: Pengguna mengubah informasi login.

26 Kondisi Akhir: Informasi login diubah dari perangkat lunak.

27

28 Skenario:

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

5 **Skenario memicu login otomatis**

6 Nama: Memicu login otomatis

7 Aktor: WiFi

8 Kondisi awal: Perangkat lunak sudah dijalankan.

9 Deskripsi: Sistem memicu login otomatis berdasarkan perubahan status WiFi.

10 Kondisi Akhir: Klien terotentikasi pada *captive portal*.

11

12 Skenario:

- 13 1. Sistem mendeteksi adanya koneksi WiFi yang terjadi.
- 14 2. Sistem mendeteksi adanya *captive portal*.
- 15 3. Sistem mendeteksi adanya informasi login untuk *captive portal* tersebut.
- 16 4. Sistem mengirimkan informasi login kepada *captive portal*.
- 17 5. Sistem mendeteksi koneksi dengan internet dan klien sudah terotentikasi pada *captive*
- 18 *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:

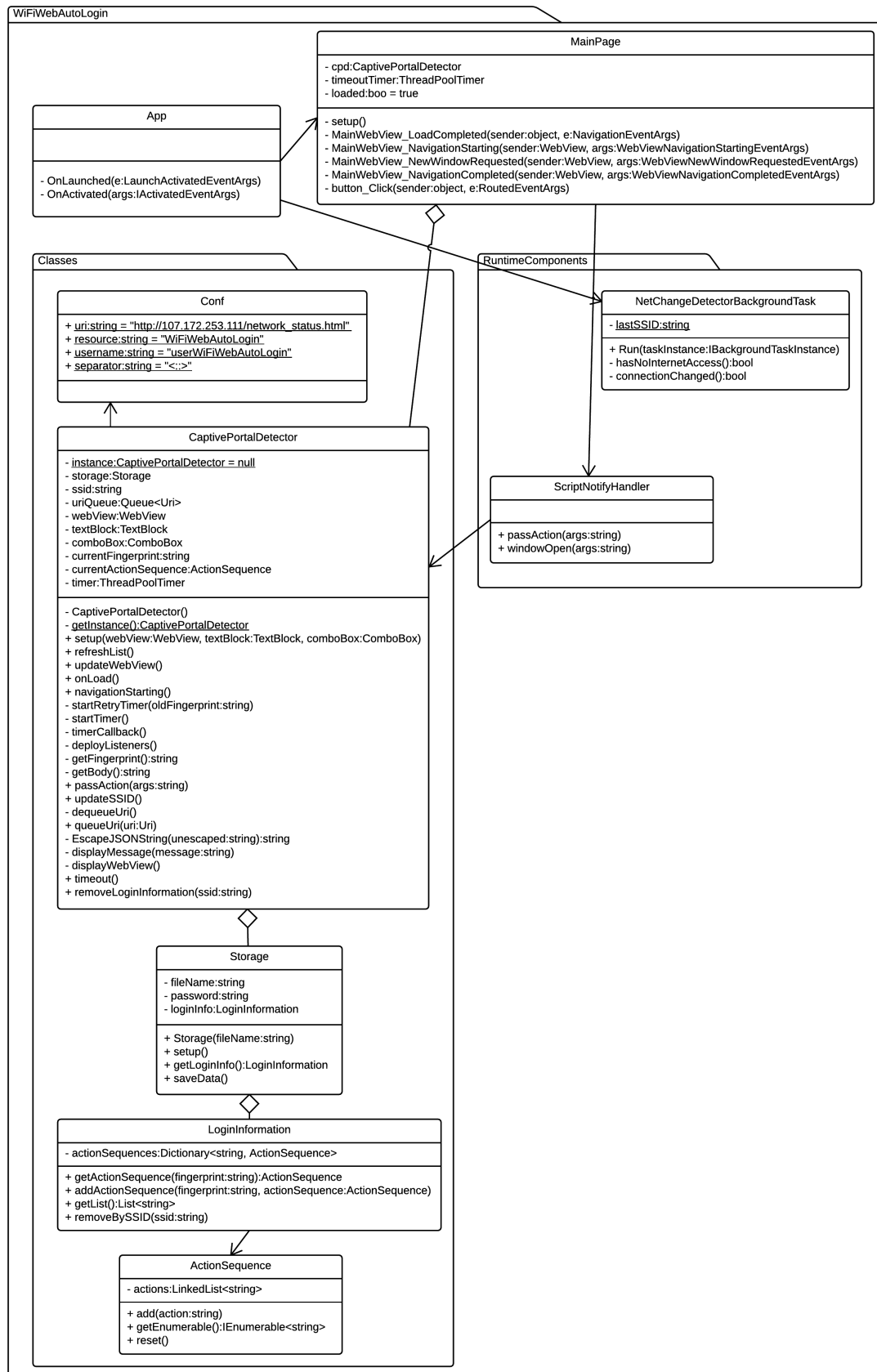
MainPage : Kelas ini merupakan kelas yang berperan sebagai tampilan utama aplikasi. Atribut-atribut yang dimiliki oleh kelas ini adalah:

- `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

- 1 melakukan *override* fungsi-fungsi JavaScript seperti `window.open()` dan `open()` agar
 2 bisa diakses dari JavaScript tanpa aksi langsung dari pengguna.



Gambar 4.1: Diagram Kelas Rinci.

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

- 5 • **lastSSID**
 6 Atribut ini menyimpan SSID terakhir yang nantinya akan dibandingkan dengan SSID
 7 terbaru untuk mendeteksi adanya perubahan SSID.

8 Metode-metode yang dimiliki oleh kelas ini adalah:

- 9 • **Run**
 10 Metode ini dipanggil saat Windows mengalami perubahan jaringan. Fungsinya adalah
 11 untuk menampilkan notifikasi apabila kondisi `connectionChanged() && lastSSID!=null`
 12 && `hasNoInternetAccess()` terpenuhi.
- 13 • **hasNoInternetAccess**
 14 Metode ini digunakan untuk mendeteksi tidak adanya akses internet menggunakan API
 15 yang diberikan oleh UWP.
- 16 • **connectionChanged**
 17 Metode ini digunakan untuk mendeteksi perubahan SSID.

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

- 21 • **passAction**
 22 Metode ini dipanggil saat *listener* yang sudah disisipkan ke dalam WebView mende-
 23 teksi *action* yang dapat direkam. *Action* yang direkam berupa teks yang berisi kode
 24 javascript yang dapat mereplikasi *action* tersebut.
- 25 • **windowOpen**
 26 Metode ini dipanggil saat javascript pada WebView memanggil fungsi `window.open`
 27 atau fungsi `open`.

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

- 34 • **instance**
 35 Atribut ini menyimpan *instance* `CaptivePortalDetector`.
- 36 • **storage**
 37 Atribut ini menyimpan objek `Storage` yang digunakan untuk menyimpan dan meng-
 38 akses informasi login.
- 39 • **ssid**
 40 Atribut ini menyimpan SSID saat ini.
- 41 • **uriQueue**
 42 Atribut ini menyimpan queue `Uri` yang perlu diakses.
- 43 • **webView**
 44 Atribut ini menyimpan `WebView` yang digunakan untuk melakukan deteksi *captive*
 45 *portal*.

- 1 • `textBlock`
2 Atribut ini menyimpan `TextBlock` yang digunakan untuk menampilkan pesan kepada
3 pengguna.
- 4 • `comboBox`
5 Atribut ini menyimpan `ComboBox` yang digunakan untuk menampilkan daftar SSID
6 yang sudah tersimpan kepada pengguna.
- 7 • `currentFingerprint`
8 Atribut ini menyimpan fingerprint saat ini.
- 9 • `currentActionSequence`
10 Atribut ini menyimpan `ActionSequence` yang terkait dengan fingerprint saat ini.
- 11 • `timer`
12 Atribut ini menyimpan timer yang digunakan untuk mengatur waktu akses Uri dalam
13 `uriQueue`.

14 Metode-metode yang dimiliki oleh kelas ini adalah:

- 15 • `getInstance`
16 Metode ini digunakan untuk mendapatkan *instance* dari `CaptivePortalDetector`.
- 17 • `setup`
18 Metode ini digunakan untuk melakukan *setup* awal yang menyimpan `WebView`, `TextBlock`, dan `ComboBox` ke dalam *instance* `CaptivePortalDetector`.
- 19 • `refreshList`
20 Metode ini digunakan untuk melakukan *refresh* tampilan `ComboBox`.
- 21 • `updateWebView`
22 Metode ini digunakan untuk menentukan melakukan deteksi *captive portal* jika ter-
23 hubung dengan koneksi WiFi, atau menampilkan pesan kepada pengguna jika tidak
24 terhubung dengan koneksi WiFi.
25 • `onLoad`
26 Metode ini dipanggil saat `WebView` sudah selesai melakukan *loading* halaman. Fung-
27 sinya adalah untuk melakukan `deployListener()`, menjalankan aksi-aksi yang sudah
28 terekam pada informasi login, dan mendeteksi sudah atau belum terjadinya koneksi
29 dengan internet.
30 • `navigationStarting`
31 Metode ini dipanggil saat `WebView` mulai melakukan navigasi ke halaman baru. Fung-
32 sinya adalah untuk membatalkan timer untuk membuka halaman-halaman popup dari
33 halaman sebelumnya.
34 • `passAction`
35 Metode ini digunakan untuk menyimpan *action* yang dikirimkan oleh `ScriptNotifyHan-`
36 dler ke dalam `currentActionSequence`.
37 • `updateSSID`
38 Metode ini digunakan untuk mendapatkan SSID terbaru.
39 • `queueUri`
40 Metode ini digunakan untuk memasukkan Uri baru ke dalam `uriQueue`.
41 • `timeout`
42 Metode ini digunakan untuk menyatakan bahwa terjadi *connection timeout*.
43

1 • `removeLoginInformation`
2 Metode ini digunakan untuk menghapus seluruh informasi login yang terkait dengan
3 SSID tertentu.

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

6 • `fileName`
7 Atribut ini menyimpan nama file yang digunakan untuk menyimpan informasi login
8 yang terenkripsi.

9 • `password`
10 Atribut ini menyimpan password yang digunakan untuk melakukan enkripsi.

11 • `loginInfo`
12 Atribut ini menyimpan objek `LoginInformation` yang digunakan untuk menyimpan
13 seluruh informasi login.

14 Metode-metode yang dimiliki oleh kelas ini adalah:

15 • `setup`
16 Metode ini digunakan untuk melakukan *setup* awal seperti membuka file dan melakuk-
17 an dekripsi.

18 • `getLoginInfo`
19 Metode ini digunakan untuk mendapatkan objek `LoginInformation`.

20 • `saveData`
21 Metode ini digunakan untuk menyimpan data yang ada pada objek `LoginInformation`
22 ke dalam file dan melakukan enkripsi pada file tersebut.

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

25 • `actionSequences`
26 Atribut ini merupakan pasangan *key-value* antara suatu fingerprint dengan `ActionSe-`
27 `quence`.

28 Metode-metode yang dimiliki oleh kelas ini adalah:

29 • `getActionSequence`
30 Metode ini digunakan untuk mendapatkan `ActionSequence` berdasarkan fingerprint
31 tertentu.

32 • `addActionSequence`
33 Metode ini digunakan untuk menambahkan `ActionSequence` untuk fingerprint tertentu.

34 • `removeBySSID`
35 Metode ini digunakan untuk menghapus `ActionSequence` milik fingerprint tertentu.

36 • `getList`
37 Metode ini digunakan untuk mendapatkan daftar SSID yang sudah direkam.

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

40 • `actions`
41 Atribut ini merupakan daftar *action* yang berupa kode javascript.

42 Metode-metode yang dimiliki oleh kelas ini adalah:

- 1 • add
- 2 Metode ini digunakan untuk menambahkan *action* ke dalam daftar ini.
- 3 • getEnumerable
- 4 Metode ini digunakan untuk mendapatkan enumerable dari daftar *actions*, sehingga
- 5 mempermudah eksekusi *actions*.
- 6 • reset
- 7 Metode ini digunakan untuk menghapus seluruh *action* yang ada pada daftar ini.

8 4.2 Perancangan Algoritma dan Struktur Data

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

12 4.2.1 Algoritma Deteksi *Captive Portal*

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

```

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

```

17 Algoritma di atas menjelaskan deteksi *captive portal* dilakukan dengan melakukan deteksi
 18 jaringan yang tidak terhubung dengan internet. Jika ditemukan jaringan yang tidak terhu-
 19 bung dengan internet, maka akan muncul notifikasi yang memungkinkan pengguna untuk
 20 menjalankan perangkat lunak. Saat perangkat lunak dijalankan, perangkat lunak akan men-
 21 coba untuk mengakses halaman panicing yang beralamatkan pada http://107.172.253.111/network_status.
 22 Jika didapat respon HTTP yang berupa *redirect*, maka pada jaringan tersebut terdapat *cap-*
 23 *tive portal*. Algoritma ini akan didaftarkan pada sistem saat perangkat lunak pertama kali
 24 dijalankan dan dipanggil saat ada perubahan status jaringan.

25 4.2.2 Struktur Data dan Format *Fingerprint*

26 *Fingerprint* suatu halaman terdiri dari SSID, uri, serta isi *tag* head pada halaman tersebut.
 27 Data ini disimpan dalam satu buah string dan dipisahkan oleh separator "<::>" (tanpa tan-
 28 da petik). Salah satu contoh *fingerprint* adalah UNPAR9<::><https://cas.unpar.ac.id/login><::><title>Hal
 29 Login</title> yang memiliki arti bahwa *fingerprint* tersebut berasal dari WiFi dengan:

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

4.2.3 Struktur Data LoginInformation

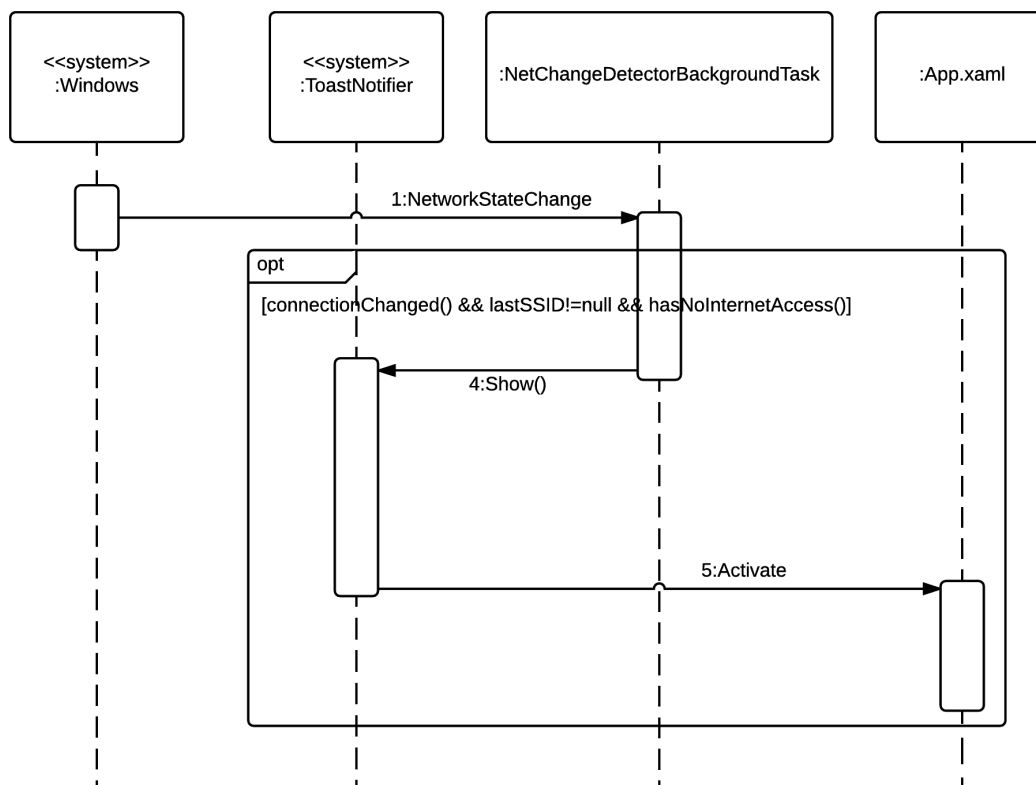
Kelas LoginInformation memiliki daftar objek bertipe ActionSequence yang disimpan pada properti actionSequences bertipe Dictionary. Kelas Dictionary dapat menyimpan data yang berupa pasangan *key-value*. *Key* yang digunakan bertipe string dan merupakan *fingerprint* suatu halaman. *Value* yang disimpan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



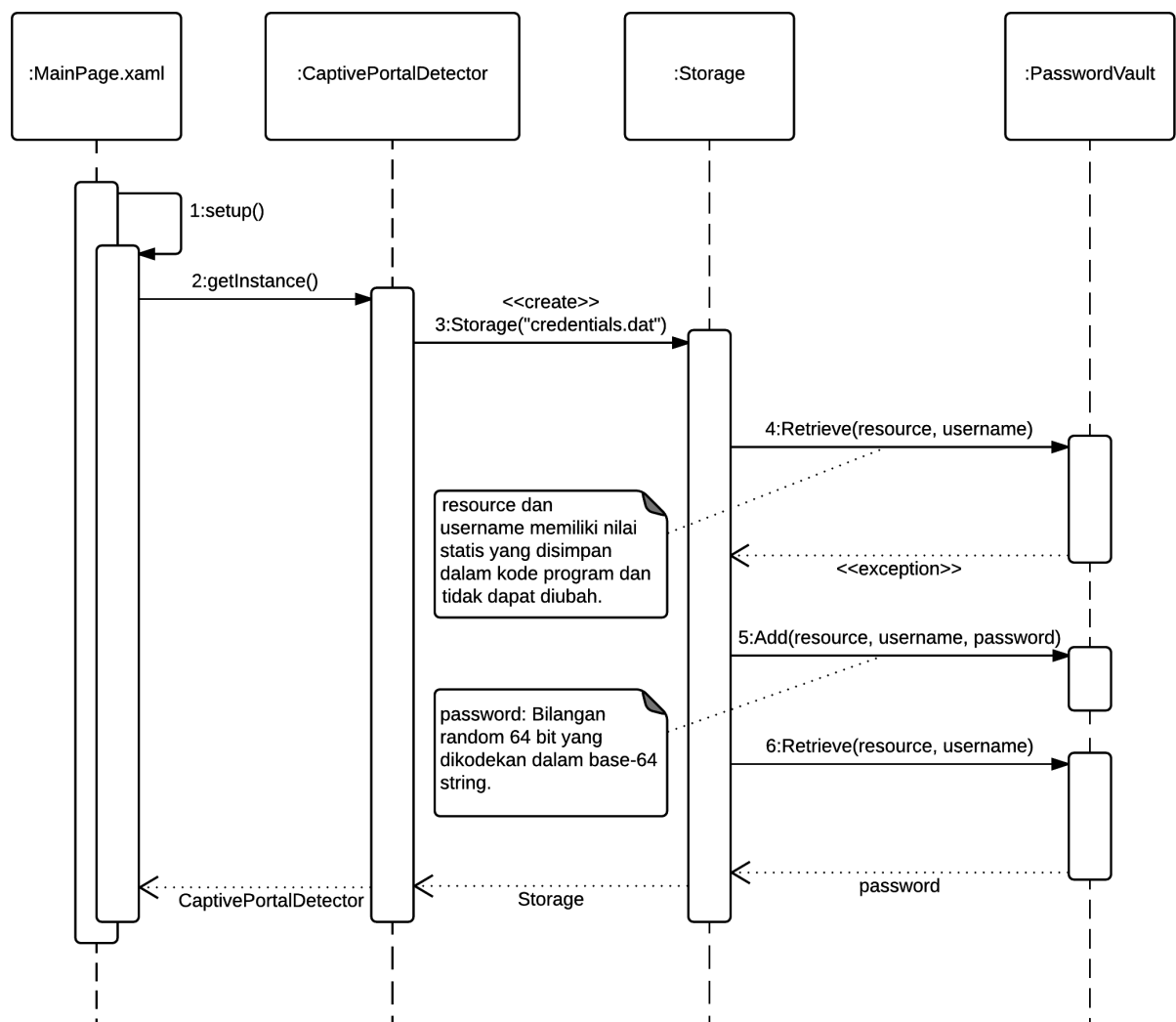
Gambar 4.2: Diagram Interaksi Deteksi Jaringan.

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

- 1 akan diaktifkan oleh Windows, dan objek `NetChangeDetectorBackgroundTask` yang
- 2 sudah didaftarkan akan menerima *trigger* tersebut.
- 3 2. Jika kondisi `connectionChanged() && lastSSID!=null && hasNoInternetAccess()`
- 4 terpenuhi, maka:
- 5 (a) `NetChangeDetectorBackgroundTask` memerintahkan `ToastNotifier` untuk memun-
- 6 culkan notifikasi menggunakan method `Show()`.
- 7 (b) Saat user menekan tombol "Yes" pada notifikasi, `App.xaml` diaktivasi.

8 4.3.2 Perancangan Interaksi Penciptaan Password



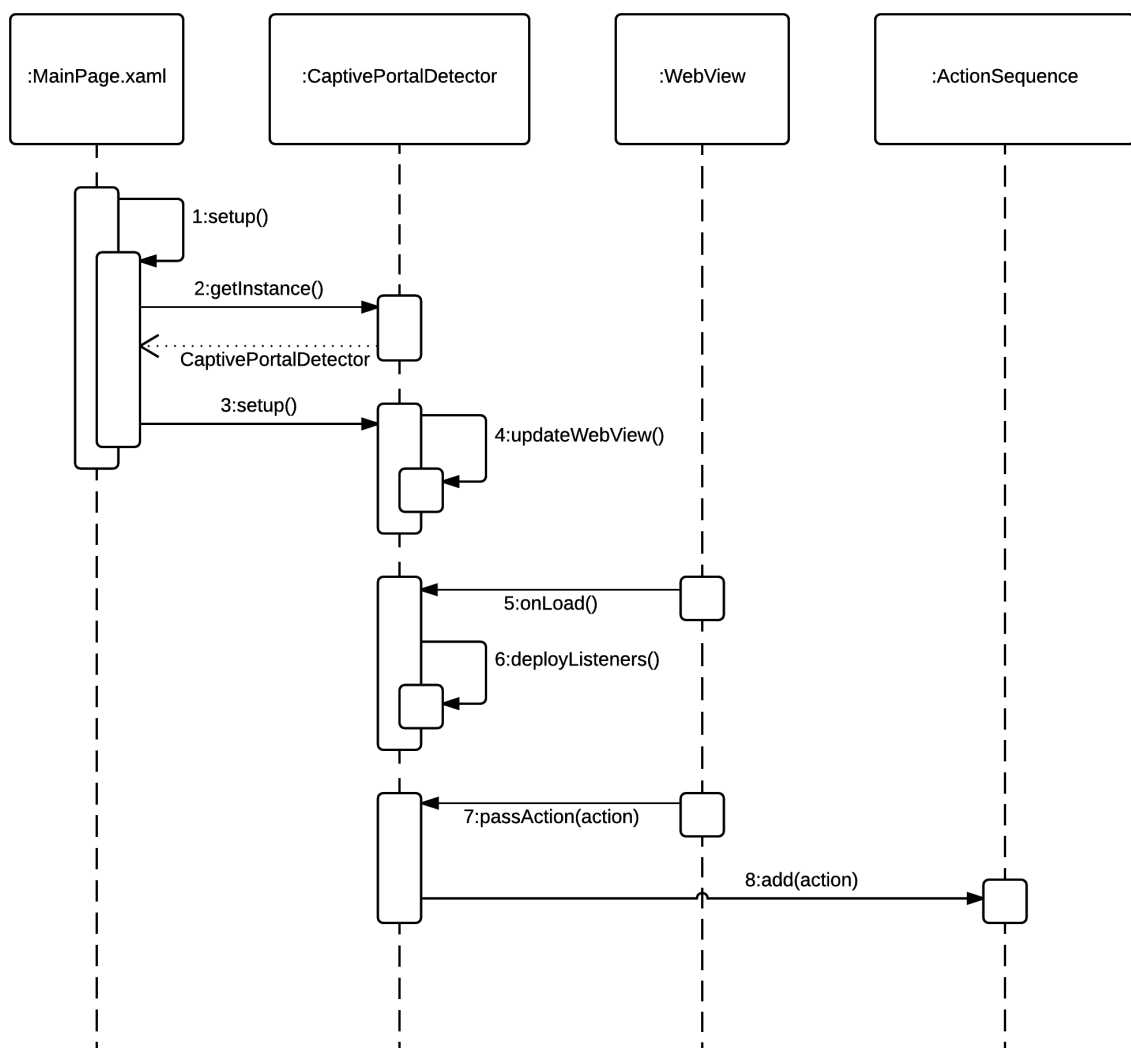
Gambar 4.3: Diagram Interaksi Penciptaan Password.

9 Gambar 4.3 menjelaskan mengenai interaksi antar objek dalam perangkat lunak untuk
 10 menciptakan password random saat perangkat lunak pertama kali dijalankan. Interaksi yang
 11 terjadi adalah sebagai berikut:

- 12 1. `MainPage.xaml` melakukan `setup()`.
- 13 2. `MainPage.xaml` memanggil metode `getInstance()` pada `CaptivePortalDetector` untuk
- 14 mendapatkan *instance* `CaptivePortalDetector`.

- 1 3. CaptivePortalDetector menciptakan objek Storage baru pada *constructor*-nya.
- 2 4. Objek Storage berusaha untuk mendapatkan password dengan memanggil metode Re-
- 3 trieve() pada objek PasswordVault, namun mendapatkan exception karena belum ada
- 4 password yang disimpan.
- 5 5. Objek Storage memasukkan password baru yang diciptakan secara random menggu-
- 6 nakan metode Add() pada PasswordVault.
- 7 6. Objek Storage memanggil metode Retrieve() kembali pada objek PasswordVault, dan
- 8 mendapatkan password yang baru saja diciptakan. Setelah itu, CaptivePortalDetector
- 9 mendapatkan objek Storage, dan MainPage.xaml mendapatkan *instance* CaptivePor-
- 10 talDetector.

11 4.3.3 Perancangan Interaksi Penyimpanan Informasi Login



Gambar 4.4: Diagram Interaksi Penyimpanan Informasi Login.

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

- 14 1. MainPage.xaml melakukan `setup()`.

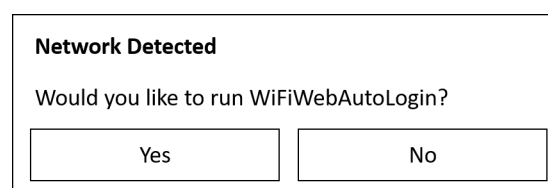
2. MainPage.xaml memanggil metode `getInstance()` pada kelas `CaptivePortalDetector` untuk mendapatkan *instance* `CaptivePortalDetector`. MainPage.xaml mendapatkan *instance* `CaptivePortalDetector`.
3. MainPage.xaml memanggil metode `setup()` pada objek `CaptivePortalDetector`.
4. `CaptivePortalDetector` 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



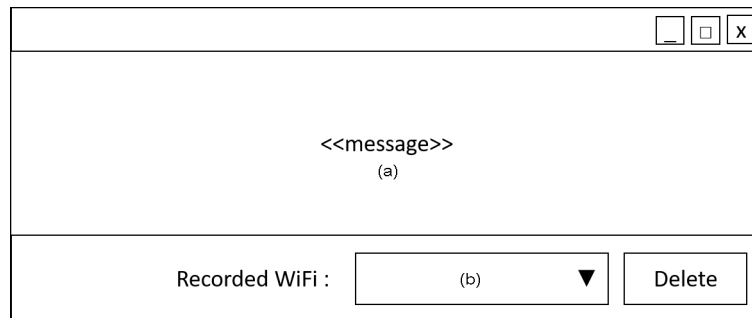
Gambar 4.5: Rancangan Antarmuka Notifikasi.

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 Web-View saat berada dalam mode web browser, dan menjadi Label saat berada dalam mode message box.

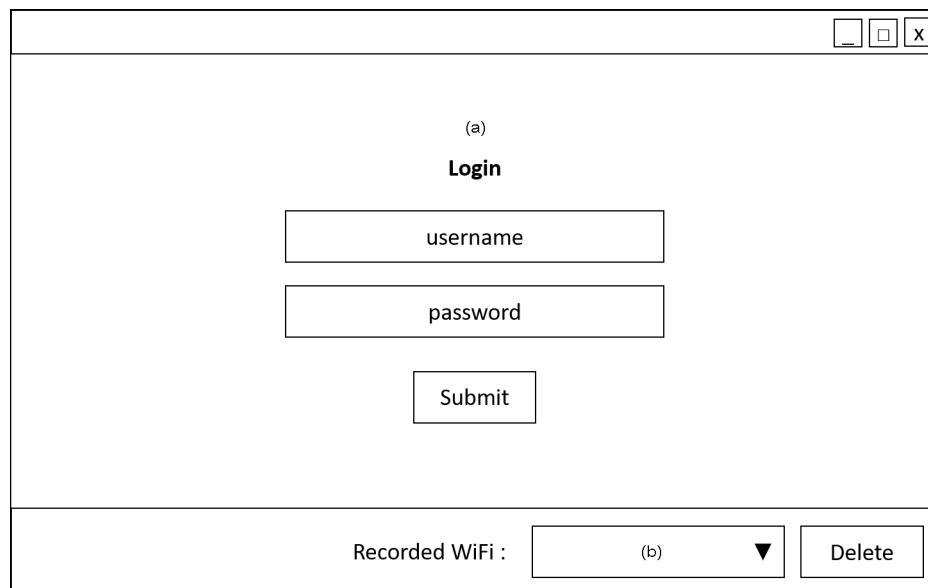
Antarmuka Message Box



Gambar 4.6: Rancangan Antarmuka Message Box.

Gambar 4.6 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 IDE¹ Microsoft Visual Studio Community 2015.

5.2 Masalah Implementasi dan Solusinya

Terdapat beberapa masalah implementasi yang membuat rancangan perangkat lunak tidak dapat sepenuhnya didasarkan pada hasil analisis, di antaranya:

- Fungsi `window.external.notify` tidak berperilaku sebagaimana yang diperkirakan. Fungsi ini diharapkan dapat dipanggil secara langsung di dalam kode javascript, namun ternyata tidak bisa. Setiap halaman yang ingin memanfaatkan fungsi ini harus didaftarkan pada `Package.appxmanifest`. Metode yang digunakan untuk mendapatkan hasil yang sama dengan fungsi yang diberikan oleh `window.external.notify` adalah dengan menggunakan kelas bertipe `RuntimeComponent` yang diizinkan untuk dapat diakses oleh JavaScript pada `WebView`. Kelas ini adalah `ScriptNotifyHandler` pada gambar 4.1.
- Fungsi `window.open` dan fungsi `open` tidak dapat dijalankan secara otomatis sehingga popup tidak muncul. Metode yang digunakan untuk mendapatkan hasil yang sama dari yang direncanakan sebelumnya adalah dengan melakukan *override* fungsi `window.open` dan fungsi `open` pada saat halaman selesai dimuat, dan menghubungkannya dengan kelas `ScriptNotifyHandler`. Akan tetapi, metode ini masih kurang memadai karena kode JavaScript yang memanggil fungsi-fungsi tersebut pada saat halaman dimuat akan dieksekusi sebelum *override* terjadi.

¹Integrated Development Environment

1 Perancangan yang tertulis pada bab 4 sudah merupakan hasil revisi dari analisis masalah
2 implementasi ini.

3 5.3 Pengujian Fungsional

4 Pengujian fungsional dilakukan pada jaringan WiFi di kost di jalan Ciumbuleuit nomor 149,
5 Bandung, dengan SSID "C149Net". Pengujian fungsional dilakukan pada tanggal 4 April
6 2017.

7 5.3.1 Rencana Pengujian Fungsional

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

- 11 • Deteksi perubahan jaringan.
- 12 • Deteksi *captive portal*.
- 13 • Login otomatis.

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

15 5.3.2 Hasil Pengujian Fungsional

16 Hasil-hasil pengujian fungsional adalah sebagai berikut:

- 17 • **Pengujian deteksi perubahan jaringan**

- 18 – **Pengujian positif**

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

21 **Hasil yang diharapkan:** Muncul notifikasi.

22 **Hasil yang didapatkan:** Muncul notifikasi "Network Detected" dengan pesan
23 "Would you like to run WiFiWebAutoLogin?".

24 **Kesimpulan:** Fungsi berjalan sesuai harapan.

- 25 – **Pengujian negatif**

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

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

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

30 **Kesimpulan:** Fungsi berjalan sesuai harapan.

- 31 • **Pengujian deteksi *captive portal***

- 32 – **Pengujian positif**

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

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

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

37 **Kesimpulan:** Fungsi berjalan sesuai harapan.

- 38 – **Pengujian negatif**

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

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

- 1 **Hasil yang didapatkan:** Muncul pesan "Operation timeout. Check your network connection."
- 2 **Kesimpulan:** Fungsi berjalan sesuai harapan.
- 3
- 4 • **Pengujian login otomatis**
- 5 – **Pengujian positif**
- 6 **Kasus:** Menghubungkan komputer dengan WiFi yang terhubung dengan *captive portal* yang sudah pernah dijalankan login secara manual.
- 7 **Hasil yang diharapkan:** Muncul pesan "Connected."
- 8 **Hasil yang didapatkan:** Muncul pesan "Executing recorded actions...", lalu setelah beberapa saat, muncul pesan "Connected."
- 9 **Kesimpulan:** Fungsi berjalan sesuai harapan.
- 10
- 11 – **Pengujian negatif**
- 12 **Kasus:** Menghubungkan komputer dengan WiFi yang terhubung dengan *captive portal* yang belum pernah dijalankan login secara manual.
- 13 **Hasil yang diharapkan:** Muncul halaman login.
- 14 **Hasil yang didapatkan:** Muncul halaman login *captive portal*.
- 15 **Kesimpulan:** Fungsi berjalan sesuai harapan.
- 16
- 17

18 5.4 Pengujian Eksperimental

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

21 5.4.1 Rencana Pengujian Eksperimental

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

- 23 • *C149Net* pada kost di jalan Ciumbuleuit nomor 149, Bandung.
- 24 • *Starbucks@wifi.id* pada Starbucks Dipatiukur, Bandung.
- 25 • *wifi.id* pada Starbucks Dipatiukur, Bandung.
- 26 • *UNPAR9* pada gedung 10 Universitas Katolik Parahyangan, Bandung.
- 27 • *FTIS.cisco* pada gedung 9 Universitas Katolik Parahyangan, Bandung.

28 5.4.2 Hasil Pengujian Eksperimental

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

32 Hasil Pengujian WiFi C149Net

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

1 Hasil Pengujian WiFi Starbucks@wifi.id

2 Pengujian pada WiFi dengan SSID Starbucks@wifi.id yang berlokasi pada Starbucks Di-
3 patiukur, Bandung, mendapatkan hasil sesuai harapan. Notifikasi muncul pada saat WiFi
4 pertama kali terhubung. Halaman *captive portal* muncul setelah tombol "Yes" pada noti-
5 fikasi ditekan. Setelah menekan tombol "continue" pada halaman tersebut, lalu menekan
6 tombol "lanjutkan" pada halaman selanjutnya, pesan "Connected." muncul. Jika informasi
7 login sudah tersimpan, pesan "Connected." akan langsung muncul setelah menekan tombol
8 "Yes" pada notifikasi.

9 Hasil Pengujian WiFi wifi.id

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

15 Hasil Pengujian WiFi UNPAR9

16 Pengujian pada WiFi dengan SSID UNPAR9 yang berlokasi pada gedung 10 Universitas
17 Katolik Parahyangan, Bandung, mendapatkan hasil yang tidak sesuai harapan. Notifika-
18 si muncul pada saat WiFi pertama kali terhubung. Halaman login muncul setelah tom-
19 bol "Yes" pada notifikasi ditekan. Setelah login dilakukan, proses terhenti pada halaman
20 <https://portal.unpar.ac.id/home>. Hal ini dikarenakan WiFi di Universitas Katolik Para-
21 hyangan menggunakan CAS². Selain itu, CAS ini menggunakan *pop-up* untuk menampilkan
22 halaman <https://wireless.unpar.ac.id/status> dan halaman tersebut adalah halaman
23 yang membuka popup yang menuju ke halaman tujuan. WebView pada UWP tidak mem-
24 perbolehkan pemanggilan fungsi `open()`, `window.open()`, `el.click()` dan `form.submit()` yang
25 bukan merupakan aksi langsung oleh pengguna. *Override* tiap fungsi tersebut dimungkinkan
26 setelah halaman selesai dimuat, namun itu berarti fungsi hasil *override* tidak akan digunak-
27 an jika fungsi tersebut dipanggil pada saat halaman pertama kali dimuat. Hal ini mencakup
28 *onload event* dan script yang dipanggil langsung pada *script tag* baik pada *body* maupun
29 *head*.

30 Hasil Pengujian WiFi FTIS.cisco

31 Pengujian pada WiFi dengan SSID FTIS.cisco yang berlokasi pada gedung 9 Universitas
32 Katolik Parahyangan, Bandung, mendapatkan hasil yang tidak sesuai harapan. Hal ini
33 terjadi karena WiFi ini merupakan jaringan internal Fakultas Teknologi dan Sains yang
34 menggunakan jaringan WiFi Universitas Katolik Parahyangan untuk akses internet. Proses
35 terhenti pada halaman yang sama dengan WiFi UNPAR9 dan penyebab terjadinya hal ini
36 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].

1

LAMPIRAN A

2

KODE SUMBER

3

A.1 Namespace: WiFiWebAutoLogin

```

WiFiWebAutoLogin/Package.appxmanifest
1 <?xml version="1.0" encoding="utf-8"?>
2 <Package xmlns="http://schemas.microsoft.com/appx/manifest/foundation/windows10"
  ↳ ows10"
  ↳ xmlns:mp="http://schemas.microsoft.com/appx/2014/phone/manifest"
  ↳ xmlns:uap="http://schemas.microsoft.com/appx/manifest/uap/windows10"
  ↳ IgnorableNamespaces="uap mp">
3 <Identity Name="24a4f351-3e45-4b12-946e-bfc3593444b4"
  ↳ Publisher="CN=Yohanes Mario" Version="1.0.0.0" />
4 <mp:PhoneIdentity PhoneProductId="24a4f351-3e45-4b12-946e-bfc3593444b4"
  ↳ PhonePublisherId="00000000-0000-0000-0000-000000000000" />
5 <Properties>
6   <DisplayName>WiFiWebAutoLogin</DisplayName>
7   <PublisherDisplayName>Yohanes Mario</PublisherDisplayName>
8   <Logo>Assets\StoreLogo.png</Logo>
9 </Properties>
10 <Dependencies>
11   <TargetDeviceFamily Name="Windows.Universal" MinVersion="10.0.0.0"
  ↳ MaxVersionTested="10.0.0.0" />
12 </Dependencies>
13 <Resources>
14   <Resource Language="x-generate" />
15 </Resources>
16 <Applications>
17   <Application Id="App" Executable="$targetnametoken$.exe"
  ↳ EntryPoint="WiFiWebAutoLogin.App">
18     <uap:VisualElements DisplayName="WiFiWebAutoLogin"
  ↳ Square150x150Logo="Assets\Square150x150Logo.png"
  ↳ Square44x44Logo="Assets\Square44x44Logo.png"
  ↳ Description="WiFiWebAutoLogin" BackgroundColor="transparent">
19       <uap:DefaultTile Wide310x150Logo="Assets\Wide310x150Logo.png">
20         </uap:DefaultTile>
21       <uap:SplashScreen Image="Assets\SplashScreen.png" />
22     </uap:VisualElements>
23     <Extensions>
24       <Extension Category="windows.backgroundTasks" EntryPoint="WiFiWebA
  ↳ utoLogin.RuntimeComponents.CustomBackgroundTask">
25         <BackgroundTasks>
26           <Task Type="systemEvent" />
27         </BackgroundTasks>
28       </Extension>

```

```

29     </Extensions>
30 </Application>
31 </Applications>
32 <Capabilities>
33     <Capability Name="internetClient" />
34 </Capabilities>
35 </Package>

```

1 _____ WiFiWebAutoLogin/App.xaml _____

```

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

```

2 _____ WiFiWebAutoLogin/App.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 Windows.ApplicationModel;
7 using Windows.ApplicationModel.Activation;
8 using Windows.ApplicationModel.Background;
9 using Windows.ApplicationModel.Core;
10 using Windows.Foundation;
11 using Windows.Foundation.Collections;
12 using Windows.UI.Notifications;
13 using Windows.UI.Xaml;
14 using Windows.UI.Xaml.Controls;
15 using Windows.UI.Xaml.Controls.Primitives;
16 using Windows.UI.Xaml.Data;
17 using Windows.UI.Xaml.Input;
18 using Windows.UI.Xaml.Media;
19 using Windows.UI.Xaml.Navigation;
20
21 namespace WiFiWebAutoLogin
22 {
23     /// <summary>
24     /// Provides application-specific behavior to supplement the default
25         ↪ Application class.
26     /// </summary>
27     sealed partial class App : Application
28     {
29         /// <summary>
30         /// Initializes the singleton application object. This is the
31             ↪ first line of authored code
32         /// executed, and as such is the logical equivalent of main() or
33             ↪ 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
40     ///   ↪ user. Other entry points
41     /// will be used such as when the application is launched to open
42     ///   ↪ a specific file.
43     /// </summary>
44     /// <param name="e">Details about the launch request and
45     ///   ↪ process.</param>
46     protected override void OnLaunched(LaunchActivatedEventArgs e)
47     {
48         // Initialize background task
49         bool taskRegistered = false;
50         string taskName = "CustomBackgroundTask";
51
52         foreach (var task in BackgroundTaskRegistration.AllTasks) {
53             if (task.Value.Name == taskName) {
54                 task.Value.Unregister(true);
55             }
56         }
57
58         if (!taskRegistered) {
59             // Register task
60             var builder = new BackgroundTaskBuilder();
61
62             builder.Name = taskName;
63             builder.TaskEntryPoint = "WiFiWebAutoLogin.RuntimeComponen_
64             ↪ ts.CustomBackgroundTask";
65             builder.SetTrigger(new
66             ↪ SystemTrigger(SystemTriggerType.NetworkStateChange,
67             ↪ false));
68             builder.AddCondition(new
69             ↪ SystemCondition(SystemConditionType.UserPresent));
70             BackgroundTaskRegistration task = builder.Register();
71         }
72
73         #if DEBUG
74         if (System.Diagnostics.Debugger.IsAttached) {
75             //this.DebugSettings.EnableFrameRateCounter = true;
76             this.DebugSettings.EnableFrameRateCounter = false;
77         }
78         #endif
79
80         Frame rootFrame = Window.Current.Content as Frame;
81
82         // Do not repeat app initialization when the Window already
83         ↪ has content,
84         // just ensure that the window is active
85         if (rootFrame == null) {

```

```

76         // Create a Frame to act as the navigation context and
77         ↪ navigate to the first page
78         rootFrame = new Frame();
79
80         rootFrame.NavigationFailed += OnNavigationFailed;
81
82         //if (e.PreviousExecutionState ==
83         ↪ ApplicationExecutionState.Terminated) {
84             //TODO: Load state from previously suspended
85             ↪ application
86         //}
87
88         // Place the frame in the current Window
89         Window.Current.Content = rootFrame;
90     }
91
92     //if (e.PrelaunchActivated == false) {
93         //if (rootFrame.Content == null) {
94             // When the navigation stack isn't restored navigate
95             ↪ to the first page,
96             // configuring the new page by passing required
97             ↪ information as a navigation
98             // parameter
99             rootFrame.Navigate(typeof(MainPage), e.Arguments);
100         //}
101         // Ensure the current window is active
102         Window.Current.Activate();
103     //}
104 }
105
106 protected override void OnActivated(IActivatedEventArgs args) {
107     if (args.Kind == ActivationKind.ToastNotification) {
108         var toastArgs = args as
109         ↪ ToastNotificationActivatedEventArgs;
110         var arguments = toastArgs.Argument;
111
112         if (arguments == "Yes") {
113             Frame rootFrame = Window.Current.Content as Frame;
114             if (rootFrame == null) {
115                 rootFrame = new Frame();
116                 Window.Current.Content = rootFrame;
117             }
118             rootFrame.Navigate(typeof(MainPage));
119             Window.Current.Activate();
120         }
121         else {
122             CoreApplication.Exit();
123         }
124     }
125
126     ToastNotificationManager.History.Remove("WWAL_TOAST");
127 }

```

```

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 " +
131             ↪ e.SourcePageType.FullName);
132     }
133
134     /// <summary>
135     /// Invoked when application execution is being suspended.
136     ↪ Application state is saved
137     /// without knowing whether the application will be terminated or
138     ↪ resumed with the contents
139     of memory still intact.
140     /// </summary>
141     /// <param name="sender">The source of the suspend request.</param>
142     /// <param name="e">Details about the suspend request.</param>
143     private void OnSuspending(object sender, SuspendingEventArgs e)
144     {
145         var deferral = e.SuspendingOperation.GetDeferral();
146         //TODO: Save application state and stop any background activity
147         deferral.Complete();
148     }
149 }
150 }

```

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

```

```

13         <WebView Name="MainWebView"
            ↪ NewWindowRequested="MainWebView_NewWindowRequested"
            ↪ DOMContentLoaded="MainWebView_DOMContentLoaded"
            ↪ ContentLoading="MainWebView_ContentLoading" LongRunningScriptDetected="
            ↪ 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 _____
2  using System;
3  using System.Collections.Generic;
4  using System.IO;
5  using System.Linq;
6  using System.Runtime.InteropServices.WindowsRuntime;
7  using System.Threading.Tasks;
8  using Windows.Foundation;
9  using Windows.Foundation.Collections;
10 using Windows.UI.Xaml;
11 using Windows.UI.Xaml.Controls;
12 using Windows.UI.Xaml.Controls.Primitives;
13 using Windows.UI.Xaml.Data;
14 using Windows.UI.Xaml.Input;
15 using Windows.UI.Xaml.Media;
16 using Windows.UI.Xaml.Navigation;
17 using Windows.Storage;
18 using System.Reflection;
19 using WiFiWebAutoLogin.RuntimeComponents;
20 using Windows.UI.ViewManagement;
21 using WiFiWebAutoLogin.Classes;
22 using System.Diagnostics;
23 using Windows.System.Threading;
24
25 // The Blank Page item template is documented at
26 ↪ 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             ↪         +
97             "var open = window.open;" +
98             "document.open = window.open;"
99         });
100     }
101 }
102
103 private void button_Click(object sender, RoutedEventArgs e) {
104     cpd.removeLoginInformation((string)comboBox.SelectedItem);
105 }
106
107 private void MainWebView_DOMContentLoaded(Webview sender,
       ↪     WebviewDOMContentLoadedEventArgs args) {
108     // DISABLED
109 }
110
111 private void MainWebView_ContentLoading(Webview sender,
       ↪     WebviewContentLoadingEventArgs args) {
112     // DISABLED
113 }

```



```

114     private void MainWebView_LongRunningScriptDetected(WebView sender,
115         ↪ WebViewLongRunningScriptDetectedEventArgs args) {
116         // DISABLED
117     }
118     private void MainWebView_PermissionRequested(WebView sender,
119         ↪ WebViewPermissionRequestedEventArgs args) {
120         // DISABLED
121     }
122 }

```

```

1
1  _____ WiFiWebAutoLogin/JavaScript/DeployListeners.js _____
2  var inputs = document.getElementsByTagName("input");
3  var ahrefs = document.getElementsByTagName("a");
4  var buttons = document.getElementsByTagName("button");
5
6  function addClickListener(el, tagName, idx) {
7      el.addEventListener("click", function () {
8          ScriptNotifyHandler.passAction("document.getElementsByTagName(\""
9          ↪ + tagName + "\"")[idx].click();");
10     });
11 }
12 function addChangeListener(el, tagName, idx) {
13     el.addEventListener("change", function () {
14         ScriptNotifyHandler.passAction("document.getElementsByTagName(\""
15         ↪ + tagName + "\"")[idx].value = " + JSON.stringify(docum_
16         ↪ ent.getElementsByTagName(tagName)[idx].value) +
17         ↪ ";");
18     });
19 }
20
21 for (var i = 0; i < inputs.length; i++) {
22     addClickListener(inputs[i], "input", i);
23     addChangeListener(inputs[i], "input", i);
24 }
25
26 for (var i = 0; i < ahrefs.length; i++) {
27     addClickListener(ahrefs[i], "a", i);
28 }
29
30 for (var i = 0; i < buttons.length; i++) {
31     addClickListener(buttons[i], "button", i);
32 }

```

2

3 A.2 Namespace: WiFiWebAutoLogin.Classes

```

1  _____ WiFiWebAutoLogin.Classes/ActionSequence.cs _____
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;

```

```

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

```

```

1  _____ WiFiWebAutoLogin.Classes/CaptivePortalDetector.cs _____
2
3  using System;
4  using System.Collections.Generic;
5  using System.Diagnostics;
6  using System.IO;
7  using System.Linq;
8  using System.Net;
9  using System.Net.Http;
10 using System.Net.NetworkInformation;
11 using System.Runtime.Serialization.Json;
12 using System.Text;
13 using System.Text.RegularExpressions;
14 using System.Threading;
15 using System.Threading.Tasks;
16 using Windows.Foundation;
17 using Windows.Foundation.Metadata;
18 using Windows.Networking.Connectivity;
19 using Windows.Storage;
20 using Windows.System.Threading;
21 using Windows.UI.ViewManagement;
22 using Windows.UI.Xaml;
23 using Windows.UI.Xaml.Controls;
24
25 namespace WiFiWebAutoLogin.Classes {
26     public class CaptivePortalDetector {

```

```
25     private static CaptivePortalDetector instance = null;
26     private Storage storage;
27     private string ssid;
28     private Queue<Uri> uriQueue;
29
30     private WebView webView;
31     private TextBlock textBlock;
32     private ComboBox comboBox;
33
34     private string currentFingerprint;
35     private ActionSequence currentActionSequence;
36     private ThreadPoolTimer timer;
37
38     private CaptivePortalDetector() {
39         this.webView = null;
40         this.storage = new Storage("credentials.dat");
41         this.uriQueue = new Queue<Uri>();
42     }
43
44     public bool isSetup() {
45         return this.webView != null;
46     }
47
48     public void setup(WebView webView, TextBlock textBlock, ComboBox
49 ↪ comboBox) {
50         this.webView = webView;
51         this.textBlock = textBlock;
52         this.comboBox = comboBox;
53
54         this.refreshList();
55
56         this.updateSSID();
57         this.updateWebView();
58     }
59
60     public void refreshList() {
61         comboBox.ItemsSource = this.storage.getLoginInfo().getList();
62     }
63
64     public WebView getWebView() {
65         return this.webView;
66     }
67
68     public void updateWebView() {
69         if (this.ssid != null) {
70             // CONNECTED
71             this.webView.Navigate(new Uri(Conf.uri));
72         }
73         else {
74             // DISCONNECTED
75             this.displayMessage("Check your network connection.");
76         }
77     }
78 }
```

```

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
93             ↪ etActionSequence(this.currentFingerprint);
94             bool hasActionSequence = true;
95             if (this.currentActionSequence == null) {
96                 hasActionSequence = false;
97                 this.currentActionSequence = new ActionSequence();
98                 this.storage.getLoginInfo().addActionSequence(this.cur
99                 ↪ rentFingerprint,
100                 ↪ this.currentActionSequence);
101                 this.storage.saveData();
102                 this.refreshList();
103             }
104
105             if (!body.Trim().Equals("connected")) {
106                 // Not Connected
107
108                 if (hasActionSequence) {
109                     this.displayMessage("Executing recorded
110                     ↪ actions...\r\n\r\n" + "(" +
111                     ↪ this.currentFingerprint.Split(new string[] {
112                     ↪ Conf.separator },
113                     ↪ StringSplitOptions.RemoveEmptyEntries)[1] +
114                     ↪ ")");
115                 }
116
117                 IEnumerable<string> actions =
118                 ↪ this.currentActionSequence.getEnumerable();
119                 string compiledActions = "";
120                 foreach (string action in actions) {
121                     compiledActions += action;
122                 }
123                 await this.webView.InvokeScriptAsync("eval", new
124                 ↪ string[] { compiledActions });
125
126                 // Deploy Listeners
127                 this.deployListeners();
128                 if (this.uriQueue.Count > 0) {
129                     this.startTimer();

```

```

120         }
121
122         if (!hasActionSequence) {
123             this.displayWebView();
124         }
125         else {
126             this.startRetryTimer(this.currentFingerprint);
127         }
128     }
129     else {
130         // Connected
131         this.displayMessage("Connected.");
132         this.uriQueue.Clear();
133     }
134 }
135
136
137 public void navigationStarting() {
138     if (this.timer != null) {
139         this.timer.Cancel();
140         this.timer = null;
141     }
142 }
143
144 private void startRetryTimer(string oldFingerprint) {
145     ThreadPoolTimer.CreateTimer(async (source) => {
146         await Windows.ApplicationModel.Core.CoreApplication.MainView.
147             ↪ ew.CoreWindow.Dispatcher.RunAsync(Windows.UI.Core.Core
148             ↪ DispatcherPriority.Normal, () =>
149             ↪ {
150                 if (this.currentFingerprint.Equals(oldFingerprint)) {
151                     this.displayWebView();
152                 }
153             });
154     }, TimeSpan.FromSeconds(5));
155 }
156
157 private async void timerCallback() {
158     await Windows.ApplicationModel.Core.CoreApplication.MainView.C
159     ↪ oreWindow.Dispatcher.RunAsync(Windows.UI.Core.CoreDispatch
160     ↪ erPriority.Normal, () =>
161     ↪ {
162         if (this.timer != null) {
163             this.timer = null;
164         }
165         this.dequeueUri();
166     });
167 }
168
169 private async void deployListeners() {
170     StorageFolder InstallationFolder =
171     ↪ Windows.ApplicationModel.Package.Current.InstalledLocation;
172     StorageFile file = await InstallationFolder.GetFilesAsync(@"Jav
173     ↪ 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 {
275             ↪ Width = 600, Height = 150 });
276         this.textBlock.Text = message;
277         this.webView.Margin = new Thickness(0, int.MaxValue, 0,
278             ↪ int.MinValue);
279     }
280
281     private void displayWebView() {
282         ApplicationView.GetForCurrentView().TryResizeView(new Size {
283             ↪ Width = 800, Height = 500 });
284         this.textBlock.Text = "";
285         this.webView.Margin = new Thickness(0, 0, 0, 0);
286     }
287
288     public void timeout() {
289         this.displayMessage("Operation timeout.\r\nCheck your network
290             ↪ connection.");
291     }
292
293     public void removeLoginInformation(string ssid) {
294         if (ssid != null) {
295             this.storage.getLoginInfo().removeBySSID(ssid);
296             this.storage.saveData();
297             this.refreshList();
298         }
299     }
300 }

```

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 _____
2
3  using System;
4  using System.Collections.Generic;
5  using System.Collections.ObjectModel;
6  using System.Linq;
7  using System.Runtime.Serialization;
8  using System.Text;
9  using System.Threading.Tasks;
10 using Windows.ApplicationModel.Contacts;
11
12 namespace WiFiWebAutoLogin.Classes {
13     [DataContract]
14     class LoginInformation {
15         [DataMember]
16         private Dictionary<string, ActionSequence> actionSequences;
17
18         public LoginInformation() {
19             this.actionSequences = new Dictionary<string,
20                 ↪ ActionSequence>();
21             ActionSequence actionSequence = new ActionSequence();
22         }
23
24         public ActionSequence getActionSequence(string fingerprint) {
25             try {
26                 return this.actionSequences[fingerprint];
27             } catch (Exception e) {
28                 return null;
29             }
30         }
31
32         public void addActionSequence(string fingerprint, ActionSequence
33             ↪ actionSequence) {
34             this.actionSequences.Add(fingerprint, actionSequence);
35         }
36
37         public List<string> getList() {
38             Dictionary<string, ActionSequence>.KeyCollection.Enumerator
39                 ↪ loginInfoEnumerator = actionSequences.Keys.GetEnumerator();
40             List<string> list = new List<string>();
41             while (loginInfoEnumerator.MoveNext()) {
42                 string ssid = loginInfoEnumerator.Current.Split(new
43                     ↪ string[] { Conf.separator },
44                     ↪ StringSplitOptions.RemoveEmptyEntries)[0];
45                 if (!list.Contains(ssid)) {
46                     list.Add(ssid);
47                 }
48             }
49         }
50     }
51 }

```

```

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

```

1 _____ WiFiWebAutoLogin.Classes/Storage.cs _____

```

1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Runtime.Serialization.Json;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Xml.Serialization;
9  using Windows.Security.Credentials;
10 using Windows.Security.Cryptography;
11 using Windows.Security.Cryptography.Core;
12 using Windows.Storage;
13 using Windows.Storage.Streams;
14
15 namespace WiFiWebAutoLogin.Classes {
16     class Storage {
17         private string fileName;
18         private string password;
19         private LoginInformation loginInfo;
20
21         public Storage(string fileName) {
22             this.fileName = new String(fileName.ToCharArray());
23             PasswordVault vault = new PasswordVault();
24
25             try {

```

```

26         this.password = vault.Retrieve(Conf.resource,
27             ↪ Conf.username).Password;
28     }
29     catch (Exception e) {
30         vault.Add(new PasswordCredential(Conf.resource,
31             ↪ Conf.username, CryptographicBuffer.EncodeToBase64Strin
32             ↪ g(CryptographicBuffer.GenerateRandom(64))));
33         this.password = vault.Retrieve(Conf.resource,
34             ↪ Conf.username).Password;
35     }
36 }
37
38 public async Task setup() {
39     StorageFolder folder = ApplicationData.Current.LocalFolder;
40     StorageFile file;
41     try {
42         file = await folder.GetFilesAsync(this.fileName);
43     } catch (Exception e) {
44         file = await folder.CreateFileAsync(this.fileName);
45     }
46
47     IBuffer encryptedJson = await FileIO.ReadBufferAsync(file);
48     SymmetricKeyAlgorithmProvider algorithmProvider =
49         ↪ SymmetricKeyAlgorithmProvider.OpenAlgorithm(SymmetricAlgor
50         ↪ ithmNames.AesCbcPkcs7);
51     IBuffer bufferedPassword =
52         ↪ CryptographicBuffer.ConvertStringToBinary(password,
53         ↪ BinaryStringEncoding.Utf8);
54     IBuffer decryptedJson = CryptographicEngine.Decrypt(algorithmP
55         ↪ rovider.CreateSymmetricKey(bufferedPassword),
56         ↪ encryptedJson, bufferedPassword);
57     DataReader dataReader = Windows.Storage.Streams.DataReader.Fro
58         ↪ mBuffer(decryptedJson);
59     string json = dataReader.ReadString(decryptedJson.Length);
60
61     if (json.Trim().Equals("")) {
62         loginInfo = new LoginInformation();
63         this.saveData();
64     }
65     else {
66         DataContractJsonSerializer serializer = new
67             ↪ DataContractJsonSerializer(typeof(LoginInformation));
68         loginInfo = (LoginInformation)serializer.ReadObject(new
69             ↪ MemoryStream(Encoding.Unicode.GetBytes(json)));
70     }
71 }
72
73 public LoginInformation getLoginInfo() {
74     return this.loginInfo;
75 }
76
77 public async void saveData() {
78     DataContractJsonSerializer serializer = new
79         ↪ DataContractJsonSerializer(typeof(LoginInformation));

```

```

66     MemoryStream stream = new MemoryStream();
67     serializer.WriteObject(stream, loginInfo);
68     stream.Position = 0;
69     StreamReader sr = new StreamReader(stream);
70     string data = sr.ReadToEnd();
71
72     SymmetricKeyAlgorithmProvider algorithmProvider =
73     ↪ SymmetricKeyAlgorithmProvider.OpenAlgorithm(SymmetricAlgori
74     ↪ thmNames.AesCbcPkcs7);
75     IBuffer keyMaterial =
76     ↪ CryptographicBuffer.ConvertStringToBinary(password,
77     ↪ BinaryStringEncoding.Utf8);
78     IBuffer bufferedData = CryptographicBuffer.CreateFromByteArray
79     ↪ (Encoding.UTF8.GetBytes(data));
80     CryptographicKey key =
81     ↪ algorithmProvider.CreateSymmetricKey(keyMaterial);
82     IBuffer encryptedData = CryptographicEngine.Encrypt(key,
83     ↪ bufferedData, keyMaterial);
84
85     StorageFolder folder = ApplicationData.Current.LocalFolder;
86     StorageFile file;
87     try {
88         file = await folder.GetFilesAsync(this.fileName);
89     }
90     catch (Exception e) {
91         file = await folder.CreateFileAsync(this.fileName);
92     }
93
94     await FileIO.WriteBufferAsync(file, encryptedData);
95 }
96 }
97 }

```

1

2 A.3 Namespace: WiFiWebAutoLogin.RuntimeComponents

```

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

```

```

17
18     public void Run(IBackgroundTaskInstance taskInstance) {
19         if (this.connectionChanged() && lastSSID!=null &&
20             ↪ this.hasNoInternetAccess()) {
21
22             string xmlText = "<?xml version=\"1.0\" encoding=\"utf-8\"
23             ↪ ?>" +
24                 "<toast launch=\"app-defined-string\">" +
25                 "<visual>" +
26                 "<binding template=\"ToastGeneric\">" +
27                 "<text>Network Detected</text>" +
28                 "<text>Would you like to run
29                 ↪ WiFiWebAutoLogin?</text>" +
30                 "</binding>" +
31                 "</visual>" +
32                 "<actions>" +
33                 "<action content=\"Yes\" arguments=\"Yes\" />"
34                 ↪ +
35                 "<action content=\"No\" arguments=\"No\"
36                 ↪ activationType=\"background\" />" +
37                 "</actions>" +
38                 "<audio src=\"ms-winsoundevent:Notification.Remind_
39                 ↪ er\"/>"
40                 ↪ +
41                 "</toast>";
42
43             XmlDocument xmlContent = new XmlDocument();
44             xmlContent.LoadXml(xmlText);
45
46             ToastNotification notification = new
47             ↪ ToastNotification(xmlContent);
48             notification.Tag = "WWAL_TOAST";
49             notification.Dismissed += (ToastNotification n,
50             ↪ ToastDismissedEventArgs args) => {
51                 ToastNotificationManager.History.Remove("WWAL_TOAST");
52             };
53             ToastNotificationManager.CreateToastNotifier().Show(notifi_
54             ↪ cation);
55         }
56     }
57
58     private bool hasNoInternetAccess() {
59         ConnectionProfile connectionProfile =
60         ↪ NetworkInformation.GetInternetConnectionProfile();
61
62         if (connectionProfile != null) {
63             if (connectionProfile.GetNetworkConnectivityLevel().ToStri_
64             ↪ ng().Trim().Equals("InternetAccess"))
65             ↪ {
66                 return false;
67             }
68         }
69     }

```

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

```
1      WiFiWebAutoLogin.RuntimeComponents/ScriptNotifyHandler.cs
2
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using Windows.Foundation.Metadata;
9  using WiFiWebAutoLogin.Classes;
10 using System.Diagnostics;
11
12 namespace WiFiWebAutoLogin.RuntimeComponents
13 {
14     [AllowForWeb]
15     public sealed class ScriptNotifyHandler
16     {
17         public async void passAction(string args) {
18             CaptivePortalDetector cpd = await
19                 ↪ CaptivePortalDetector.GetInstance();
20             cpd.passAction(args);
21         }
22
23         public async void windowOpen(string args) {
24             CaptivePortalDetector cpd = await
25                 ↪ CaptivePortalDetector.GetInstance();
26             cpd.queueUri(new Uri(args));
27             Debug.WriteLine(args);
28         }
29
30         public void testDebug(string args) {
31             Debug.WriteLine(args);
32         }
33     }
34 }
```