



Team W.A.T.E.R

Max Wason

John Sadie

Peter Haschke

Jason Le

Mentor: Dr. Mohamed M. Elwakil
Sponsor: Choice - James Harms



Website Automated Testing for Enterprise Reliability

Problem Statement

Websites

Omnipresent nowadays due to their usefulness.

Hotel Websites

Offer a **valuable service**.

Example: Priceline - \$9.22 billion in revenue in 2015.

They must be **reliable** to turn these kinds of profits.

Choice Hotels' Websites

Another **large** company, with huge profit margins.

They need to easily **manage** and **run** many websites **smoothly**.

Their Problem

The **testing** process isn't all automated.

Getting information **from a website** **into a testing suite** is time consuming and therefore expensive.

Current Testing Workflow

Navigate to a web page

Inspect elements

Transpose information

Automatically **run** test suite

Element Acquisition



```
@Test  
public void testFirstScript() throws Exception {  
    driver.get(baseUrl + "/ServiceLogin?service=mail&passive=true");  
    driver.findElement(By.id("Email")).clear();  
    driver.findElement(By.id("Email")).sendKeys("XXXXXX");  
    driver.findElement(By.id("Passwd")).clear();  
    driver.findElement(By.id("Passwd")).sendKeys("XXXXXX");  
    driver.findElement(By.id("signIn")).click();
```

Inspection

- ← Manually inspect elements on a webpage to obtain data
 - ◎ Time intensive (and therefore expensive), tedious, and repetitive.
 - Exactly the strengths of computers

Transposition

- ← Manually plug in that information to the testing suite
 - ◎ Same issues as above, plus human error.

There's got to be a better way!



Introducing
Chrome Testing Plugin

Solution Overview

Our proposed solution is to automatically extract web page elements and package them for various testing suites

Example Input

The screenshot shows the homepage of Choice Hotels International. At the top, there's a navigation bar with links for "MANAGE RESERVATIONS", "CUSTOMER SUPPORT", "INCREASE CONTRAST", and "SELECT A LANGUAGE". Below the navigation is a main menu with "HOME", "VACATION RENTALS", "GROUPS", "EXPLORE", and a "+" sign. To the right of the menu is the "CHOICE privileges REWARDS" logo and a "SIGN IN / JOIN" button. The main content area features a large image of a modern hotel building at sunset. Overlaid on the image is the slogan "How simple is it to get the lowest price? BADDABOOK. BADDABOOM.™" in large, bold letters. Below the slogan is the text "It's that simple." and a "LEARN MORE" button. At the bottom, there's a search bar with placeholder text "CITY, AIRPORT CODE, ATTRACTION" and date fields for "CHECK-IN" (Mon, 04/10/2017) and "CHECK-OUT" (Tue, 04/11/2017).

Choice Hotels – Official S X

Choice Hotels International, Inc. [US] | https://www.choicehotels.com

MANAGE RESERVATIONS | CUSTOMER SUPPORT | INCREASE CONTRAST | SELECT A LANGUAGE

CHOICE HOTELS

HOME VACATION RENTALS GROUPS EXPLORE +

CHOICE privileges[®] REWARDS

SIGN IN / JOIN

How simple is it to get the lowest price?

BADDABOOK.
BADDABOOM.™

It's that simple.

LEARN MORE

CITY, AIRPORT CODE, ATTRACTION

CHECK-IN

CHECK-OUT

What is Your Destination?

Mon, 04/10/2017

Tue, 04/11/2017

Example Output

```
Webpage elements retrieved from: https://www.choicehotels.com/ at Mon Apr 10 2017 20:07:10 GMT-0700 (Mountain Standard Time)
34 total elements retrieved, separated into 4 categories.
{
  "descriptiveName": "username",
  "fullHTML": "<input type=\"text\" aria-describedby=\"cpSignInUsernameError\" autocapitalize=\"none\" class=\"form-control ng-pristine ng-untouched ng-empty ng-invalid ng-invalid-required ng-valid-maxlength\"",
  "type": "Input",
  "class": "form-control ng-pristine ng-untouched ng-empty ng-invalid ng-invalid-required ng-valid-maxlength",
  "id": "cpSignInUsername",
  "name": "username",
  "xpath": "//*[@id=\"cpSignInUsername\"]"
}
---
{
  "descriptiveName": "password",
  "fullHTML": "<input type=\"password\" aria-describedby=\"cpSignInPasswordError\" autocomplete=\"off\" class=\"form-control ng-pristine ng-untouched ng-empty ng-invalid ng-invalid-required ng-valid-maxlength\"",
  "type": "Input",
  "class": "form-control ng-pristine ng-untouched ng-empty ng-invalid ng-invalid-required ng-valid-maxlength",
  "id": "cpSignInPassword",
  "name": "password",
  "xpath": "//*[@id=\"cpSignInPassword\"]"
}
---

/*Webpage elements retrieved from: https://www.choicehotels.com/ at Fri Apr 07 2017 09:51:55 GMT-0700 (Mountain Standard Time) */
import org.openqa.selenium.*;

public class SampleTestClass {
    public WebDriver webDriver;

    /*<input type="text" aria-describedby="cpSignInUsernameError" autocapitalize="none" class="form-control ng-pristine ng-untouched ng-empty ng-invalid ng-invalid-required ng-valid-maxlength"*/
    public void InputcpSignInUsername() {
        WebElement webElement = webDriver.findElement(By.name("username"));
        webElement.getAttribute("value");
    }

    public void setInputcpSignInUsername(String value) {
        WebElement webElement = webDriver.findElement(By.name("username"));
        webElement.sendKeys(value);
    }

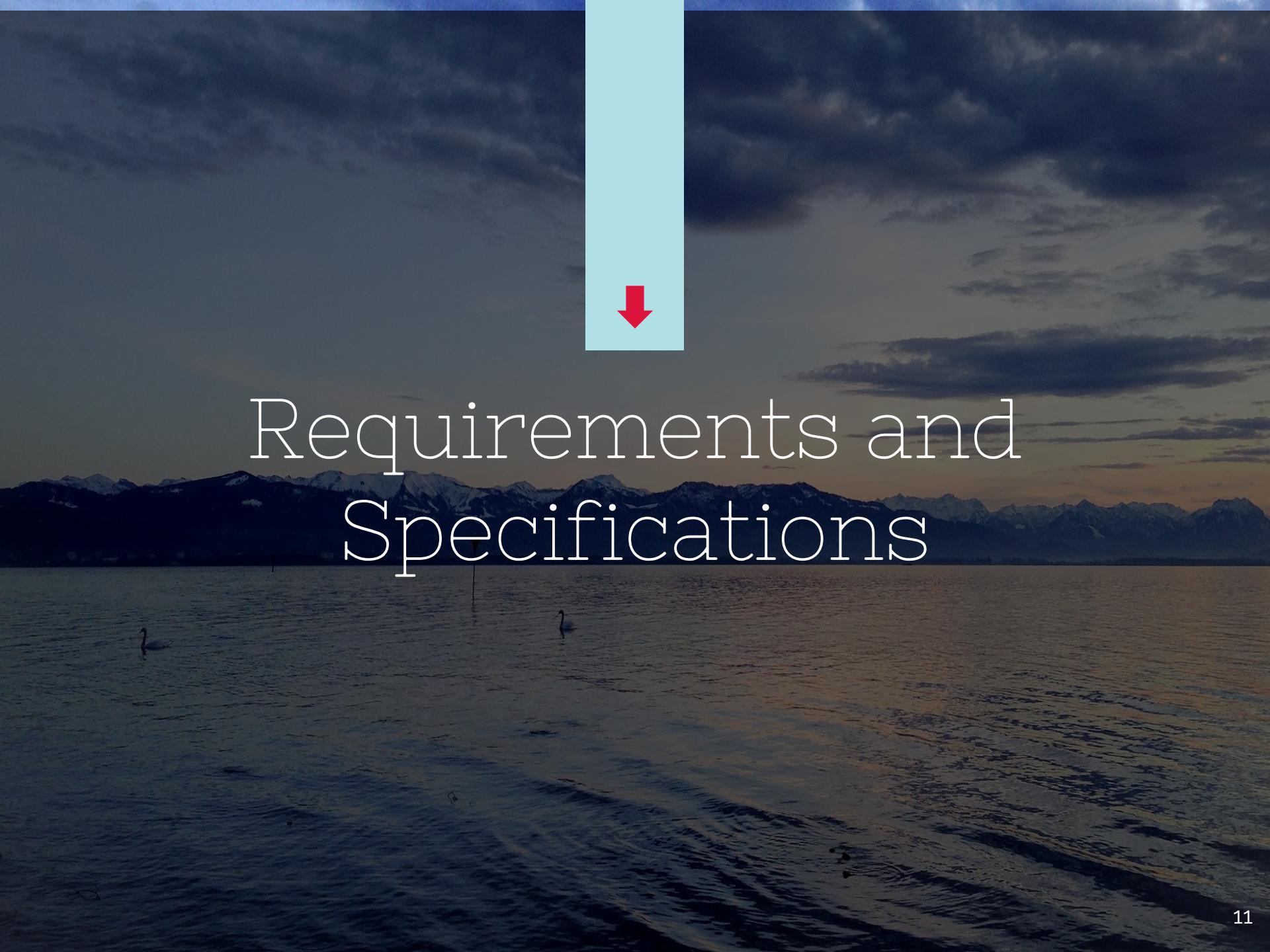
    /*<input type="password" aria-describedby="cpSignInPasswordError" autocomplete="off" class="form-control ng-pristine ng-untouched ng-empty ng-invalid ng-invalid-required ng-valid-maxlength"*/
    public void InputcpSignInPassword() {
        WebElement webElement = webDriver.findElement(By.name("password"));
        webElement.getAttribute("value");
    }
}
```

Solution Details

◎ Key Features:

- Pulls the UI Elements from a specific web-page
- Filters through for specific UI Elements
- Outputs into multiple file types
 - Some output file types can be used in testing suites

Overall, by using this extension the time it takes to test a Web-page would **significantly** drop.



Requirements and Specifications

Requirements and Acquisition revisited

Key Requirements:

- ◎ The tool needs to be able to be used on any webpage.
- ◎ The tool needs to be able to get the attributes of ALL UI elements on a page.
- ◎ The tool needs to output formatted files to be used in various testing suites.
- ◎ The tool should be fast, effective, and easy to use.



Architecture Overview

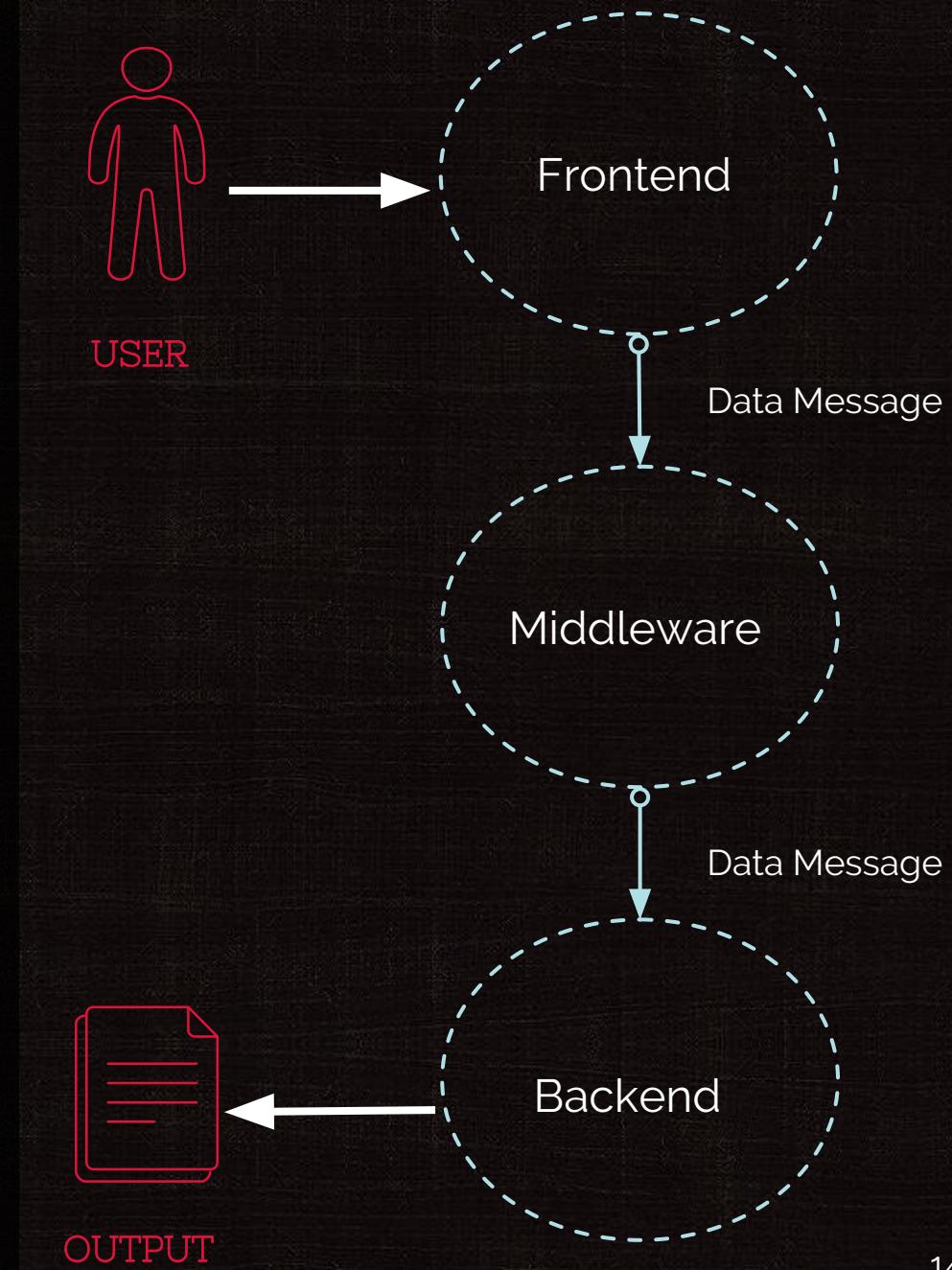
Structured Solution

Why A Chrome Extension?

- ◎ Browser Extension vs. Desktop application.
- ◎ Chrome Extension vs. Firefox Plugin

The Chrome Extension Structure.

- ◎ Javascript
- ◎ **Three** modules of execution:
 - **Frontend** (Interactive User Interface or UI)
 - **Middleware** (The "heavy lifting")
 - **Backend** (Make and deliver outputs)



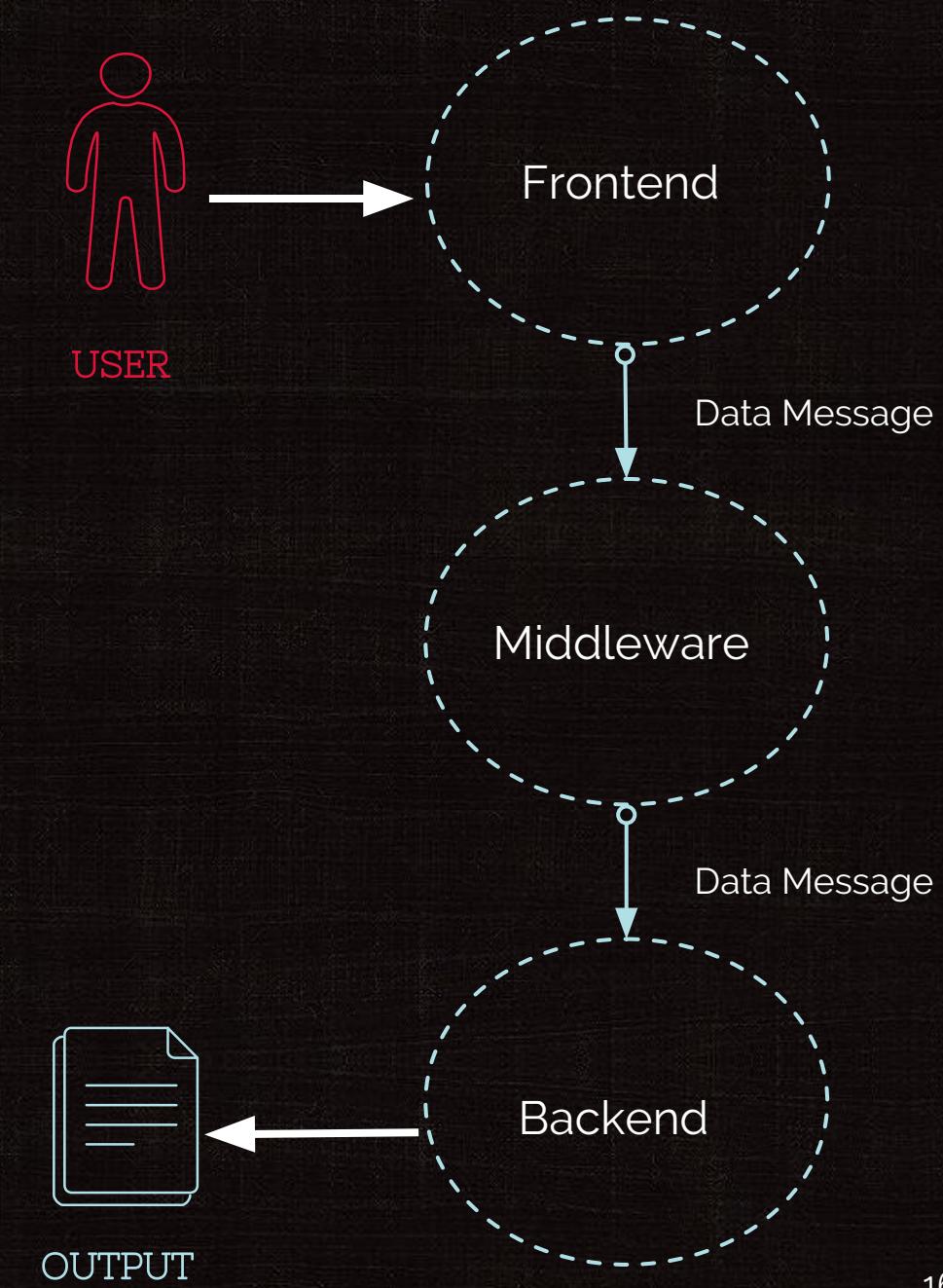
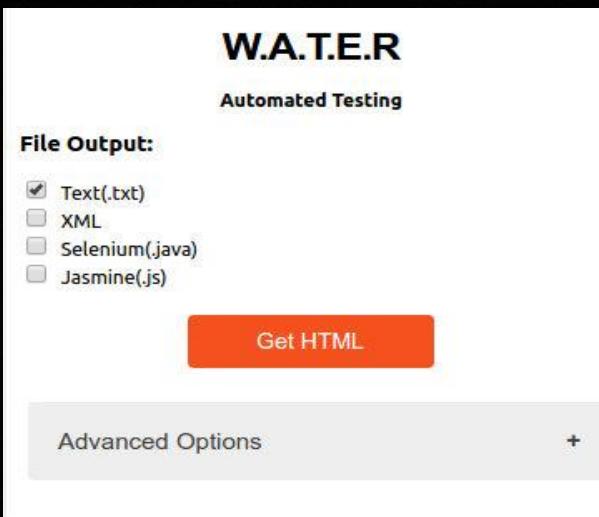


Implementation Overview

Frontend

Functionality

- ◎ Provide GUI functionality.
(Error Checking)
- ◎ Retrieve and package data
from the GUI.
- ◎ Send GUI data to the
Middleware.

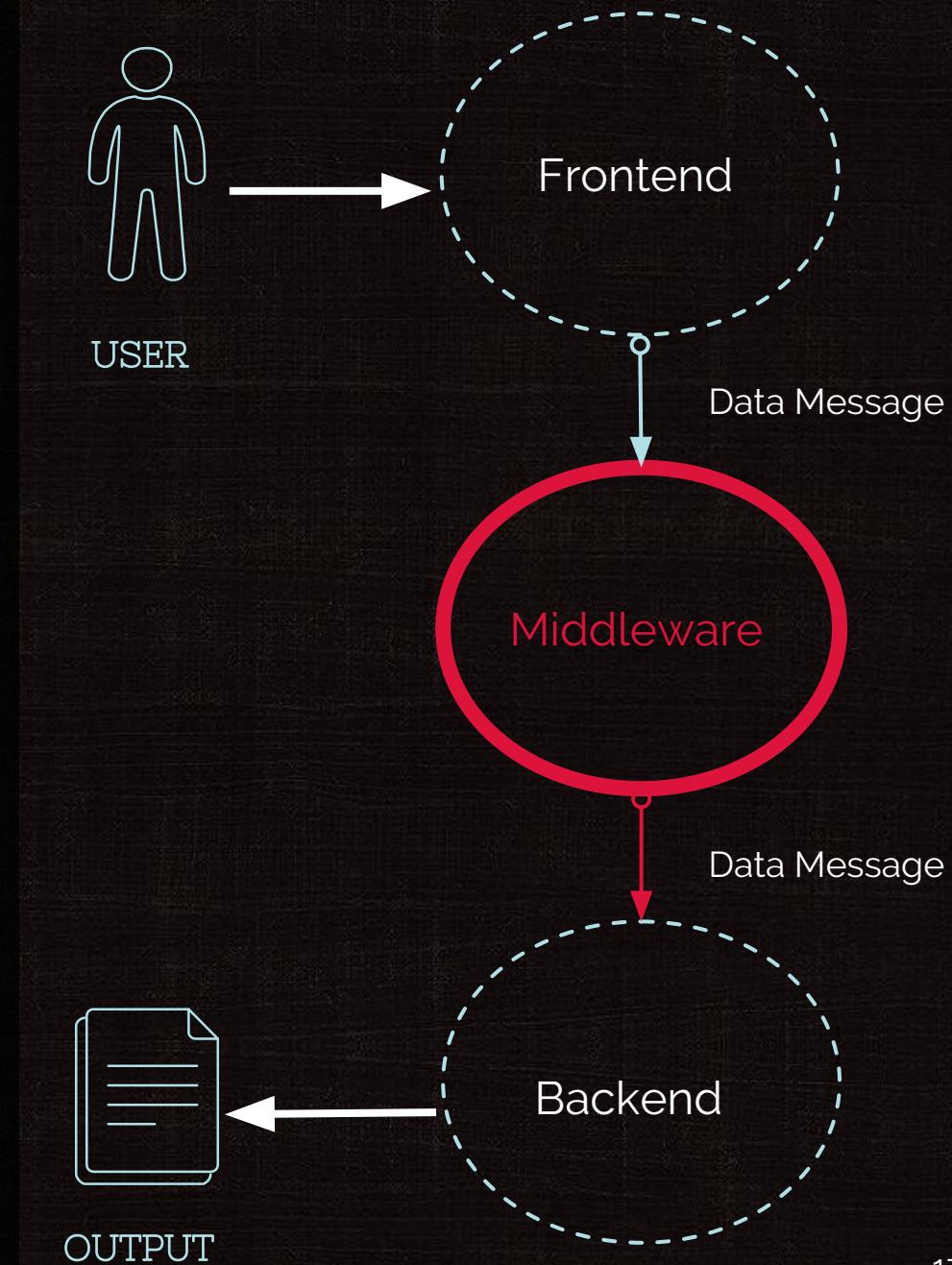


Plugin GUI Sample

Middleware

Functionality

- ◎ Retrieve user GUI checkbox data from frontend module.
- ◎ Pull all elements from the current page.
- ◎ Filter out all UI elements. (Buttons, Links, Inputs, etc.)
- ◎ Get attributes from each UI element (id, name, XPath, formulate descriptive name).
- ◎ Package element data and send to the backend.



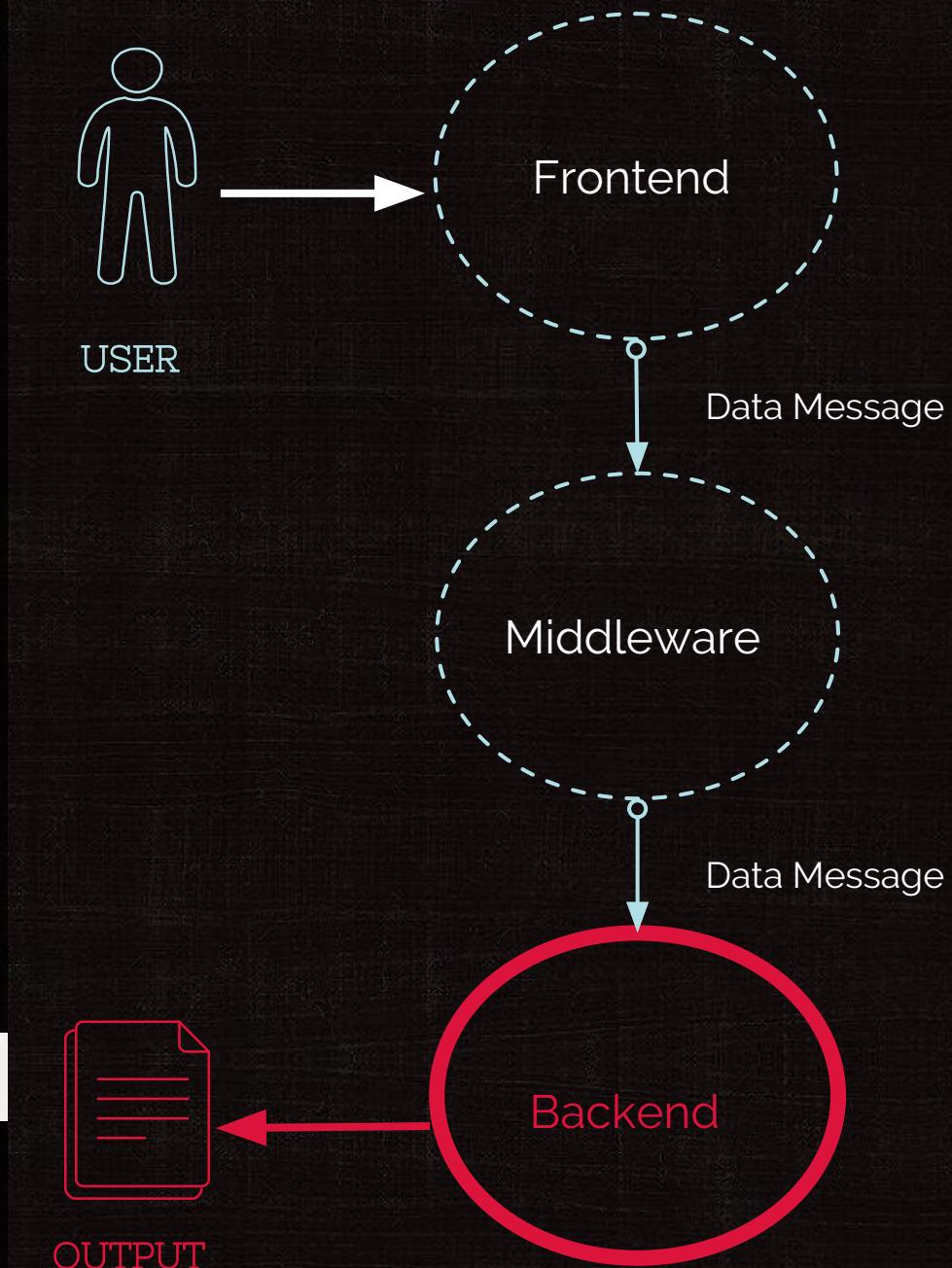
Backend

Functionality

- ◎ Retrieve element data and GUI checkbox data (passed through the middleware).
- ◎ Depending on output file selection, from UI data, create the files.
- ◎ Download the files using Chrome API class.

```
water_result....java  water_results....xml
```

File Output Downloads





Prototype Review (Demo)



Challenges and Resolutions

Challenges and Resolutions

Challenge 1:

Completed

Adapting from
executable jar, into
browser plugin

Resolution 1:

We are fully integrated into the Chrome plugin.
We refined it into what we have just shown
you.

Challenge 2:

In Progress

Accommodating
Jasmine -

Getting JavaScript
object output

Resolution 2:

We are currently communicating with one of
the engineers at Choice to acquire more
knowledge on how this output file should look.

Challenges and Resolutions

Challenge 3:

Completed

Getting XPath of each element

Resolution 3:

We have successfully acquired the XPath of every element, using a component from an open source project.

Challenge 4:

In Progress

Supporting dynamic frameworks: JavaScript, Angular

Resolution 4:

Parsing unique elements within those frameworks. Such as: ng-click & onclick. We are successfully creating a separate method that grabs the particular data.



Schedule

Rally

ca technologies | Team Water | Plan Track Quality Portfolio Reports | Search | User Icon

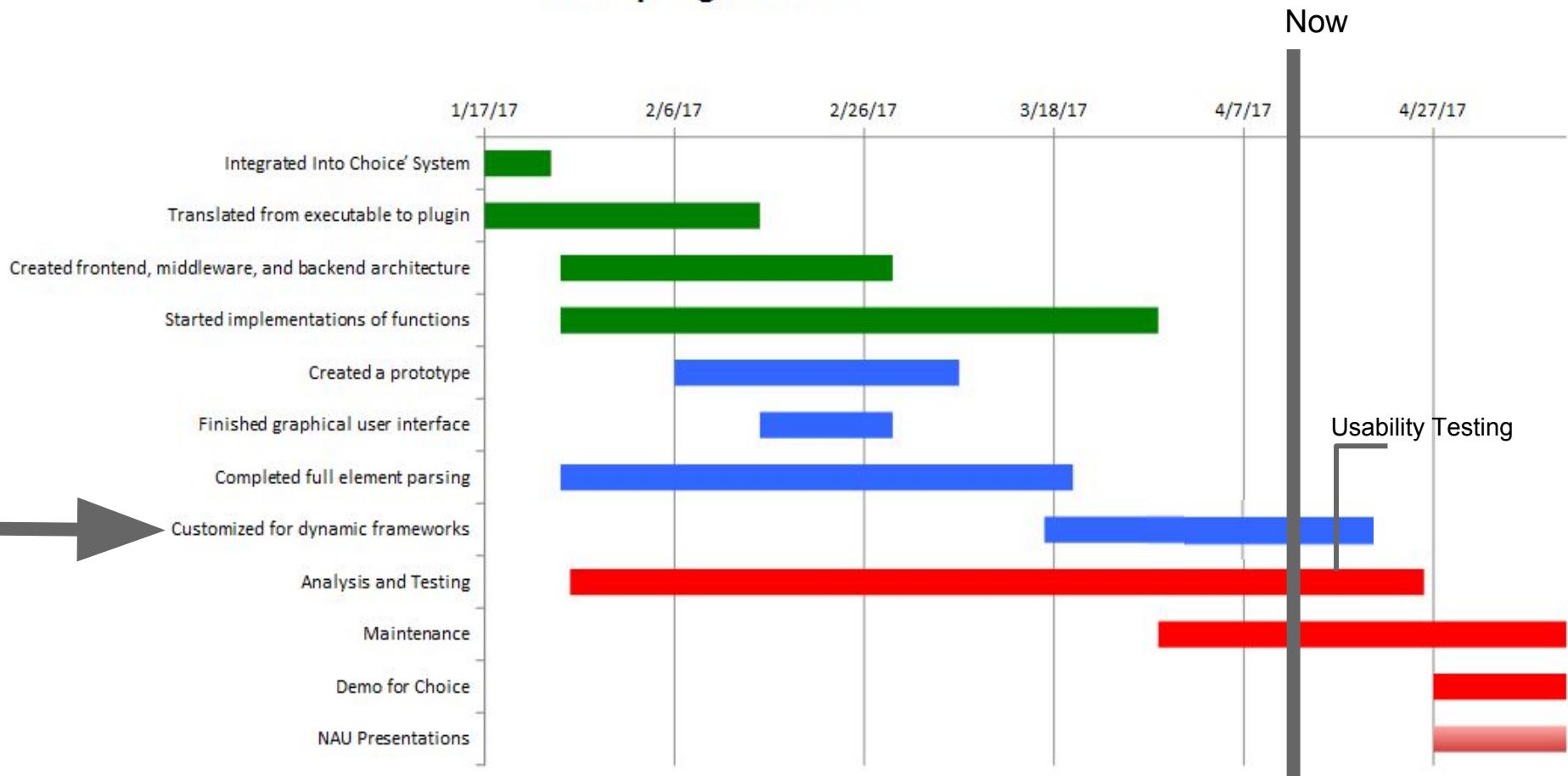
Team Status

Actions ▾

Iteration Two (02-23-2017 to 03-09-2017) ▾		Work Product	Release	State	Capacity	Estimate	To Do	Actuals	Owner					
All	Rank ▲ ID	Name	All	All	0.0	H	40.0	H	23.0	H	1.0	H	Type to Filter...	Filter
<input type="checkbox"/>	#													
<input type="checkbox"/>	Jason L (2 Tasks)													
	TA331356	Add further elements	US70024: getElementsToParse()	D P C	3.0	0.0							Jason L	  
	TA332115	Work on sorting (via UI options)	US70021: sortElementArray()	D P C	3.0	0.0							Jason L	  
<input type="checkbox"/>	John S (3 Tasks)													
	TA331348	Create GUI	US70258: create GUI	D P C	3.0	0.0							John S	  
	TA332080	Add Advanced GUI Elements/Filters	US70258: create GUI	D P C	2.0	2.0							John S	  
	TA331351	Work on xPath implementation	US70032: getElementXPath()	D P C	4.0	4.0							John S	  
<input type="checkbox"/>	Max Wason (2 Tasks)													
	TA331358	Work on incremental implementation	US70014: getElements()	D P C	4.0	0.0							Max Wason	  
	TA332111	Filter element acquisition	US70014: getElements()	D P C	5.0	5.0							Max Wason	  
<input type="checkbox"/>	Peter H (6 Tasks)													
	TA332113	Initial Research/Implementation	US70036: writeJObjectFile()	D P C	3.0	3.0							Peter H	  
	TA332114	Add basic objects/syntax	US70033: writeSeleniumFile()	D P C	3.0	3.0							Peter H	  
	TA332116	Initial implementation/research	US70030: getJSElements()	D P C	2.0	2.0							Peter H	  
	TA332117	Initial Research	US70031: getAngularElements()	D P C	2.0	2.0							Peter H	  
	TA331353	Initial XML Implementation	US70022: writeXMLFile()	D P C	4.0	0.0	1.0						Peter H	  
	TA332112	Flesh out with elements	US70022: writeXMLFile()	D P C	2.0	2.0							Peter H	  

2017 Spring Schedule

2017 Spring Schedule





Usability Testing

Testing Procedure

We will proceed with these following elements as our test procedure:

Scope

This begins with the W.A.T.E.R team explaining a small background of our product to the participants

Purpose:

We will identify concerns, goals, and questions for our test.

There will be an underlying theme of questions we will take into account during the test.

Schedule & Location

The test will be done at one of the homes of a W.A.T.E.R member.

Metrics:

Focused mainly on understanding, overall ease of use and satisfaction.

This will also be monitored before, during, and after the test.

Analysis

Quantitative Data

Record Data Such As:

- Success rates
- Task time
- Error Rates
- Satisfaction ratings

Qualitative Data

Record Data Such As:

- Observations about the navigations participants took
- Problems experienced
- Comments/recommendations
- Answers to questions



Conclusion



*Our project is an ambitious one.
The solution must be clean and elegant.*



Our solution, once implemented, will save our client tremendous resources.

From all of us at Team W.A.T.E.R

Thank you!

ANY
QUESTIONS?

