Misconduct Detection Project(MDP) Tool
Testing and Debugging Report No.2

Yucheng Xie
September 24, 2018
Supervisor: Dr. Kyriakos Kalorkoti
Master of Science
School of Informatics
University of Edinburgh
Software Version: 1.0

Contents

1 Summary

This “testing and debugging report” is used as a record for later developers referring. Each bug in this report has been illustrated by following three aspects:

1. The details of the bug. This part is mainly giving the details of the bug, including how to reproduce the bug.
2. The analysis of the bug. This part is mainly focusing on analyzing the bug and providing fixing solutions to the bug.
3. The test cases of the bug. This part is providing the test cases for verifying the bug is resolved.

Especially, in the above information, later developers should always rerun the test cases of a bug if any code related to the bug is changed. Rerunning test cases could guarantee that later changes on the code are not making these bugs recur.

Involved bugs in this report:

1. UI-001: Programming Language Selection Panel
2. UI-002: Auto Programming Language Selection
3. DET-003: Results Retrieve Error
4. UI-003: Selection Page Style Loss Error

2 UI-001: Programming Language Selection Panel

2.1 Bug Report Part

General Information

Title	Programming Language Selection Panel
ID	(user interface) UI-001
Priority	Critical

Description

Allow users to decide which detection package and programming language to use. Due to time limitation, this function have not been finished in formal developing stage. Therefore, this function is planned and finished in extra debugging stage. The function in use is shown in Appendix B.

How to Reproduce

Not available.

Screenshots

See appendix.

2.2 Bug Analysis Part

Not available.

2.3 Test Cases Part

Test Case 1:

1. Normally upload file and folder, select segments.
2. Provide correct settings, i.e. manually give correct file extension.
3. Perform detection.

Expected Results for this Test Case:

The results are shown correctly.

Test Case 2:

1. Normally upload file and folder, select segments.
2. Provide incorrect settings, i.e. manually give incorrect file extension.
3. Perform detection.

Expected Results for this Test Case:

The “No Results” error page is shown. (details see Appendix B)

Test Case 3:

1. Register a new detection package in the system.

Expected Results for this Test Case:

The panel updates the package information and supported programming languages information automatically.

3 UI-002: Auto Programming Language Selection

3.1 Bug Report Part

General Information

Title	Auto Programming Language Selection
ID	UI-002
Priority	Critical

Description

Providing suggestions for users to decide which detection package and programming language to use. Due to time limitation, this function have not been finished in formal developing stage. Therefore, this function is planned and finished in extra debugging stage. The function in use is shown in Appendix C.

How to Reproduce

Not available.

Screenshots

See appendix.

3.2 Bug Analysis Part

Not available.

3.3 Test Cases Part

Test Case 1:

1. Normally upload file and folder, select segments.
2. Provide correct settings, i.e. manually give correct file extension.
3. Perform detection.

Expected Results for this Test Case:

The results are shown correctly. Nothing else happens.

Test Case 2:

1. Normally upload file and folder, select segments.
2. Provide incorrect settings, i.e. manually give incorrect file extension.
3. Perform detection.

Expected Results for this Test Case:

The auto selection function open a pop-up window to give the user suggestions about the optimal settings and give warnings to the user. However, if the user insist, auto selection function will follow the user's settings rather than using the optimal settings.

Test Case 3:

1. Normally upload file and folder, select segments.
2. Provide no settings, i.e. manually give no settings to tool.
3. Perform detection.

Expected Results for this Test Case:

The auto selection function prevent the user from performing detection and open a pop-up window to force the user provide settings.

4 DET-003: Results Retrieve Error

4.1 Bug Report Part

General Information

Title	Results Retrieve Error
ID	DET-003
Priority	Normal

Description

If the server is shut-down, the result keys will be lost and the user will not be able to retrieve the results. The user will come across a “key error” when trying to retrieve results.

How to Reproduce

1. Perform detection normally.
2. Jump to the “result page”, it should work correctly.
3. Reboot the server.
4. Jump to the “result page”, now it should indicates a “key error”.

4.2 Bug Analysis Part

In our tool, we used a “file name manager” to manage the actual file names and the copied renamed files. However, this “file name manager” store file relationships in the memory. Therefore, if the server is rebooted or shut down, the file relationships will be lost and thus causes the “key error”.

Therefore, we store the file relationships on the disk rather than memory to solve this problem.

4.3 Test Cases Part

Test Case 1:

1. Perform detection normally.
2. Jump to the “result page”, it should work correctly.
3. Reboot the server.
4. Jump to the “result page” again.

Expected Results for this Test Case:

The “result page” should be loaded correctly.

5 UI-003: Selection Page Style Loss Error

5.1 Bug Report Part

General Information

Title	Selection Page Style Loss Error
ID	UI-003
Priority	Normal

Description

When refreshing the selection page, the left side highlighted segments will loss their format and become plain text again.

How to Reproduce

1. Normally select segments on the “selection page” and save the selection.
2. Refresh the “selection page” and all format on the left side should be reset to plain text.

5.2 Bug Analysis Part

Here we used a redraw function to redraw the segments when the page is reload. However, the originally highlighted segments got the indexes from the user’s selection. In this situation, the user ask to reload the page, the segments cannot obtain the indexes since the user is not highlighting any segment.

To resolve this bug, we decide not to redraw each segment one by one. Instead, we save the HTML file and redraw the whole page. Although this also causes a problem that when we remove any segment, we need to carefully rewrite the HTML page to ensure it is restored to the correct state.

5.3 Test Cases Part

Test Case 1:

1. Normally select segments on the “selection page” and save the selection.
2. Refresh the “selection page”.

Expected Results for this Test Case:

The “selection page” left side block should be loaded correctly.

Test Case 2:

1. Remove some segments, add some other segments and save the selections.
2. Refresh the “selection page”.

Expected Results for this Test Case:

The “selection page” left side block should be loaded correctly. The ordering of the segments should be correct. The name of the segments should be correct.

Appendix

A Explanation of Bug Priorities

In this report, we divided all bugs into three different priority levels. They are:

1. Critical:

   Something could case the whole system to be stopped or case the system producing wrong results.

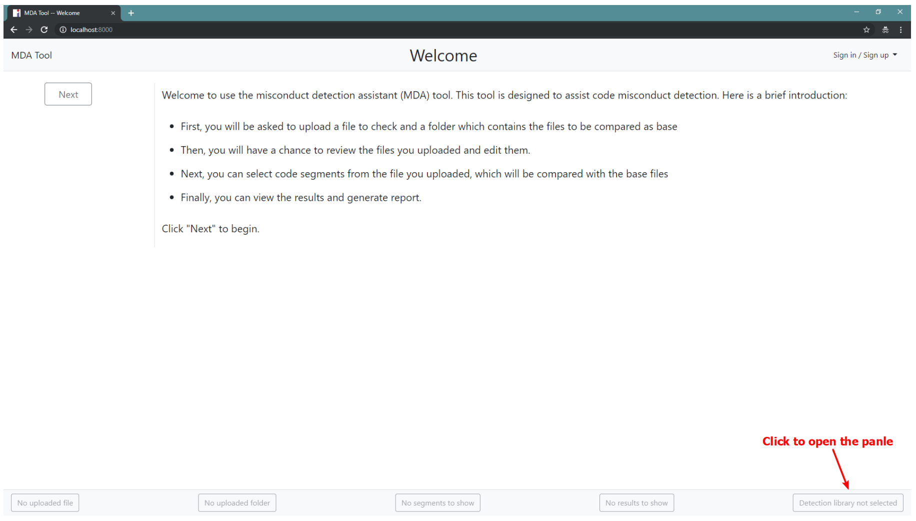
2. Normal:

   Something might affect later developing. Or something might case some further problems if it is not properly resolved.

3. Minor:

   Annoying but not harmful to the functionalities. Such as layout displaying error in different browsers.

B Programming Language Selection Panel Function in Use



Welcome

Welcome to use the misconduct detection assistant (MDA) tool. This tool is designed to assist code misconduct detection. Here is a brief introduction:

- First, you will be asked to upload a file to check and a folder which contains the files to be compared as base
- Then, you will have a chance to review the files you uploaded and edit them.
- Next, you can select code segments from the file you uploaded, which will be compared with the base files
- Finally, you can view the results and generate report.

Click "Next" to begin.

Figure 1: UI-001: Programming Language Selection Panel Screenshot 1

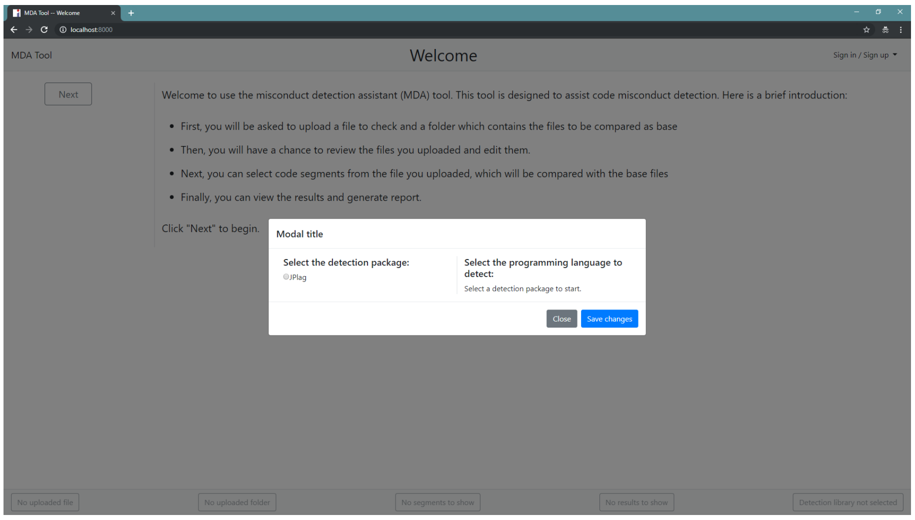
First, click the highlighted button shown in Figure 1 to open the selection panel.

Then, the panel will be shown as Figure 2.

As shown in Figure 3, the user need to first select detection package on the left side. After selected detection package, programming languages supported by the detection package will be shown on the right side. The user need to select one programming language and click “save changes”

Finally, current settings will be shown at the bottom bar as shown in Figure 4.

If the settings go wrong and no results returned (for JPlag, if the settings are incorrect, JPlag will return an empty folder to indicate that), the tool will show a “No Results” error page to remind the user that the settings might be incorrect. The error page is shown in Figure 5.


MDA Tool -- Welcome

localhost:8000

MDA Tool

Welcome

Sign in / Sign up

Next

Welcome to use the misconduct detection assistant (MDA) tool. This tool is designed to assist code misconduct detection. Here is a brief introduction:

- First, you will be asked to upload a file to check and a folder which contains the files to be compared as base
- Then, you will have a chance to review the files you uploaded and edit them.
- Next, you can select code segments from the file you uploaded, which will be compared with the base files
- Finally, you can view the results and generate report.

Click "Next" to begin.

Modal title

Select the detection package:

JPlag

Select the programming language to detect:

Select a detection package to start.

Close

Save changes

No uploaded file

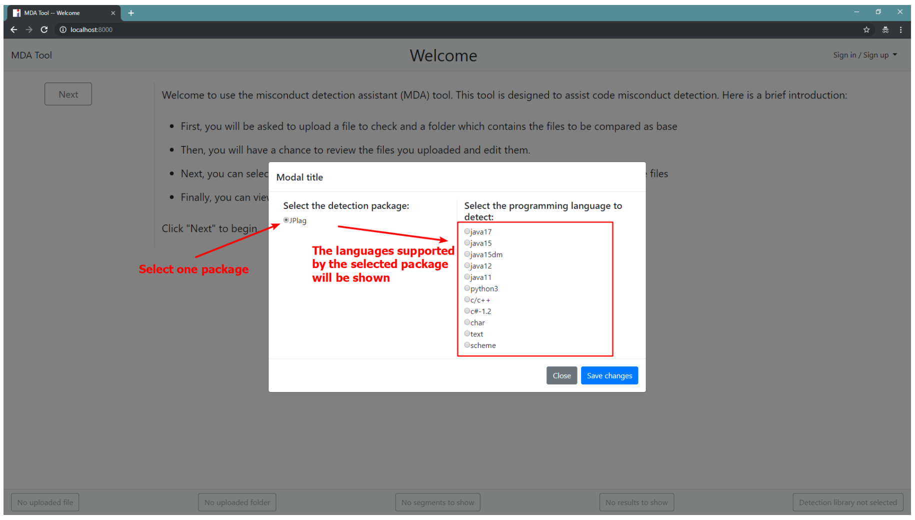
No uploaded folder

No segments to show

No results to show

Detection library not selected

Figure 2: UI-001: Programming Language Selection Panel Screenshot 2


MDA Tool -- Welcome

localhost:8000

MDA Tool

Welcome

Sign in / Sign up

Next

Welcome to use the misconduct detection assistant (MDA) tool. This tool is designed to assist code misconduct detection. Here is a brief introduction:

- First, you will be asked to upload a file to check and a folder which contains the files to be compared as base
- Then, you will have a chance to review the files you uploaded and edit them.
- Next, you can selec files
- Finally, you can vie

Click "Next" to begin.

Modal title

Select the detection package:

JPlag

Select the programming language to detect:

java17
java15
java15dm
java12
java11
python3
c/c++
c#-1.2
char
text
scheme

Close

Save changes

No uploaded file

No uploaded folder

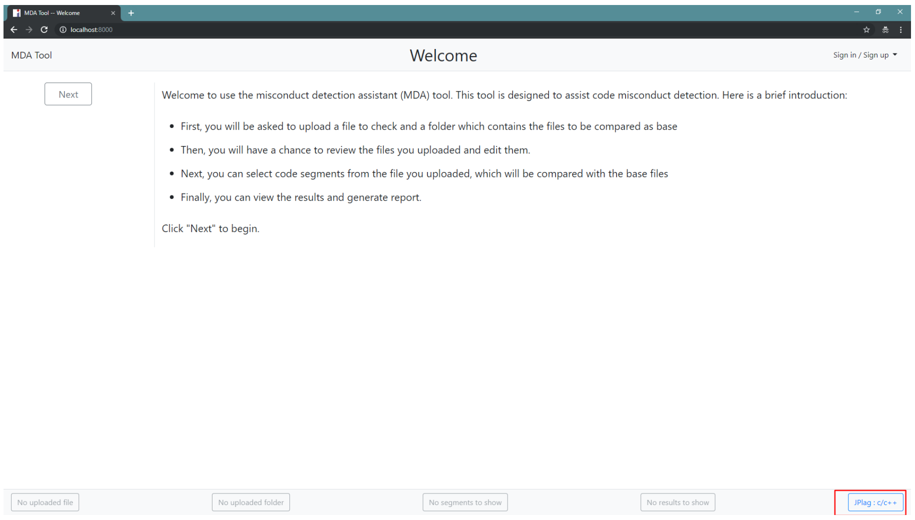
No segments to show

No results to show

Detection library not selected

Select one package
The languages supported by the selected package will be shown

Figure 3: UI-001: Programming Language Selection Panel Screenshot 3


MDA Tool -- Welcome

localhost:8000

MDA Tool

Welcome

Sign in / Sign up

Next

Welcome to use the misconduct detection assistant (MDA) tool. This tool is designed to assist code misconduct detection. Here is a brief introduction:

- First, you will be asked to upload a file to check and a folder which contains the files to be compared as base
- Then, you will have a chance to review the files you uploaded and edit them.
- Next, you can select code segments from the file you uploaded, which will be compared with the base files
- Finally, you can view the results and generate report.

Click "Next" to begin.

No uploaded file

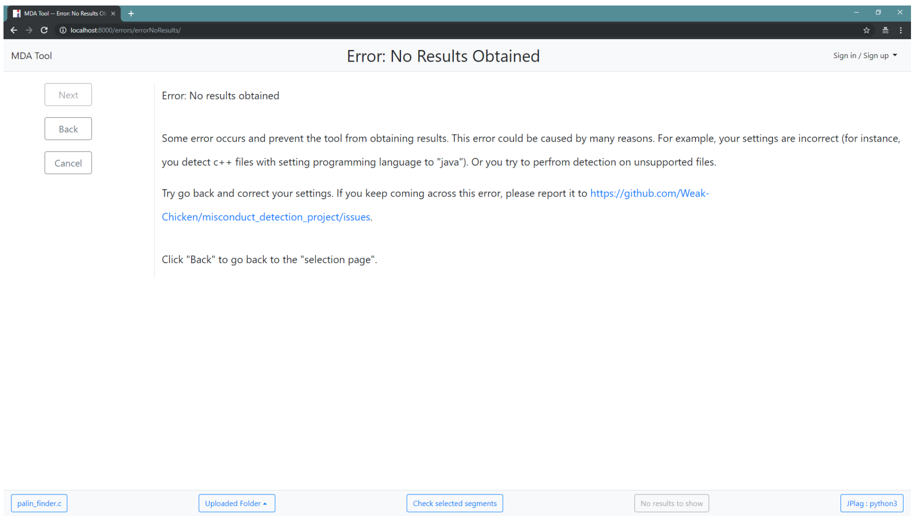
No uploaded folder

No segments to show

No results to show

JPlag : c/c++

Figure 4: UI-001: Programming Language Selection Panel Screenshot 4



Error: No Results Obtained

Error: No results obtained

Some error occurs and prevent the tool from obtaining results. This error could be caused by many reasons. For example, your settings are incorrect (for instance, you detect c++ files with setting programming language to "java"). Or you try to perfrom detection on unsupported files.

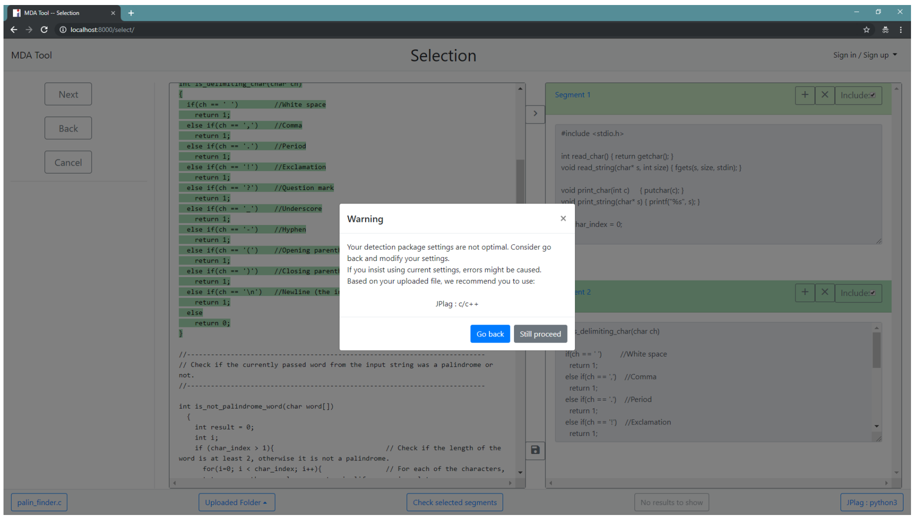
Try go back and correct your settings. If you keep coming across this error, please report it to https://github.com/Weak-Chicken/misconduct_detection_project/issues.

Click "Back" to go back to the "selection page".

Figure 5: UI-001: Programming Language Selection Panel Error Page Screenshot 5

Please notice that this error will be shown when the returned results folder is empty. Therefore, there also might be some other errors could raise this error.

C Auto Programming Language Selection Function in Use



Selection

```
int is_delimiting_char(char ch)
{
  if(ch == ' ')         //White space
    return 1;
  else if(ch == ',')    //Comma
    return 1;
  else if(ch == '.')    //Period
    return 1;
  else if(ch == '!')    //Exclamation
    return 1;
  else if(ch == '?')    //Question mark
    return 1;
  else if(ch == '_')    //Underscore
    return 1;
  else if(ch == '-')    //Hyphen
    return 1;
  else if(ch == '(')    //Opening parenth
    return 1;
  else if(ch == ')')    //Closing parenth
    return 1;
  else if(ch == '\n')   //Newline (the i
    return 1;
  else
    return 0;
}

//---------------------------------------------------------------------------
// Check if the currently passed word from the input string was a palindrome or
not.
//---------------------------------------------------------------------------

int is_not_palindrome_word(char word[])
  {
    int result = 0;
    int i;
    if (char_index > 1){                           // Check if the length of the
word is at least 2, otherwise it is not a palindrome.
      for(i=0; i < char_index; i++){               // For each of the characters,
```

Segment 1

```
#include <stdio.h>

int read_char() { return getchar(); }
void read_string(char* s, int size) { fgets(s, size, stdin); }

void print_char(int c)     { putchar(c); }
void print_string(char* s) { printf("%s", s); }

har_index = 0;
```

```
_delimiting_char(char ch)

if(ch == ' ')       //White space
  return 1;
else if(ch == ',')    //Comma
  return 1;
else if(ch == '.')    //Period
  return 1;
else if(ch == '!')    //Exclamation
  return 1;
```

Warning

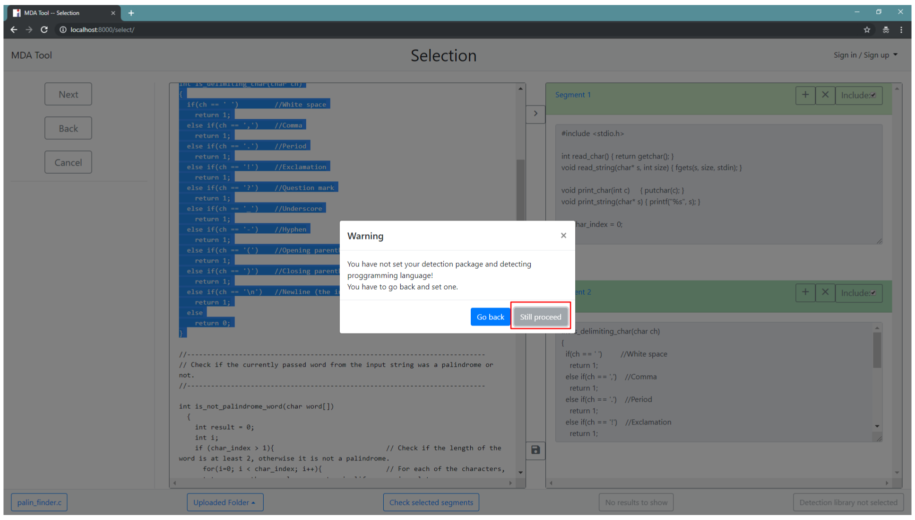
Your detection package settings are not optimal. Consider go back and modify your settings.
If you insist using current settings, errors might be caused.
Based on your uploaded file, we recommend you to use:

JPlag : c/c++

Figure 6: UI-002: Auto Programming Language Selection Screenshot 1

As shown in Figure 6, when the user is not using the optimal settings saved in the server, the auto programming language selection function will give corresponding suggestions and warning to the user. However, the user can ignore the suggestion and still proceed.

If the user has not provided any settings, the function will stop the user from performing detection and require the user to go back and give settings. As shown in Figure 7, the "still proceed" button is unclickable.



```
{
  if(ch == ' ')        //White space
    return 1;
  else if(ch == ',')   //Comma
    return 1;
  else if(ch == '.')   //Period
    return 1;
  else if(ch == '!')   //Exclamation
    return 1;
  else if(ch == '?')   //Question mark
    return 1;
  else if(ch == '_')   //Underscore
    return 1;
  else if(ch == '-')   //Hyphen
    return 1;
  else if(ch == '(')   //Opening parent
    return 1;
  else if(ch == ')')   //Closing parent
    return 1;
  else if(ch == '\n')  //Newline (the i
    return 1;
  else
    return 0;
}

//----------------------------------------------------------------------------
// Check if the currently passed word from the input string was a palindrome or
not.
//----------------------------------------------------------------------------

int is_not_palindrome_word(char word[])
  {
    int result = 0;
    int i;
    if (char_index > 1){                            // Check if the length of the
word is at least 2, otherwise it is not a palindrome.
      for(i=0; i < char_index; i++){                // For each of the characters,
```

Warning

You have not set your detection package and detecting proggramming language!
You have to go back and set one.

```
#include <stdio.h>

int read_char() { return getchar(); }
void read_string(char* s, int size) { fgets(s, size, stdin); }

void print_char(int c)     { putchar(c); }
void print_string(char* s) { printf("%s", s); }

har_index = 0;
```

```
_delimiting_char(char ch)
{
  if(ch == ' ')        //White space
    return 1;
  else if(ch == ',')   //Comma
    return 1;
  else if(ch == '.')   //Period
    return 1;
  else if(ch == '!')   //Exclamation
    return 1;
```

Figure 7: UI-002: Auto Programming Language Selection Screenshot 2