TOOL FOR CHECKING CODE COMPATIBILITY WITH PROGRAMMING LANGUAGE RELEASES

Jehan Ryan Benjamin

179308M

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

May 2019

TOOL FOR CHECKING CODE COMPATIBILITY WITH PROGRAMMING LANGUAGE RELEASES

Jehan Ryan Benjamin

179308M

This dissertation submitted in partial fulfillment of the requirements for the Degree of MSc in Computer Science specializing in Software Architecture

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2019

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works.

....
Jehan Ryan Benjamin Date

The above candidate has carried out research for the Masters thesis under my supervision.

....
Dr. Indika Perera Date

ABSTRACT

Programming languages tend to evolve with new methodologies, user requirements, technology, security constraints etc. In order to stay relevant with changing demands, programming languages needs to keep adopting new features and enhancements. When programming languages make a new version release, it creates a necessity to upgrade the systems developed using older versions. Based on the situation of the project status, ongoing, completed or new, developers need to make the decision whether to proceed with the current version or change to new release. If the developers decides to stay with the older version of release they wouldn't get the latest features and fixes, eventually the system code would be outdated. This would lead the system to be vulnerable and prone to compatibility concerns.

The main intention of this research is to develop a simplified tool for checking code compatibility along with programming language version releases. Currently there few ways that developers check the compatibility (e.g. IDE plugins, CLI commands, compile time reports, etc.) but these are with limitations and not user friendly, aim of this research is to provide a combination of those features, in a more easy to use, optimized, customizable compatibility tool. The users will be able to run compatibility test on the desired version effortlessly, developers would be able to add their own custom rule sets to verify the project code across the selected version by using the tool. Once the tool completes the validation process, it will generate a user friendly report with the findings. The report would contain charts with error types and percentages, filenames, error line number, location, error type and possible fixes, which would be useful for developers. The developers would be able to fix the notified sections and re-run the verification process. The tool will also provide the option to get a deployable image along with updated code and version.

Keywords: programming languages, versions, code compatibility, deployable image.

ACKNOWLEDGEMENT

I like to convey my appreciation and gratitude to the project supervisor Dr. Indika Perera, for the knowledge, guidance and suggestions during this research project which was a driving force behind the completion of this work on time.

My heartfelt gratitude to my friends especially to Mr. Tiran Wijesekara, Miss. Tirsha Melani, Mr. Vilochane Vidyarathne, and all other friends which were not mentioned here whose friendship, hospitality and support in the preparation during this research.

I'd like to thank my current company Netstarter (Pvt) Ltd, especially to my project manager and colleagues for the endless support and words of encouragements throughout.

Finally, I would like to extend my deepest gratitude to my parents for their never ending support and encouragement.

TABLE OF CONTENTS

DECLARATION i

ABSTRACT ii

ACKNOWLEDGEMENT iii

LIST OF FIGURES vi

LIST OF TABLES ix

LIST OF ABBREVIATIONS x

CHAPTER 01: INTRODUCTION 1

1.1 Programming language Types 3

1.2 High-level languages 4

1.3 Programming Language Version 7

1.3.1 Benefits of version updates 8

1.3.2 Cost of updates 8

1.3.2 Dual version support 9

1.4 Backwards compatibility 9

1.5 Problem / Opportunity 11

1.6 Motivation 12

1.7 Objectives 12

CHAPTER 02: LITERATURE REVIEW 14

2.1 Programming language complications 15

2.2 Tools for sensing errors on software projects 16

2.3 PHP Code Fixer 18

2.3.1 Usage 19

2.4 Mining Software Repositories Tools 20

2.5 Deprecated methods 20

2.6 PHP built in lint command 21

CHAPTER 03: METHODOLOGY 23
3.1 System 24
3.1.1 High level architecture 25
3.1.2 Components architecture 26
3.2 Input output and processing 27
3.2.1 Custom rule sets 29
3.2.2 Type of notification outputs 30
3.2.3 Deployment process 30
4.1 Scope of implementation 32
4.2 Compatibility checker 33
4.2.1 Application implementation 33
CHAPTER 05: EVALUATION 49
5.1 Usability 50
5.1.1 Readiness check 50
5.1.2 Docker build, run process. 52
5.2 Application results evaluation 56
5.2.1 Code sniffer results 56
5.2.2 Evaluation of the test results 59
5.3 Reporting and error information verification 61
CHAPTER 06: CONCLUSION AND FUTURE WORK 63
Summary 64
6.1 Problems encountered 65
6.2 Future work 65
REFERENCES 67
APPENDIX A 70

LIST OF FIGURES

Figure 1: Computer language and its types [26] 3

Figure 2: Transitions of a High-level Language Program [30] 5

Figure 3: The fifteen most popular languages on GitHub [27] 6

Figure 4: Top 10 Programming Languages [3] 6

Figure 5: PHP error reporting options 10

Figure 6: PHP supported versions [9] 13

Figure 7: C++ Semantic error sample code 15

Figure 8: PHP Related Semantic Error Code Sample 16

Figure 9: Zend Studio Semantic Analysis [28] 16

Figure 10: PHP CodeSniffer findings against PEAR coding standards [11] 17

Figure 11: LGTM alert samples [12] 18

Figure 12: PHP Code Fixer terminal command 19

Figure 13: phpcf sample results [13] 19

Figure 14: Candoia work flow [15] 20

Figure 15: PHP Deprecated Methods Finding Script 21

Figure 16: PHP Lint script 22

Figure 17: Proposed work flow 24

Figure 18: Proposed System Architecture 25

Figure 19: Components architecture 26

Figure 20: system input output 27

Figure 21: Activity flow 29

Figure 22: Custom rules 30

Figure 23: package.json 34

Figure 24: Electron init code 35

Figure 25: Readiness check 35

Figure 26: Docker checker 36

Figure 27: Docker Install option 36

Figure 28: Docker information 36

Figure 29: Docker project info 37

Figure 30: Docker image create code sample 38

Figure 31: Container terminal output 39

Figure 32: Docker starting view 40

Figure 33: Docker and Project information 41

Figure 34: Docker image build 42

Figure 35: Dockerfile contents 42

Figure 36: supervisord.conf file 43

Figure 37: Composer package.json 44

Figure 38: Custom rule set 44

Figure 39: Pie chart 45

Figure 40: Error notification 46

Figure 41: Warning notification 46

Figure 42: Container diff 47

Figure 43: Docker deployment process [21] 48

Figure 44: log files 50

Figure 45: Application landing page 51

Figure 46: Docker information 51

Figure 47: Project info form 52

Figure 48: Docker image run and container start 53

Figure 49: container and project info 53

Figure 50: Docker executes and results 54

Figure 51: Phpstorm code sniffer setup 54

Figure 52: Docker build commands 55

Figure 54: webpack build scripts 55

Figure 53: platform related builds 55

Figure 55: Terminal summary output 57

Figure 56: report summary on application 57

Figure 57: misleading information 62

Figure 58: Updated pie chart view 62

LIST OF TABLES

Table 1: Types of High Level Language 4

Table 2: Compatibility checker Input and result output 27

Table 3: Sample projects information 56

Table 4: Compatibility results summary 58

Table 5: Compatibility results summary for 5.6 59

Table 6: Compatibility results summary for 7.0 60

Table 7: Results summary for 7.1 61

LIST OF ABBREVIATIONS

Abbreviation	Description
PEAR	PHP Extension Add-On Repository
MSR	Mining Software Repositories
PHP	Recursive acronym for PHP: Hypertext Preprocessor
CPU	Central Processing Unit
PM	Project Manager
BA	Business Analysis
IT	Information Technology
HTML	HyperText Markup Language
OOP	Object Oriented Programming
IDE	Integrated Development Environment
LGTM	Looks Good To Me
CLI	Command Language Interpreter
CSV	Comma Separated Values
XML	Extensible Markup Language
JS	JavaScript
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
Amazon ECS	Amazon Elastic Container Service