

**FRAMEWORK TO MIGRATE ANGULARJS BASED
LEGACY WEB APPLICATION TO REACT
COMPONENT ARCHITECTURE**

Thilanka Kaushalya Lanka Geeganage

179328A

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

March 2021

**FRAMEWORK TO MIGRATE ANGULARJS BASED
LEGACY WEB APPLICATION TO REACT
COMPONENT ARCHITECTURE**

Thilanka Kaushalya Lanka Geeganage

179328A

Thesis submitted in partial fulfillment of the requirements for the degree Master of
Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

March 2021

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for 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 University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, 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 (such as articles or books).

Signature:

Date:.....

Name: Thilanka Kaushalya Lanka Geeganage

I certify that the declaration above by the candidate is true to the best of my knowledge and that this report is acceptable for evaluation for the Masters thesis.

Signature of the supervisor:

Date:.....

Name: Dr. Indika Perera

ABSTRACT

Due to the increased complexity and the high expectation of the modern web applications, the legacy applications built on top of previous generation frontend frameworks such as AngularJS fail to serve both end users and developers. The AngularJS based legacy applications start poorly performing when the application grows larger. At the same time they become highly challenging for developers to maintain and add new functionality over time. The next generation component based frontend frameworks such as React has outperformed AngularJS from both the end user and developer expectations. Since the whole industry has evolved to gain the benefits of the component architecture, sooner or later all these legacy applications need to migrate to new component based architecture. The concepts, patterns and the architecture of the old school AngularJS application and the component based React applications are different so the migration is not a straightforward step. The successor of AngularJS which is named as Angular2 provides some tooling to support migrating old AngularJS applications to Angular2 but those are not directly compatible with React migration. This makes the organizations and developers who are planning to migrate AngularJS application to stick with Angular2 which is a less popular and flexible framework when compared with React. This research tries to define a clear migration path from AngularJS applications to React applications which will allow organizations and developers to choose React also as a goto framework option and will save hundreds of research and development hours in setting up the migration.

This research compares and contrasts approaches of legacy frontend application migrations with respect to steps, best practices, technologies tools and pain points. By comparing different approaches, the research provides a step by step guideline which can be referred to and followed when migrating any AngularJS based legacy application to component base React application. These guidelines are organized as a migration framework with steps of refactorings and provide as the first outcome of this research. As the second step of this research, a migration assistant tool which is called as Ng-React Copilot was implemented to guide the developer through the migration steps and apply them to the existing code base. The tool was developed by converting the critical refactorings proposed in the framework into a set of detection algorithms and providing the ability to scan against the provided codebase as a command line tool as well as an integrated tool with populer IDEs.

The framework and the tool were validated by applying them to selected small, medium and enterprise level AngularJS legacy applications. One application was selected from each category and used the tool and the framework to step by step migrating to React where the small application was fully migrated to React while the medium and enterprise applications were migrated partially with the interest of the time. The results from the migration assistant tool and the framework were collected and validated the accuracy of them and except for a few false positives, the migration assistant tool was detecting the required refactoring accurately and was making the migration straight forward. Improving the algorithms and configurations to avoid false positives, improve the tool to guide the users as a step by step workflow rather than showing all the refactorings at once, automating few more time consuming manual refactorings, improving the framework to address external library coupling and improve the tool to guide the user on where in the application to start the migration were identified as future improvements for the identified limitations during the evaluation process. By looking at the evaluation results, the migration framework and the Ng-React Copilot tool can be considered as good starting materials to get assistance for AngularJS to React application migration and the effectiveness can be further improved by applying the identified future improvements.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Dr. Indika Perera, my project supervisor for providing advice and guiding the project into success. Even though I had an idea about the domain initially I didn't have a clear idea on how to converge the scope of the research into a meaningful outcome. Dr. Indika helped me and guided on how to narrow down the scope and focusing on building up the framework proposed in this report. Also I would like to express my gratitude to my family for supporting me in focusing on the research. Last but not least I wish to thankful for all my colleagues who have encouraged me to continue the research by expressing their experiences.

TABLE OF CONTENTS

DECLARATION	i
ABSTRACT	ii
ACKNOWLEDGEMENTS	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES.....	viii
LIST OF ABBREVIATIONS	x
1. INTRODUCTION	1
1.1. From Javascript libraries to frameworks	1
1.2. What did AngularJS offer	2
1.3. Limitations and anti-patterns in AngularJS applications	4
1.4. AngularJS successors which provides component architecture	8
1.5. Using React is a business decision, not a technology decision[15]	10
1.6. Problem.....	12
1.7. Objectives.....	13
2. LITERATURE REVIEW	15
2.1 Choosing the migration strategy	15
2.2 Prepare AngularJS code for migration	16
2.3. Migrate component by component	26
2.4. Migrate the forms	29
3. METHODOLOGY	31
3.1. A migration framework from AngularJS to React	31
3.2. Migration assistant tool.....	35
3.3. Set of algorithms to detect the required refactoring	47

3.4. Evaluating the migration framework and the assistant tool.....	48
4. IMPLEMENTATION.....	50
4.1. Implementation of the Ng-React-Copilot tool	50
4.2. Implementation of the refactoring detection rules	52
5. RESULTS AND EVALUATION.....	82
5.1. Migrating small AngularJS applications having less complicated code	83
5.2. Migrating medium level of AngularJS applications.....	87
5.3. Migrating Enterprise level of AngularJS applications	91
5.4. Evaluation of the results	96
6. CONCLUSION	100
6.1. Limitations	103
6.2. Future work.....	104
7. REFERENCES.....	106

LIST OF FIGURES

Figure 1: How complex are the AngularJS aspects and features [3]	6
Figure 2: Data flow in a react application and a MV* application.....	12
Figure 3: How data flow happens through React2Angular middleware.....	26
Figure 4: Workflow of the migration framework.....	34
Figure 5: Ng-React Copilot tool logo in the command line	44
Figure 6: Ng-React Copilot component diagram	45
Figure 7: Ng-React Copilot flow diagram	46
Figure 8: Ng-React Copilot select scan mode	50
Figure 9: Ng-React Copilot highlights of the analysis results.....	51
Figure 10: Ng-React Copilot summary report.....	51
Figure 11: Ng-React Copilot detailed report.....	52
Figure 12: Ng-React Copilot rules highlighting a required refactoring in a code opened with Visual Studio	52
Figure 13: AST for detecting a directive.....	54
Figure 14: AST for detecting a directive factory defined as a function.....	65
Figure 15: AST for detecting a directive factory defined as an identifier	67
Figure 16: AST for detecting a directive factory defined as function returned from a function expression	69
Figure 17: Initial scan results for Angular-Phonecat application.....	83
Figure 18: Scan results for Angular-Phonecat application after generating PhoneDetailsComponent.....	84
Figure 19: Highlighted AngularJS specific feature refactoring in PhoneDetailsComponent.....	85
Figure 20: Highlighted AngularJS scope reference refactoring in PhoneDetailComponent	85
Figure 21: Migrated PhoneListComponent.....	86
Figure 22: Migrated PhoneDetailedComponent.....	86
Figure 23: Initial scan results for AngularJS CRUD application	87
Figure 24: Scan results for ToolbarComponentof the AngularJS CRUD application	89
Figure 25: Scan results for FormComponent of the AngularJS CRUD application ..	90

Figure 26: Migration completed Login Form of the AngularJS CRUD application..	91
Figure 27: Initial scan for OrangeHRM application	92
Figure 28: Scan results for List directive of the the OrangeHRM application	93
Figure 29: Scan results for ListComponent of the the OrangeHRM application	94
Figure 30: Migration completed ListComponent of the OrangeHRM application	96
Figure 31: Workflow of the migration framework	102

LIST OF TABLES

Table 1: Comparison between Angular 2 and React[14]	9
Table 2: Alternatives for AngularJS features[4].....	24
Table 3: Refactoring and their groupings.....	33
Table 4: List of refactoring and their automation feasibility	38
Table 5: Sample AST for JS code.....	42
Table 6: Event list of the HTMLHint code scanner.....	43
Table 7: Summary of selected technologies for the migration process	44
Table 8: Code samples to define an angular directive and a component.....	53
Table 9: Code samples to define an angular using different syntaxes.....	56
Table 10: Pseudo code for rule - Place all the code of a directive together.....	58
Table 11: Code samples to define an angular directive restrict.....	59
Table 12: Pseudo code for rule - Favor directives as elements over attributes, comment and classes.....	59
Table 13: Code samples on how to use ng-transclude directive	60
Table 14: Pseudo code for rule - Eliminate transclusion	60
Table 15: Code samples on how to use ng-include directive.....	61
Table 16: Pseudo code for HTMLHint rule - Extract directive from large templates.....	62
Table 17: Pseudo code of the isTwoWayDataBindingNotUsed method.....	64
Table 18: Pseudo code of the extractFactoryFunctionObject method when directive factory defined as a function	66
Table 19: Pseudo code of the extractFactoryFunctionObject method when directive factory defined as an identifier	68
Table 20: Pseudo code of the extractFactoryFunctionObject method when directive factory returned from another function.....	70
Table 21: Pseudo code of the extractFactoryFunctionObject method when directive function defined with dependency injection	71
Table 22: Consolidated pseudo code of the isTwoWayDataBindingNotUsed method	72
Table 23: List of AngularJS inbuilt directives selected to detect inside the JSX.....	74
Table 24: AST generated for AngularJS directive usage detection in JSX	75

Table 25: Pseudo code for algorithm to detect AngularJS directive usage in JSX	76
Table 26: AST generated for variable references used in JSX.....	78
Table 27: Pseudo code for algorithm to detect scope references in in JSX	80
Table 28: Scan results for Angular-Phonecat application after generating PhoneDetailComponent	84
Table 29: Initial scan results for AngularJS CRUD application	88
Table 30: Scan results for ToolbarComponentof the AngularJS CRUD application .	89
Table 31: Scan results for FormComponent of the AngularJS CRUD application....	90
Table 32: Initial scan results for OrangeHRM application	92
Table 33: Initial scan results for List directive in OrangeHRM application.....	93
Table 34: Initial scan results for ListComponent in OrangeHRM application	94
Table 35: Summary of all detected refactorings.....	97
Table 36: Migration Framework.....	101

LIST OF ABBREVIATIONS

DOM	Document Object Model
MVC	Model View Controller
MVVM	Model View View Model
MVP	Model View Presenter
MV*	Model View Any
AST	Abstract Syntax Tree
JS	Javascript
CSS	Cascading Style Sheets
XML	Extensible Markup Language
JSX	Javascript XML