COMPREHENSIVE TESTING FRAMEWORK FOR MICROSERVICES

Vajiramal Vilochane Vidyarathne

(179355D)

M.Sc. in Computer Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

March 2021

COMPREHENSIVE TESTING FRAMEWORK FOR MICROSERVICES

Vajiramal Vilochane Vidyarathne

(179355D)

This dissertation submitted in partial fulfilment 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

March 2021

DECLARATION

I declare that this is my own work and this MSc Thesis Project Report does not incorporate without acknowledgment 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 acknowledgment 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.

---------- ----------

B.K.V. V.Vidyarathne Date

I certify that the declaration above by the candidate is true to the best of my knowledge and that this project report is acceptable for evaluation for the MSc Thesis (CS6997).

---------- ----------

Dr. Indika Perera Date

ABSTRACT

An emerging trend has begun with the introduction of microservices architecture, transforming monolithic applications into microservices. These services are focused on smaller independent autonomous services, which encourage development, deployment, scale, and maintain each service independently with improved parallel development among multiple autonomous development teams. Unlike the monolith microservices, architecture introduces additional testing challenges.

Microservices integration testing, a crucial process that ensures the communication paths are correct while interacting with other services. Because of this unique architecture, testing integration in isolation impose a challenge due to many reasons, such as each microservice is developed and maintained by individual autonomous teams, unavailability or instability of service due to parallel continuous rapid development, deployment and difficulty in setting up all the services and seed the necessary data for testing in a local or a remote development environment. This process requires effort to initialize and maintain the resource-intensive environment. Currently, available testing approaches or tools have particular limitations. Such approaches or tools require extra effort to create, initialize and maintain with the rapidly changing unstable services and requirements. Gradually with the introduction of new services as the number of services is greater this process turns in to a development overhead.

The main objective of this research is to overcome the integration testing challenges in REST-based microservices, via a service virtualization solution based on widely adapted Open API specification. The proposed implemented solution was evaluated against the existing services. The solution is proven to mitigate the identified integration testing challenges in microservices and successfully emulates the tested producer services via the virtualization of its Open API specification. The solution facilitates reproducing the edge cases and failures, such is difficult to produce in the real services, and this helps to reduce the development time significantly. The solution can evolve with the services due to the OAS of the respective service reflects the producer services' additions or its changes. Because of this reason, unlike the existing approaches or tools, this solution does not require any maintenance via mocking, simulation by the record and replay requests or as a third-party dependency package or a library to the producer and consumer source code to reflect the service changes or additions. This report concludes by mentioning research contributions, limitations, and future works of the implemented solution.

Keywords: monolithic, microservice, REST, integration, testing, development, architecture, OAS.

ACKNOWLEDGEMENTS

I would first like to express my sincere gratitude to my research supervisor Dr. Indika Perea for the guidance, knowledge, suggestions, and encouragement throughout the research which steered for the successful completion of this research.

Secondly, I would like to thank my family members, friends, and peers for the support and encouragement for the research work.

Finally, I would like to thank my current workplace CMS (PVT) Ltd, and my colleagues for their support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

APPENDICES

LIST OF ABBREVIATIONS AND SYMBOLS

OAS	Open API specification
API	Application programming interface
REST	Representational state transfer
HTTP	Hypertext Transfer Protocol
Groovy	Java syntax compatible object oriented programming language
PHP	Hypertext Preprocessor
CPU	Central Processing Unit
RAM	Random access memory
HDD	Hard disk drive
XML	Extensible Markup Language
JSON	JavaScript Object Notation
NPM	Node package manager
GraphQL	Open source data query and manipulation language
CTFM	Comprehensive testing framework for microservices