**A WEB APPLICATION**

**FOR CODE-TESTING**

**By**

**Huzaifa Yusufbhai Mankda (16CEUOS117)**

**Khadela Kishan (16CEUOG077)**

**A project submitted**

**In**

**partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF TECHNOLOGY**

**in**

**Computer Engineering**

**Internal Guide**

*Prof. Pandav Patel*

*Assistant Professor*

*Dept. of Comp. Engg.*



**Faculty of Technology Department of Computer Engineering Dharmsinh Desai University**

**April 2019**

CERTIFICATE

This is to certify that the project work titled “Semi-Automated Code Testing Environment” is the bonafide work of

Huzaifa Yusufbhai Mankda (16CEUOS117)

Khadela Kishan (16CEUOG077)

carried out in the partial fulfillment of the degree of Bachelor of Technology in Computer Engineering at Dharmsinh Desai University in the academic session

December 2018 to April 2019.

Prof. Pandav Patel

Asst. Prof.

Dept. of Computer Engg.



Dr. C. K. Bhensdadia

Head,

Dept. of Computer Engg.

**Faculty of Technology Department of Computer Engineering Dharmsinh Desai University**

**April 2019**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| Chapter |  | Page |
| I. | Introduction | 1 |
| II. | About the System | 29 |
| III. | Analysis and Design | 56 |
| IV. | Implementation |  |
| V. | Screenshots and Demo |  |
| IV. | Conclusion and Future Extensions | 102 |

**Chapter 1**

Introduction



Testing is an integral aspect of any System. The system must pass rigorous testing phase before it is ready to deploy. Here we have tried to create a tool that performs unit testing for standalone code or functions. Generating Test Cases is a crucial task for system testing. Using this tool is can very helpful for white box testing of various functions. This tool works on some assumed threshold for correctness of code. Continuously test cases are generated and primary as well as secondary codes are then run. The generated output files are compared for exactness. If the expected result is achieved, next test case is generated. This process continues, until time threshold for correctness is achieved. Thus, verdict is generated accordingly.

The result file of secondary code contains the result. This tool can be very effective to perform stub-based testing. Using time-based Threshold is a trade-off between time and randomness. Though overall, it can be observed that time required to generate corner cases, is way more than the threshold testing using random generation. Also generation of random test cases ensures that almost all the corner cases are handled.

**Chapter 2**

About the System



**Technologies Used:**

**Front End:** HTML, CSS, JavaScript

**Back End:** C#

**Framework:** ASP .NET, Bootstrap, WCF Services

**Development Tool:** Visual Studio

**Diagram Tool:** Umlet

**SRS Document:**

After Analysing all the requirements of system, the below Software Requirement Specification is prepared.

Software Requirements Specification

for

Semi-Automated Code Testing Environment

Version 1.0 approved

Prepared by

Huzaifa Mankda(CE-039)

Kishan Khadela(CE-051)

Dharmisnh Desai University

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Intended Audience and Reading Suggestions 1

1.3 Product Scope 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 Operating Environment 2

2.4 Design and Implementation Constraints 2

2.5 Assumptions and Dependencies 3

3. System Features 4

3.1 Code Testing 4

4. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

The purpose of this tool is to provide a white box testing environment where the codes are tested based on time threshold observed for correctness of code. The subsequent result file contains the verdict of success or failure and in case of failure, it consists of testcase that fails the result.

## Intended Audience and Reading Suggestions

The audience of this product is very large. This tool can be used for industrial level testing and domestic level testing. The audience of this product include (but not limited to) Industries, University and school Professors, Students etc.

## Product Scope

The product can be used at industrial scale without hesitation.

# Overall Description

## Product Perspective

The product is made with a vision to ease the process of test generation for software testing. This product is based on time-based threshold, hence white box testing becomes easier due to random test case generation. Thus, corner cases are automatically taken care of.

## Product Functions

All in all the product helps to detect the difference in behavior of two codes on certain test cases, or it determines that both the codes gives exactly same output.

## Operating Environment

The Product is developed in C# and hence requires a Windows server to host the tool.

## Design and Implementation Constraints

The tool is implemented in WCF Services. Hence it has loosely coupled design.

## Assumptions and Dependencies

The Product requires a Windows server as host.

# System Features

## Code Testing

3.1.1 Description and Priority

The most important and the only feature of the system, It test’s for exactness in the output of two codes.

3.1.2 Functional Requirements

The system takes two code file and a test generation file as input and generates the corresponding result file.

Input: Code files and test generation file

Output: Verdict file

# Other Nonfunctional Requirements

## Performance Requirements

The tool needs pre-configured environment of languages like C++, JAVA, python and C, to work and test on codes of these languages.

## Safety Requirements

The tool extensively depends on executer Service hence it has to be explicitly taken care of.

## Security Requirements

The versioning of language package library as well as services must be well maintained. This is one of the greatest security vulnerabilities.

## Software Quality Attributes

The tool needs Windows server to host WCF service. Also deal with system, with lot of patience as it is based on time threshold for testing.

**Chapter 3**

Analysis and Design



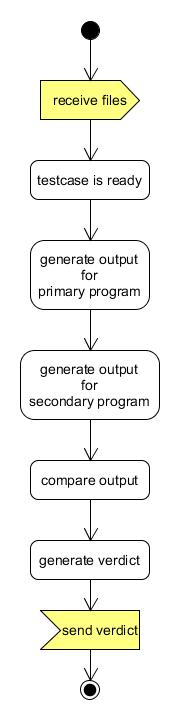


Fig: Activity Daigram for the sytem

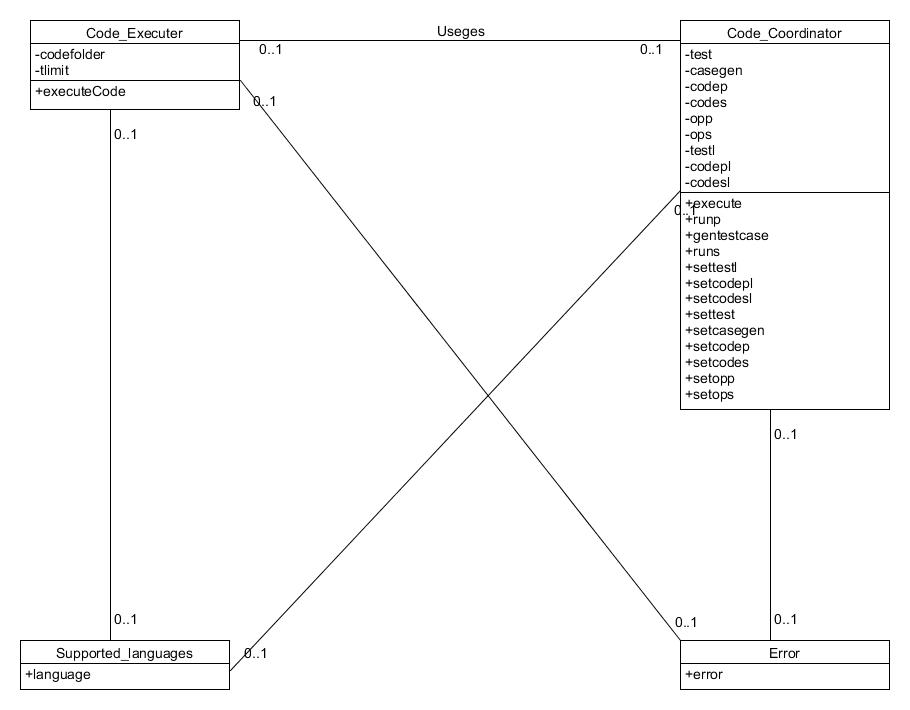


Fig: Class Diagram of the system

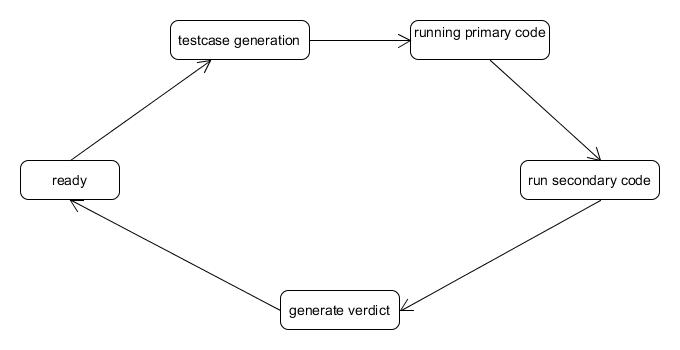


Fig: State Diagram of the System

**Chapter 4**

Implementation



The following are Components of the System:

1. **CodeExecuter Service**

This Service takes the code file, input file, output file and language of code and Executes the code and generates the output. Thus, the basic task of this service is to take a code file and generate output for it.

**2.CodeCoordinaton Service**

This Service takes Test Generation file, and primary as well as Secondary code file and compares them. First, the Test Generation file runs and produces input for other code files. Then Primary code runs with produced Test Cases and produces, primary Output. Then Secondary Code file runs using Test Cases and produces Secondary Output. Now, both the output files are compared, if output are alike then a new test case is produced, and the cycle continues until time threshold is reached. If output doesn’t match then, a verdict file is generated which is the result of the testing.

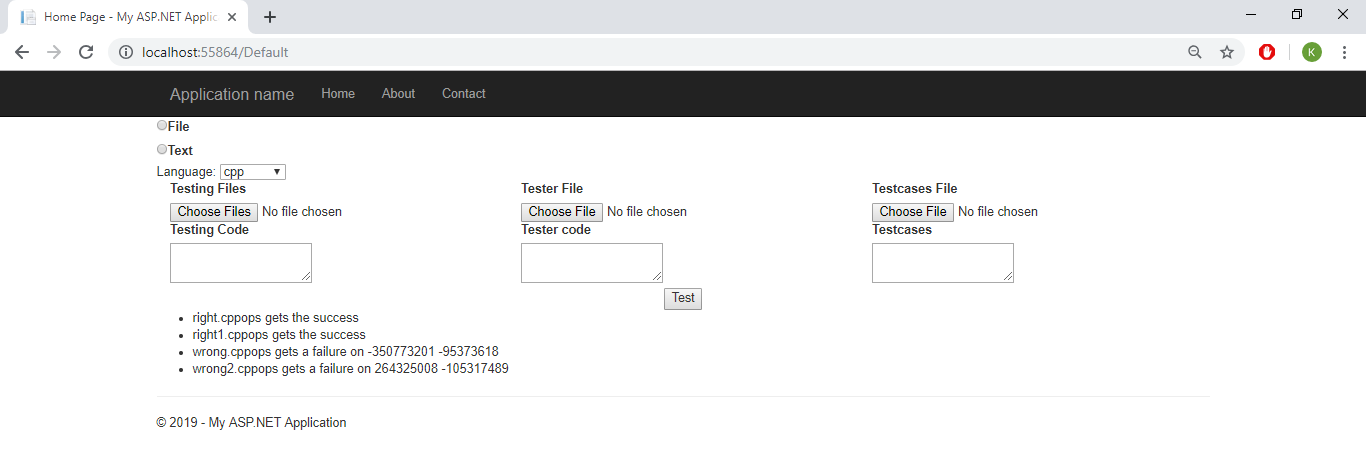
**3.** **Client**

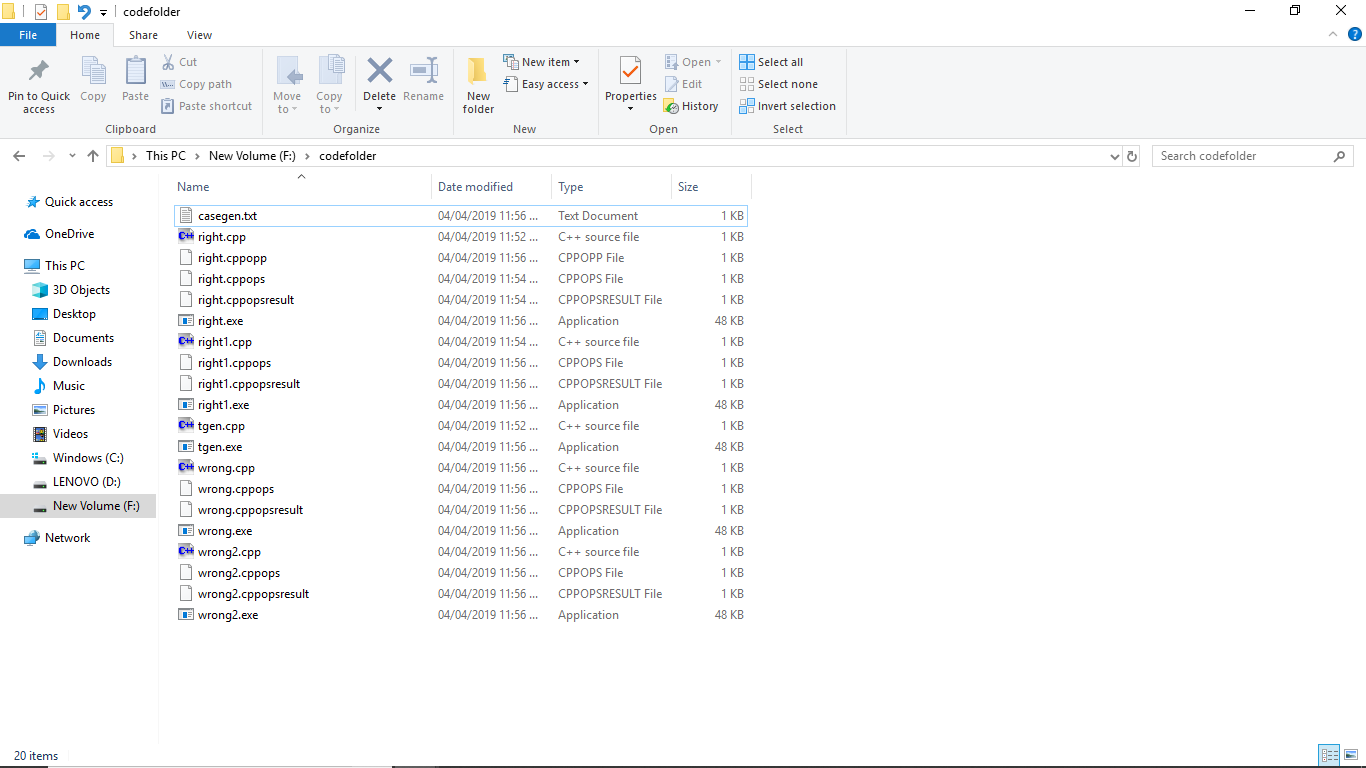
This Client Web-App takes all the required input files and passes it to the Coordination Service and generates the Verdict. There is separate output file generated for each testing code.

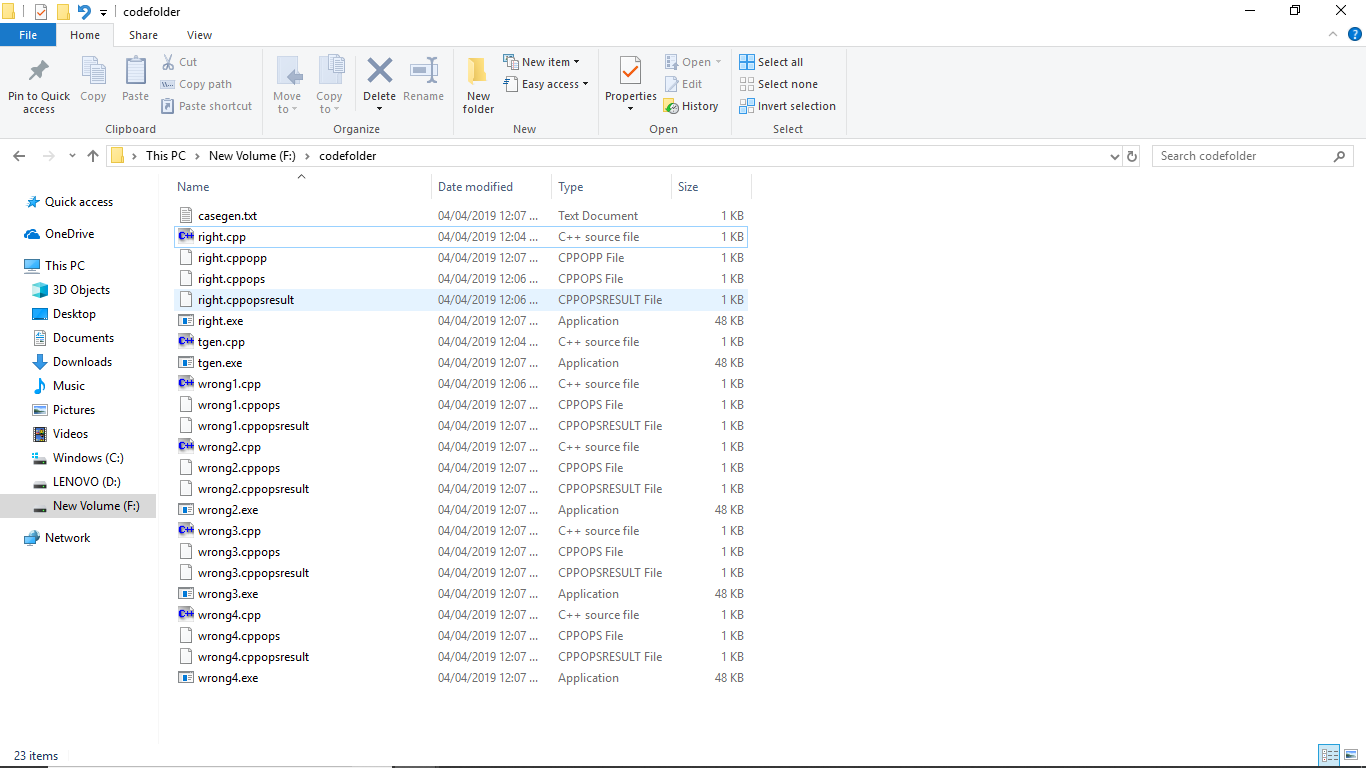
**Chapter 5**

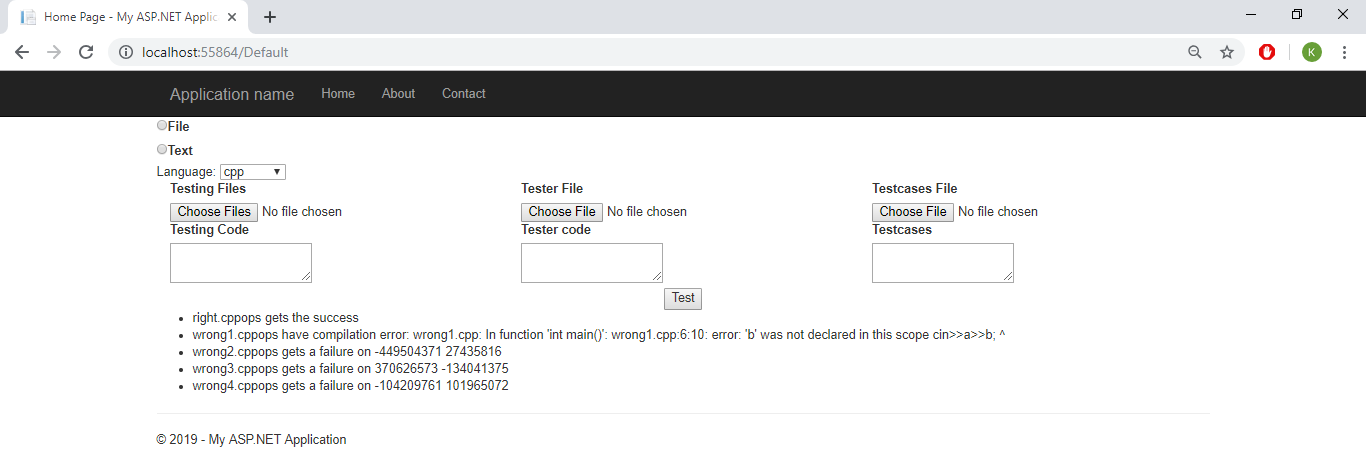
Screenshots and Demo











**Chapter 6**

Conclusion and Future Extensions



As of now the project is semi-automated as it needs the format of input. Further extension can be added by implementing a machine learning module that recognizes types of input and output and can use random test generation library to generate required testcases. This can be used as checker for code-synthesis paradigm. Thus, this project has a large scope of extensions and it can affect, help and revolutionise the way in which we perform software testing today.