

# **Software Code analyzing tool**

Project ID: 18-055

## **Software Requirements Specification**

**M.A.N.S.U.K. Uvindasiri IT15413802**

Bachelor of Science (Honors) Degree in Information Technology

Department of IT / SE / ISE

Sri Lanka Institute of Information Technology

May 2018

## **DECLARATION**

I hereby declare that the submitted project Software Requirements Specification document for Software Code analyzing tool is an original work done by M.A.N.S.U.K. Uvindasiri. This document is proprietary and an exclusive property of the SLIIT project group 18-055. List of references I referred for the preparation of this document are given as references at the end of the document.

Member M.A.N.S.U.K. Uvindasiri IT15413802

Signature: .....

## **ABSTRACT**

The main objective of this research is to develop a program analyzing tool in a manner that it allows its users to easily understand a given program. The tool is initially built to analyze OO programs. In addition, it provides ratings regarding the performance of each programmer/developer and suggestions to reduce the complexity of programs which are identified as complex. This tool could be used by anyone involved in programming such as students, teachers and lecturers, researchers and any personnel involved in the information technology (IT) industry such as developers, maintainers, project managers etc.

With the existing tools we can measure the quality of the source code and warnings/errors produced. Even, Most of the warnings are spurious and the developers are not paying attention to the output. This is why many teams require that code performance very clearly. Through the application the research will provide a wide range of users, from students to lecturers to industry personal, an efficient and convenient manner in which to measure the complexity of both small and large Java programs, allowing them to quickly and easily remove code bottlenecks thus reducing maintenance and testing phase costs and efforts significantly through the Software measuring tool will provide a wide range of users, from students to lecturers to industry personal, an efficient and convenient manner in which to measure the complexity of both small and large Java programs, allowing them to quickly and easily remove code bottlenecks thus reducing maintenance and testing phase costs and efforts significantly.

It's a major requirement to determine the capacity and performance of Software developers assigning to a project in terms of gaining the maximum productivity of projects in commercial scale. Hence we are focusing to develop a more interactive Software measuring tool to analyze the source code quality and the performance of Software developers assigned to a certain project.

**LIST OF TABLES**

Table 1.1 ..... 2

Table 2.1 ..... 16

Table 2.2 ..... 17

Table 2.3 ..... 17

Table 2.4 ..... 18

Table 4.1 ..... 28

## LIST OF FIGURES

Fig. 2 1 .....	4
Fig. 2 2 User Interface 1 .....	6
Fig. 2 3 User Interface 2 .....	7
Fig. 2 4 User Interface 3 .....	8
Fig. 2 5 User Interface 4 .....	9
Fig. 2 6 User Interface 5 .....	9
Fig. 2 7 User Interface 6 .....	10
Fig. 2 8 User Interface 7 .....	10
Fig. 2 9 User Interface 8 .....	11
Fig. 2 10 User Interface 9 .....	11
Fig. 2 11 XAMPP interface .....	12
Fig. 2 12 Software Interface 2.....	13
Fig. 2 13 Use case diagram of <i>ProAnalýza</i> .....	16
Fig. 3 1 User Interface 8 .....	21
Fig. 3 2 User interface 9.....	21
Fig. 3 3 User Interface 10 .....	22
Fig. 3 4 User Interface 11 .....	22
Fig. 3 5 Class diagram of <i>ProAnalýza</i> .....	24
Fig. 4 1 Comparision of existing static code analyzing tool with reference to the features that included in <i>ProAnalýza</i> .....	27
Fig. 4 2 Gantt Chart .....	29

## CONTENTS

DECLARATION .....	i
ABSTRACT .....	ii
LIST OF TABLES .....	iii
LIST OF FIGURES .....	iv
CONTENTS .....	v
1 INTRODUCTION .....	1
1.1 Purpose .....	1
1.2 Scope .....	1
1.3 Definitions, Acronyms, And Abbreviations .....	1
1.4 Overview .....	2
2 OVERALL DESCRIPTIONS .....	4
1.5 Product Perspective .....	5
1.5.1 System interfaces .....	6
1.5.2 User interfaces .....	6
1.5.3 Hardware interfaces .....	12
1.5.4 Software interfaces .....	12
1.5.5 Communication interfaces .....	13
1.5.6 Memory constraints .....	13
1.5.7 Operations .....	14
2.1.8 Site adaptation requirements .....	15
1.6 Product Functions .....	15
1.7 User characteristics .....	18
1.8 Constraints .....	19
1.9 Assumptions and dependencies .....	19
1.10 Apportioning of requirements .....	19
2 SPECIFIC REQUIREMENTS .....	20
3.1 External Interface Requirements .....	20
3.1.1 User interfaces .....	20
3.1.2 Hardware interfaces .....	23
3.1.3 Software interfaces .....	23
3.1.4 Communication interfaces .....	23
3.2 Classes/Objects .....	24
3.4 Design Constraints .....	25
3.5 Software System Attributes .....	25
3.5.2 Availability .....	25
3.5.3 Security .....	25
3.5.4 Maintainability .....	26
3.6 Other Requirements .....	26
4 SUPPORTING INFORMATION .....	27
4.1 Appendices .....	27
5 REFERENCES .....	30

# 1 INTRODUCTION

## 1.1 Purpose

This document provides the requirements for the “Software Code Analyzing Tool” (SCAT) which is the outcome of project group 18-055. It will be illustrating the purpose and complete declaration for the development of the system. It will also explain system constraints, interface, and interactions with other external applications. This document is primarily intended to be proposed to the project supervisor for his approval and a reference for developing the first version of the system for the development workflow.

## 1.2 Scope

This document covers the requirements for the initial version release of SCAT. Further, this document especially describes my (M.A.N.S.U.K. Uvindasiri IT15413802) development part such as Number of control structures, Number of conditions checked by a single control structure, The types of control structure, the nesting levels of control structures, Operand and operators based analysis.

## 1.3 Definitions, Acronyms, And Abbreviations

Term	Definition
SRS	Software Requirements Specification
Source code	A series of programming statements written in Java language
User	Someone who interacts with the proposed tool
Stakeholder	Any person who is interact with the system
ANLTR	Another Tool for Language Recognition
OOP	Object-Oriented Programming
GUI	Graphical User Interface
JVM	Java Virtual Machine
MySQL	My Structured Query Language.

HTML5	The fifth and current major version of Hyper Text Markup Language
PHP5	The fifth and current major version of Hypertext Preprocessor which is a server-side scripting language designed for web development
CSS3	The latest evolution of the Cascading Style Sheets language
AJAX	Asynchronous JavaScript and XML
Bootstrap	A free and open-source front-end library for designing websites and web applications
jQuery	A cross-platform JavaScript library designed to simplify the client-side scripting of HTML
TCP/IP	Transmission Control Protocol/Internet Protocol
RAM	Random Access Memory
developer	Software developer/Software engineer who develops a software
NLP	Natural Language Processing
MTBF	Mean Time between Failures
MTTF	Mean Time To Failures

**Table 1.**Error! No text of specified style in document.1

## 1.4 Overview

This SRS document intends to cover all the functional and non-functional requirements of the SCAT. This documentation is expected to be used in future development progress and as a snapshot to the supervisor to keep the evaluation process easier. This document includes five main sections and each of those sections covers up a different perspective.

Rest of the document will cover up the overall description, specific requirements and supporting information of the SCAT. Overall description includes product perspective, product functions, user characteristics, limitations, assumptions, and dependencies. Under the product perspective, system, user, and hardware interfaces are covered. Next section will be describing specific requirements.

Product requirements compared to performance and design are explained under specific requirements section. This section includes external interface requirements, entities,



performance requirements, design constraints, software system attributes and other requirements. Under external interface requirements, user, hardware, software and communication interfaces will be discussed. Under software system attributes, reliability, availability, security, and maintainability will be discussed. Finally supporting information section and the references section are given. The appendix will include a detailed class diagram of the system and other supporting information. This system was proposed to overcome some of the miss conveniences of existing code analyzers.

There are many code analyzing tools which have the capability of analyzing the code both static and dynamic ways. SCAT was initially proposed with two main goals as given below.

- i. To create a comprehensive tool to analyze and increase the maintainability of the software, measure and effectively display complexity of code using complexity metrics, clearly identifying complex code segments and plan the productivity and maintainability of the software product .
- ii. To create a comprehensive tool that will check the quality of a program to industry or company specific quality standards. The standards will be selectable by the user through an interactive interface and developers rank according there productivity.

## 2 OVERALL DESCRIPTIONS

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

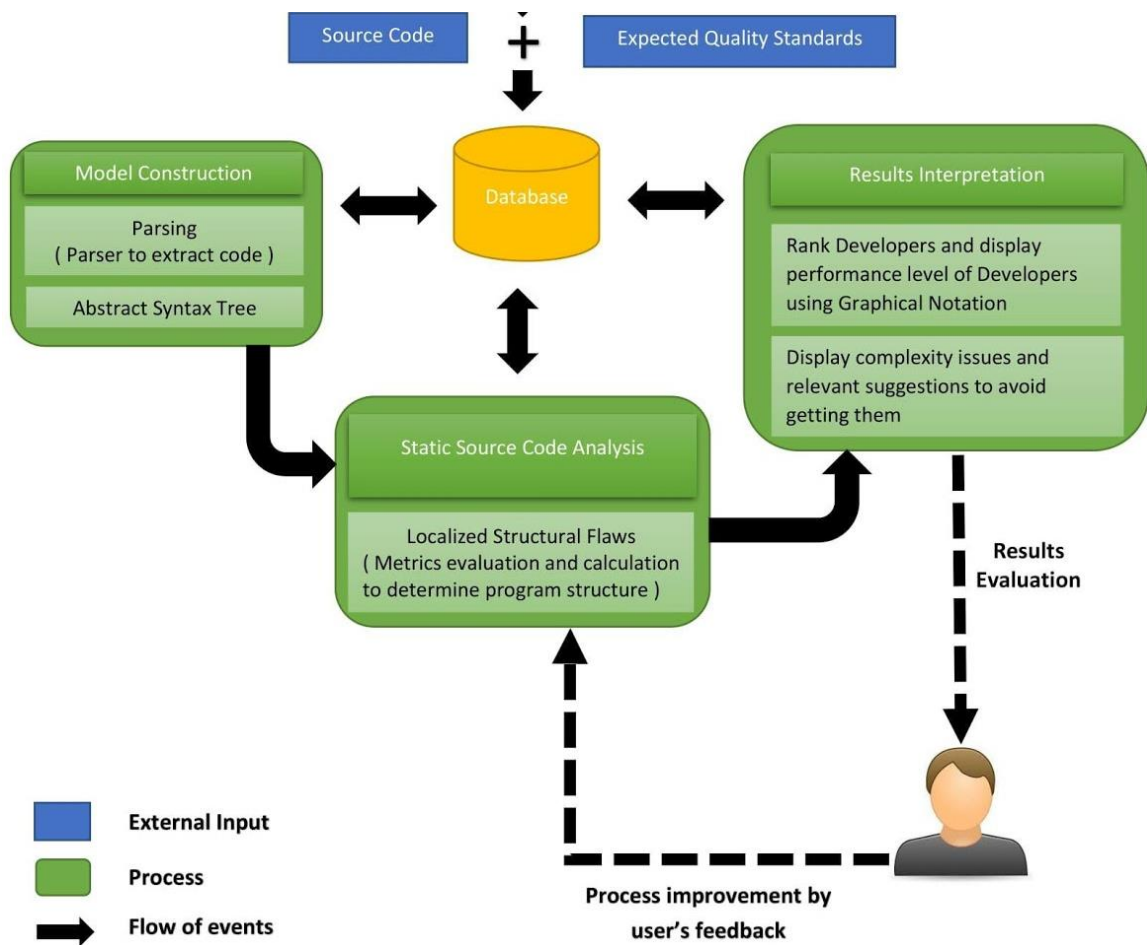


Fig. 2 1

## 1.5 Product Perspective

This system will consist of three main components: model construction, static source code analysis and results interpretation.

When user input the source code and expected quality standards, model construction will be initiated. There parsing the source code in which character stream of the source code breaks up into constituent pieces and imposes a grammatical structure on them will be take place. In this approach, NLP is use for code parsing. Rather than developing a parser from scratch with the use of NLP, one of the existing parsing tool, ANLTR would be used in integration.

After parsing the code and generating an abstract syntax tree, static source code analysis is carry out by evaluation of complexity metrics based on the program structure. We are focusing on following major factors of an OOP concepts in complexity metrics evaluation.

1. Coupling
2. Exception handling
3. Inheritance
4. Control Structures

Based on these main factors, in *ProAnalýza*, the following criteria will be taken into consideration as my (M.A.N.S.U.K. Uvindasiri IT15413802) development part.

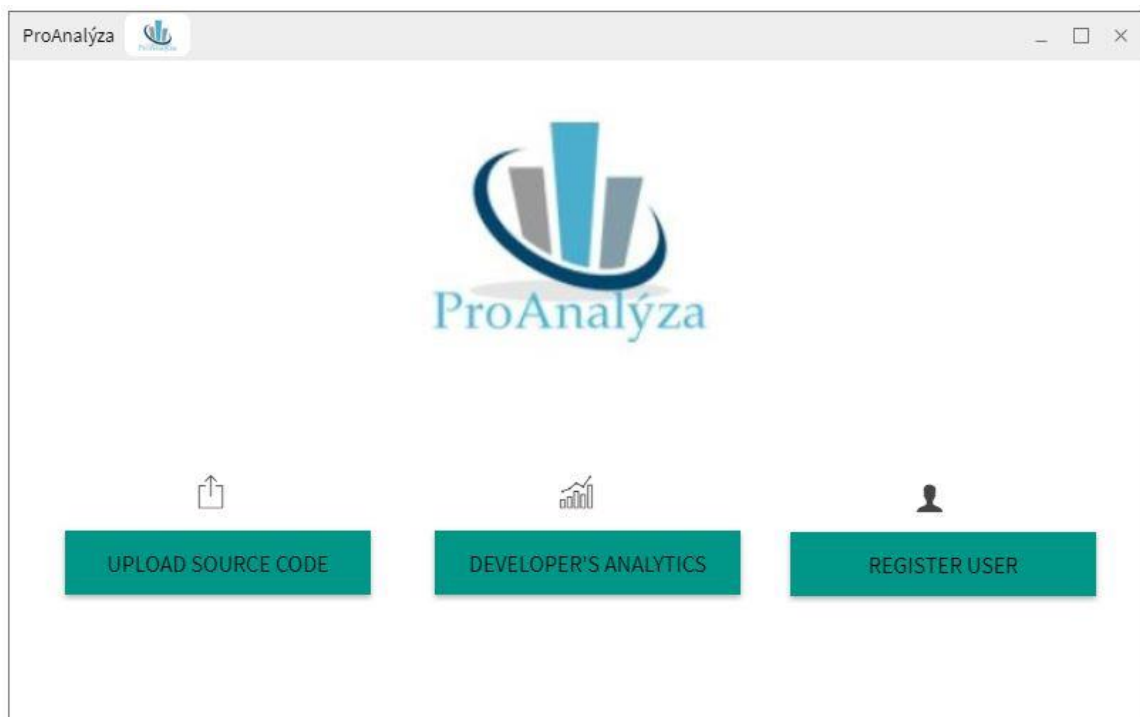
- Number of control structures.
- Number of conditions checked by a single control structure.
- The types of control structure.
- The nesting levels of control structures. (iterations in iterative control structure)
- Operand and operators.

### 1.5.1 System interfaces

Windows 10 operating system will be used as the development platform. One of the major technology that we use in software invocation is based on Java language. The linking between the system and the Operating system is handled by the Microsoft Windows environment and JVM.

### 1.5.2 User interfaces

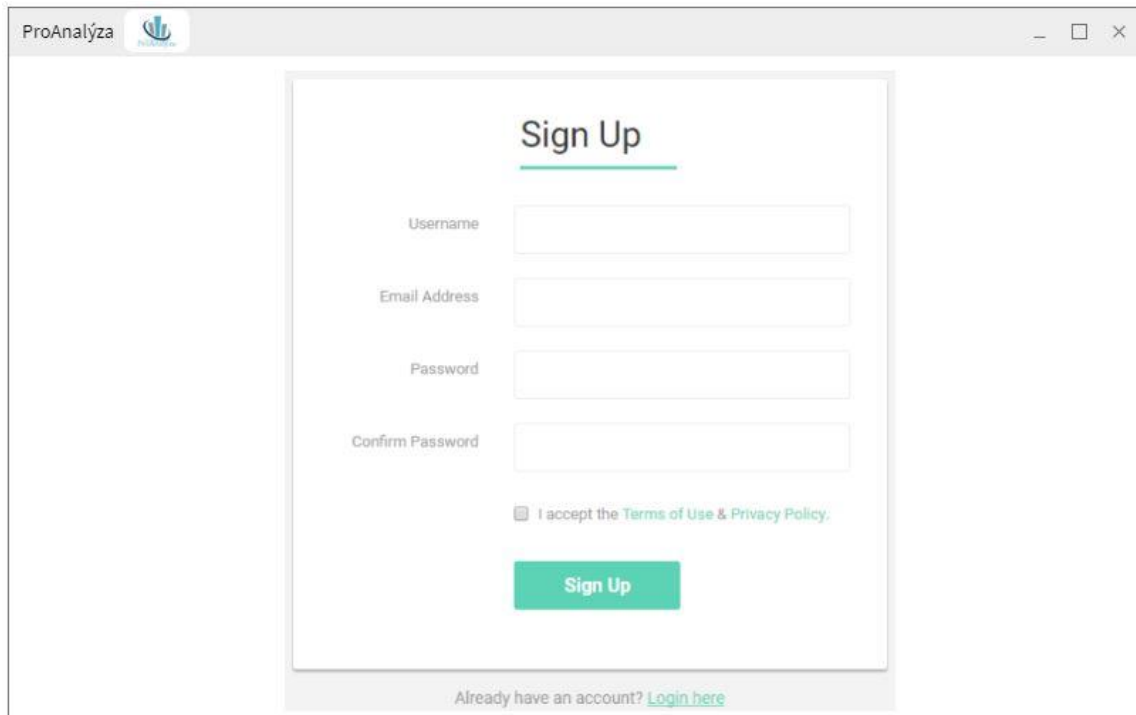
The following is the main GUI of *ProAnalýza* where user can gain access to all functionalities available in it.



**Fig. 2 2 User Interface 1**

The screenshots of the GUI for the core functionalities of *ProAnalýza* are as follows.

## 1. User Registration



The screenshot shows a web browser window titled "ProAnalýza" with a logo. The main content is a "Sign Up" form. The form has the following fields and elements:

- Sign Up**: The title of the form, underlined.
- Username**: A text input field.
- Email Address**: A text input field.
- Password**: A text input field.
- Confirm Password**: A text input field.
- ☐ I accept the [Terms of Use & Privacy Policy](#).
- Sign Up**: A green button to submit the form.
- [Already have an account? Login here](#): A link at the bottom of the form.

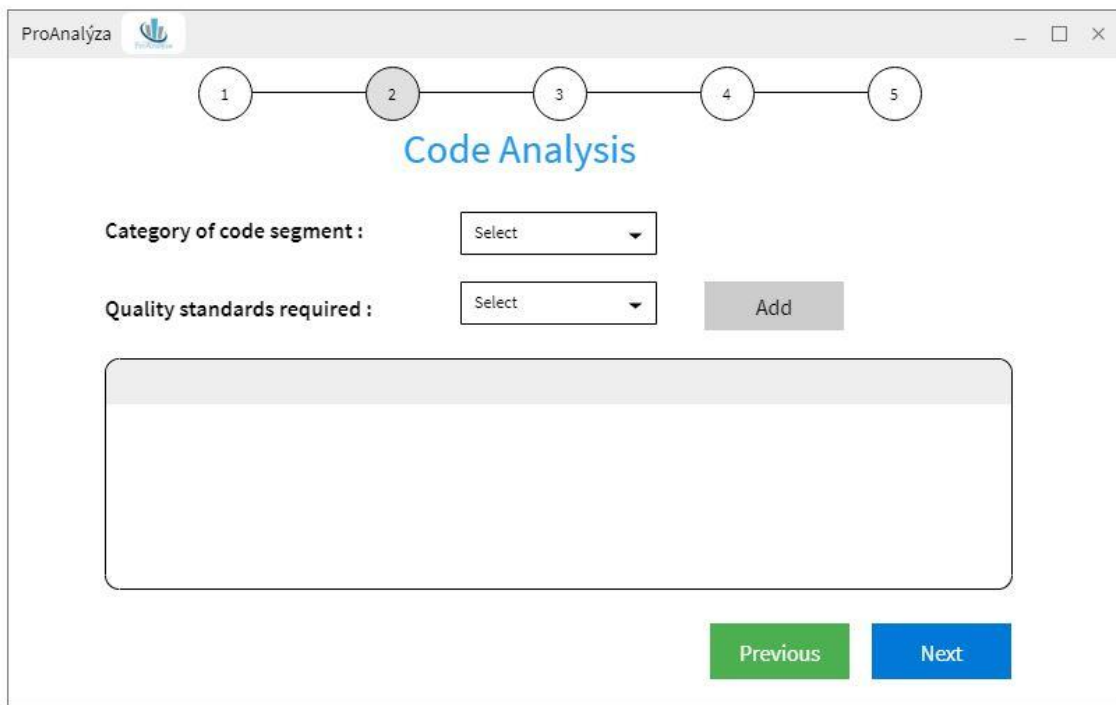
**Fig. 2 3 User Interface 2**

## 2. Upload source code

The screenshot shows a web application window titled "ProAnalýza". At the top, there is a progress bar with five numbered circles (1 to 5). Below the progress bar, the title "Upload Source Code" is displayed in blue. The form contains two input fields: "User ID of Developer :" and "Project Name :". Below these fields is a large dashed rectangular area for file upload, with a blue arrow pointing down and the text "You can drag and drop files here to add them." Below the upload area are two buttons: "Save changes" and "Cancel". At the bottom right, there is a blue "Next" button.

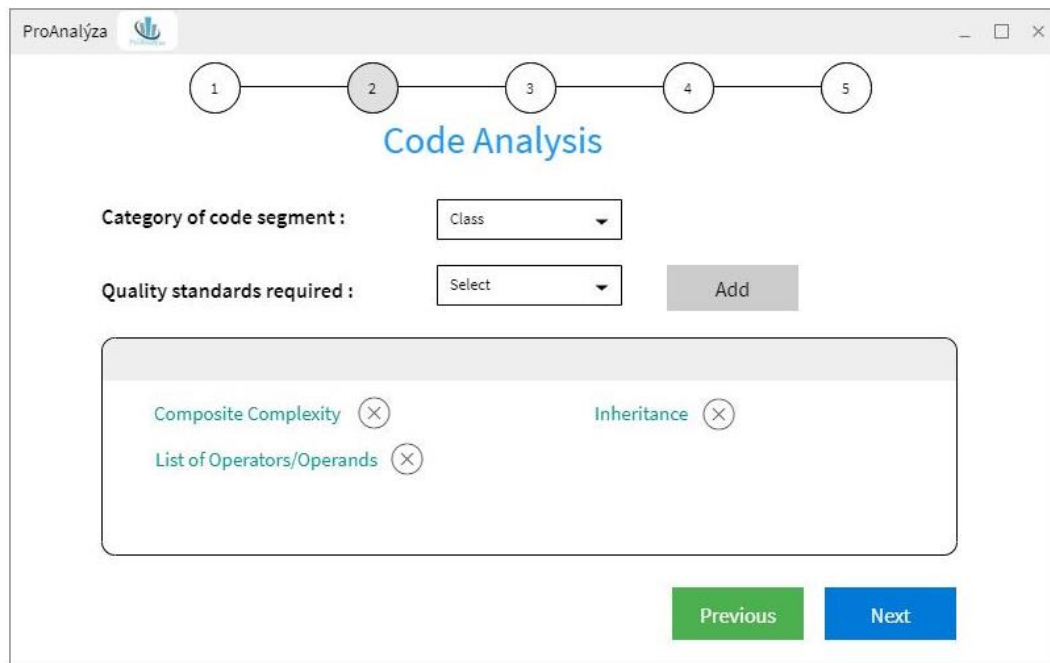
**Fig. 2 4 User Interface 3**

### 3. Input expected quality standards



The screenshot shows a web application window titled "ProAnalýza". At the top, there is a progress bar with five numbered circles (1-5). Circle 2 is highlighted, indicating the current step. Below the progress bar, the title "Code Analysis" is displayed in blue. The main form area contains two dropdown menus: "Category of code segment :" with a "Select" option, and "Quality standards required :" with a "Select" option. To the right of the second dropdown is a grey "Add" button. Below these elements is a large, empty rectangular box for listing standards. At the bottom right, there are two buttons: a green "Previous" button and a blue "Next" button.

**Fig. 2 5 User Interface 4**



The screenshot shows the same web application window as Fig. 2 5, but now the "Quality standards required" dropdown is populated with three items: "Composite Complexity", "List of Operators/Operands", and "Inheritance". Each item has a small "X" icon next to it, indicating it can be removed. The "Add" button remains grey. The "Previous" and "Next" buttons are still present at the bottom right.

**Fig. 2 6 User Interface 5**

4. Tabular representation of complexity metrics evaluation of source code

ProAnalýza

1 — 2 — 3 — 4 — 5

### Composite Complexity

Method Name	Statement Number	Executable statement	List of operators	List of operands	Size (S)	Weight due to nesting level of control structures (Wn)	Weight due to inheritance level of statements (Wi)	Weight due to type of control structures (Wc)	Total weight (Wt)	S x Wt	Complexity of a method
										5	
										6	
										7	
Complexity of Method A											18
										4	
										5	
										9	
										2	
Complexity of Method B											20
Complexity of class											38

Previous Next

Fig. 2 7 User Interface 6

5. Provide user feedback to determine developer's performance rate

ProAnalýza

1 — 2 — 3 — 4 — 5

### User Feedback

User ID of Software Developer : SE129

1. Issue/requirement/customer understanding :

2. Transparency,updating,updating status reporting :

3. Project performance improvements & upgrades :

Previous Next

Fig. 2 8 User Interface 7



## 6. Performance evaluation of the developer

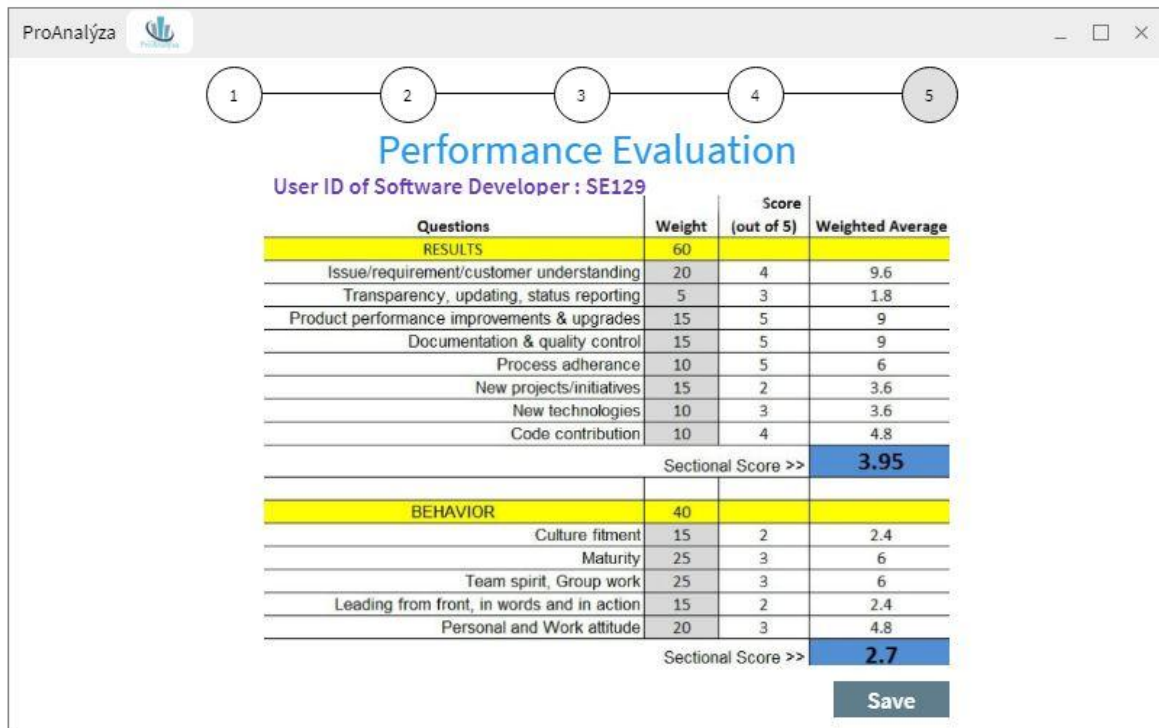


Fig. 2 9 User Interface 8

## 7. View developer's performance analytics

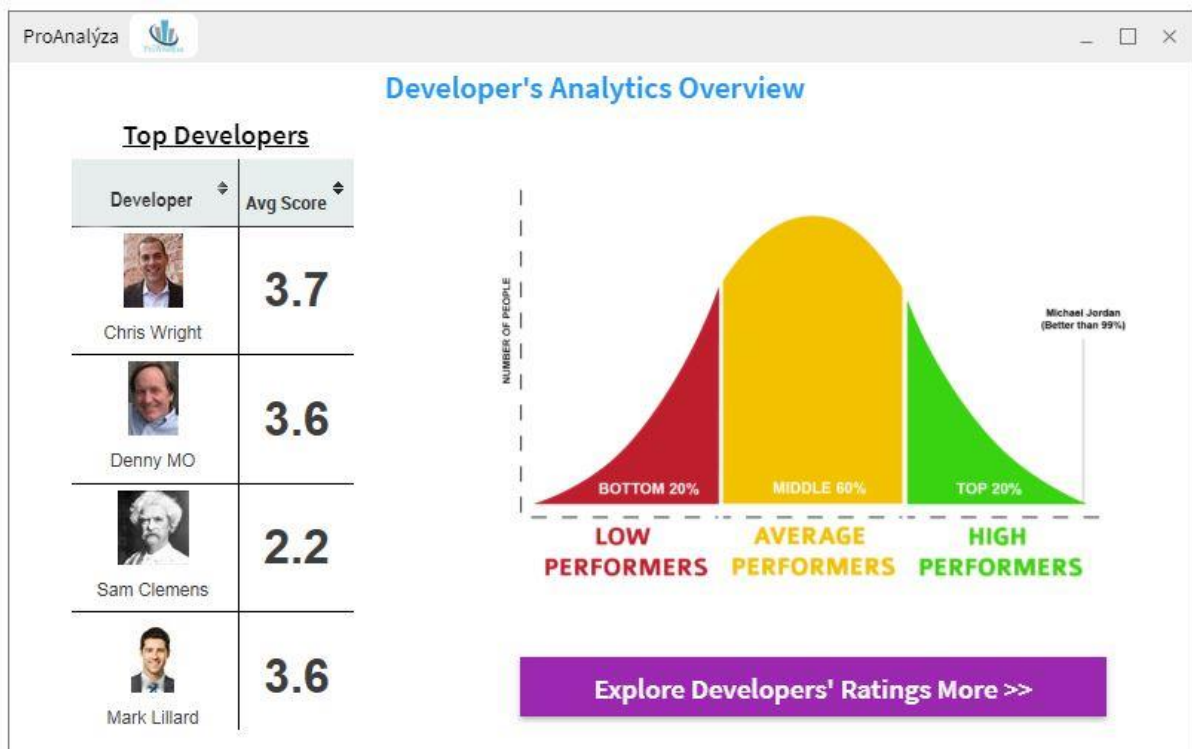


Fig. 2 10 User Interface 9

### 1.5.3 Hardware interfaces

No specific hardware interface is required apart from a standard PC/Laptop with minimum requirement of Intel Pentium IV with at least 1.6GHz processor or above and hardware devices such as USB dongle/ADSL router to connect to the internet.

### 1.5.4 Software interfaces

Following software interfaces are required to use *ProAnalyza*.

- In our approach, XAMPP is used to connect to MySQL database and to interpret scripts written in PHP5 programming.

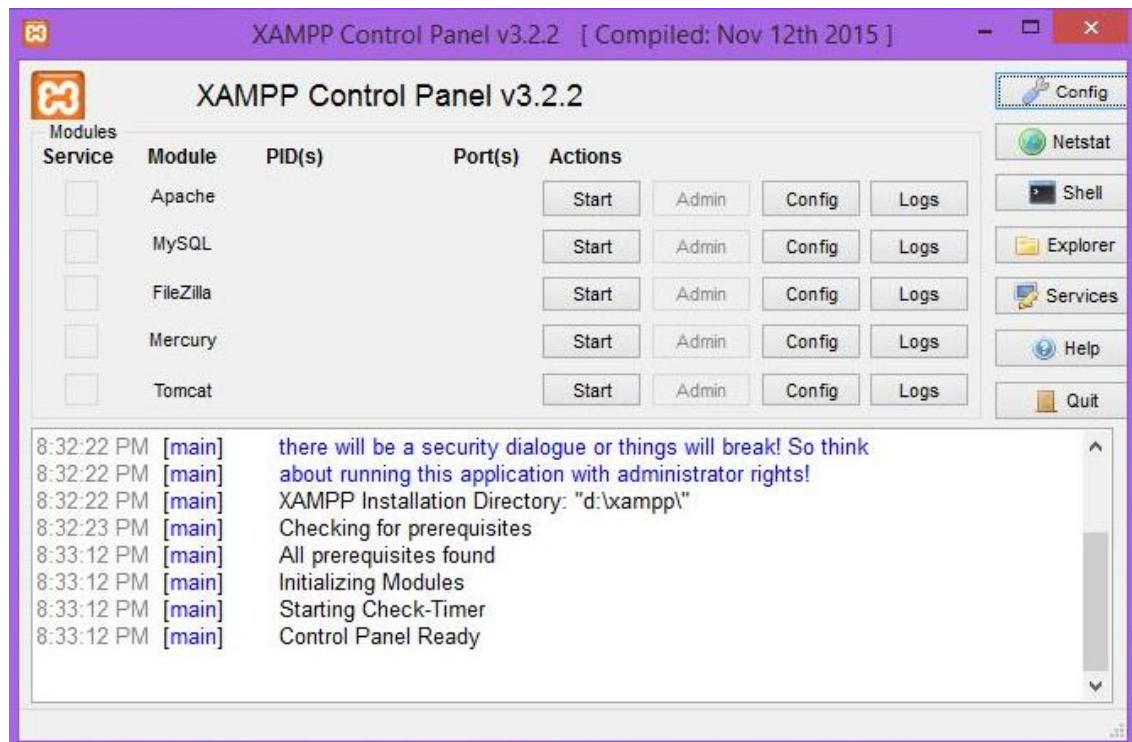
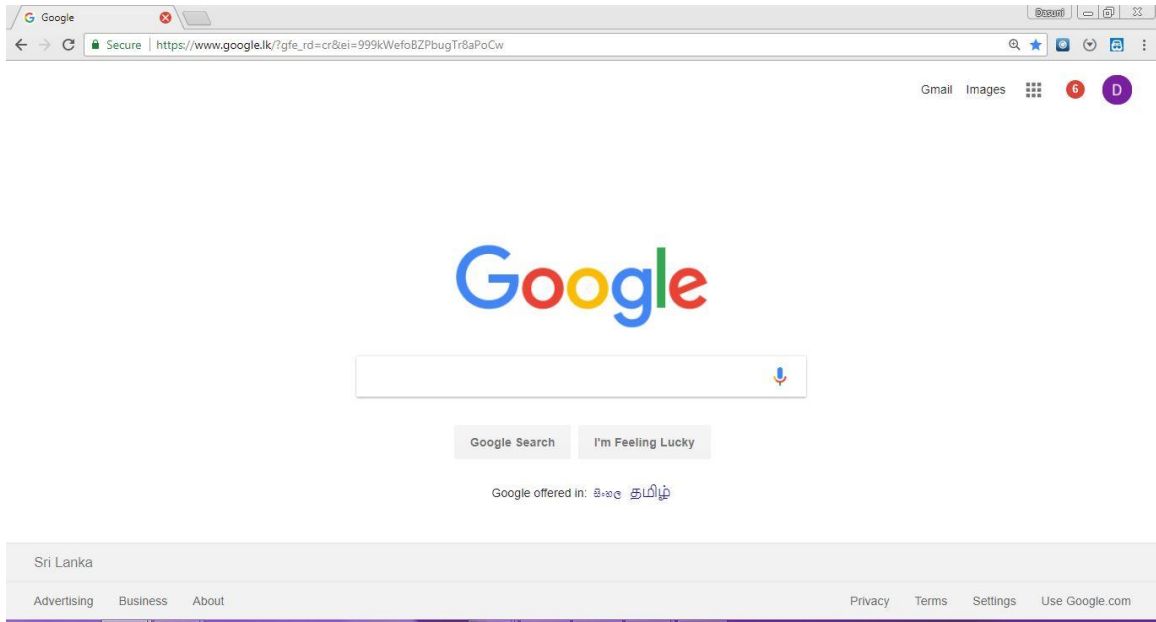


Fig. 2 11 XAMPP interface

- Latest updated web browser that supports JQuery, AJAX, HTML5, PHP5, CSS3 and Bootstrap technologies.

Eg: Internet Explorer version 5.0 or above, Mozilla Firefox version 3.5 or above, Google Chrome etc.



**Fig. 2 12 Software Interface 2**

### **1.5.5 Communication interfaces**

Following communication interfaces are required to use *ProAnalyza*.

- Internet connection with fairly high bandwidth will be required to run the system efficient since it deals with large about of data. It will use the HTTP protocol for communication over the internet and for the intranet communication will be through TCP/IP protocol suite.
- Database connection interface will be required to connect with the database.

### **1.5.6 Memory constraints**

The device with *ProAnalyza* installed should have at least RAM of 2GB or higher to run the application.

### 1.5.7 Operations

The following are the main operations that will be available to a user of the *ProAnalýza*.

1. User registration

Prior to user login, the user must provide relevant information and get register to the system.

2. Upload source code

When the user successfully login to the system, the user will be able to upload source code through an interface that will allow him/her to ingest their code to the tool.

3. Input expected quality standards

After uploading the source code, the user will first be prompted with an interface where he/she can input the quality standard expected to measure and evaluate source code with respect to complexity metrics.

4. Tabular representation of complexity metrics evaluation of source code

Results after analyzing the source code complexity with respect to the user expected quality standards will be represented in the form of tables.

5. Provide user feedback to determine developer's performance rate

To determine performance of a developer, user feedback must be provided in addition to the complexity metrics evaluation.

6. Performance evaluation of the developer

With user feedback and complexity metrics evaluation of source code, each developer will be rank and through an interface developer's performance analysis will be displayed.

### **2.1.8 Site adaptation requirements**

Following site adaptation requirements are identified regarding the implementation of *ProAnalýza*.

- The server must have MySQL installed on it.
- Server machine must be running Apache server to deploy the web application.
- The user machine should have Java Virtual Machine installed.

### **1.6 Product Functions**

This section includes the requirements that specify all the fundamental functions of the *ProAnalýza*.

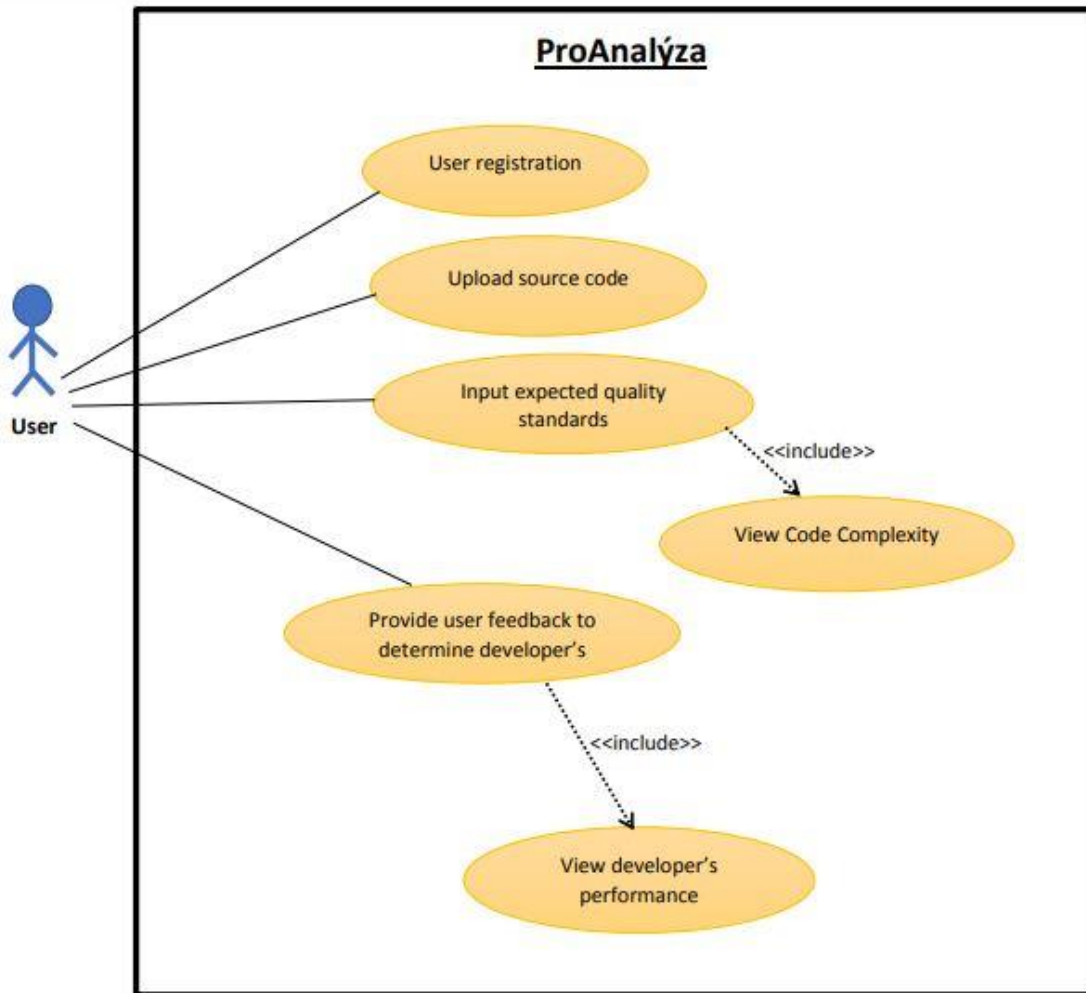


Fig. 2 13 Use case diagram of *ProAnalýza*

Table Error! No text of specified style in document..1 Use Case Scenario 1

<b>Use Case ID:</b>	1
<b>Use Case Name:</b>	User registration
<b>Actors:</b>	User
<b>Pre-conditions:</b>	Program successfully launched Database connection is live User can view the main interface of <i>ProAnalýza</i>
<b>Post-conditions:</b>	User registration successful
<b>Flow of events:</b>	1. Navigate to the user registration menu 2. Enter relevant information in the registration form 3. Then press “Sign Up” button
<b>Alternative flows:</b>	2a. Upon validation of user information, if there’s an error or wrong format of details being entered by the user, he/she will be prompt a notification by the system.

<b>Exceptions:</b>	None

**Table 2.2 Use Case Scenario 2**

<b>Use Case ID:</b>	2
<b>Use Case Name:</b>	Upload source code
<b>Actors:</b>	User
<b>Preconditions:</b>	Program successfully launched Database connection is live User can see the upload interface
<b>Post-conditions:</b>	Program code successfully uploaded
<b>Normal Flow:</b>	1. Click Browse button 2. Select the source code folder 3. Click upload button
<b>Alternative Flows:</b>	1a. Drag and drop program code folder into upload area
<b>Exceptions:</b>	None

**Table Error! No text of specified style in document..3 Use Case Scenario 3**

<b>Use Case ID:</b>	3
<b>Use Case Name:</b>	Input expected quality standards
<b>Actors:</b>	User
<b>Preconditions:</b>	Program code successfully uploaded Database connection is live User can see the quality standards interface
<b>Post-conditions:</b>	User successfully submitted quality standards preference and user can see the table representation of complexity metrics evaluation of source code in an interface according to the control structure types and their nesting levels.
<b>Normal Flow:</b>	1. Select desired metric from the combo box 2. Click "Add button" 3. Click "Next" button
<b>Alternative Flows:</b>	2a. Click "Previous" button to upload the source code
<b>Exceptions:</b>	None

**Table 2.4 Use Case Scenario 4**

<b>Use Case ID:</b>	4
<b>Use Case Name:</b>	Provide user feedback to determine developer's performance rate
<b>Actors:</b>	User
<b>Preconditions:</b>	Database connection is live User can see the user feedback interface
<b>Post-conditions:</b>	User successfully submitted user feedback about the developer's performance and user able to see the ratings of developer's performance in an interface
<b>Normal Flow:</b>	1. Fill the relevant information in the form 2. Click "Next" button
<b>Alternative Flows:</b>	2a. Click "Previous" button to see complexity evaluation again to provide user feedback
<b>Exceptions:</b>	None

### 1.7 User characteristics

Basic knowledge of using computers and Java programming is adequate to use this application. Knowledge of how to use a mouse or keyboard and internet browser is necessary. The user interface will be friendly enough to guide the user.

There will only be one user (this can range from an industry expert, to a student, to a lecturer/teacher or industry employee) that will have the same level of privileges and will have access to the complete functionality of the tool.



## 1.8 Constraints

- Access to the web is required (Hence an internet connection with fairly high bandwidth is necessary).
- All the tools and technologies used for the development should be open source. Since we could accommodate a low budget for the implementation.
- *ProAnalýza* shall operate on PCs running Windows 95 or later at a minimum speed of 100 MHZ.
- Java, HTML5, PHP5, CSS3, Bootstrap, NLP shall be the implementation technologies.

## 1.9 Assumptions and dependencies

It is assumed that the user is familiar with an internet browser and also familiar with handling the keyboard and mouse (Users should have basic knowledge using a computer).

Since the application is a web application there is a need for the internet browser. It will be assumed that the users will possess decent internet connectivity.

Server should be up and running 24x7 hours.

There should be sufficient memory and processing power to run the system (as specified in 2.1.6)

## 1.10 Apportioning of requirements

The requirements described in sections 1 and 2 of this document are referred to as primary specifications; those in section 3 are referred to as requirements (or functional) specifications. The two levels of requirements are intended to be

consistent. Inconsistencies are to be logged as defects. In the event that a requirement is stated within both primary and functional specifications, the component will be built from functional specification since it is more detailed.

## **2 SPECIFIC REQUIREMENTS**

This section includes specific requirements related to analyzing source code based on number of control structures, number of conditions checked by a single control structure, the types of control structure, the nesting levels of control structures, Operand and operators which are components of complexity measuring module.

### **3.1 External Interface Requirements**

#### **3.1.1 User interfaces**

The basic GUIs of my (*M.A.N.S.U.K.Uvindasiri*) parts on this project are shown in the section 2.1.2.

*All the GUIs that will indicate the results of evaluation and measure of complexity of a Java program are as follows according to* Number of control structures, number of conditions checked by a single control structure, the types of control structure, the nesting levels of control structures, Operand and operators based analysis.

- List of operators

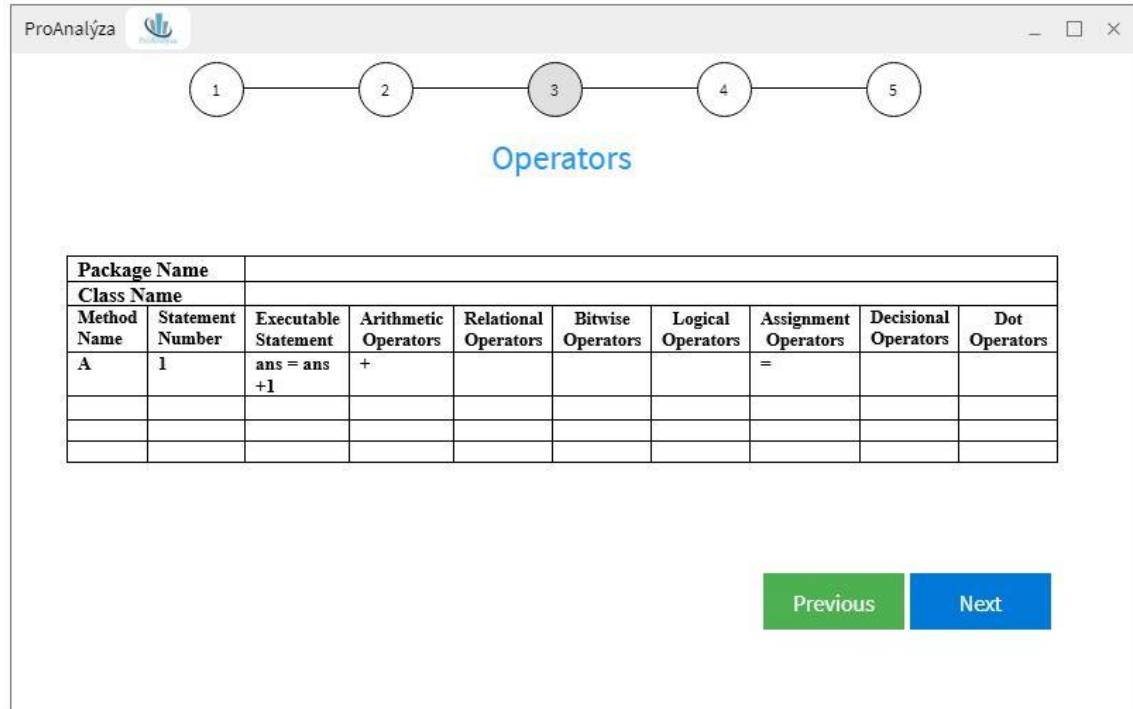
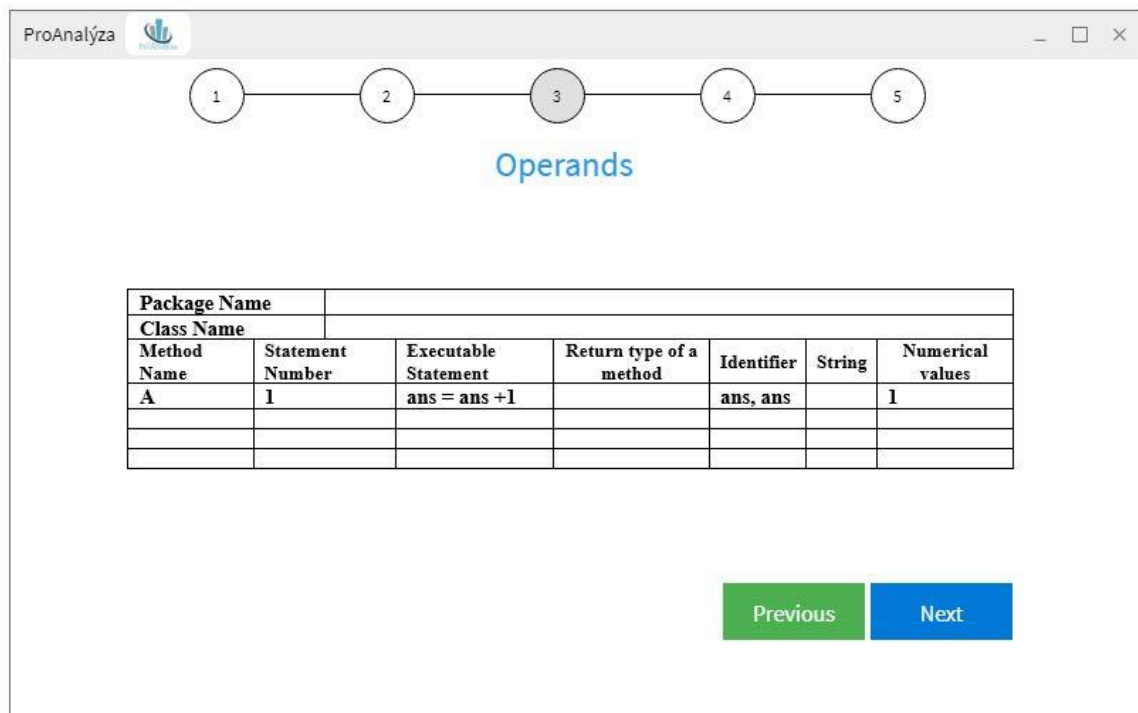


Fig. 3 1 User Interface 8

- List of operands



ProAnalýza

1

2

3

4

5

### Type and nesting levels

Package Name												
Class Name												
Method Name	Statement Number	Executable Statement	If		For		While		Do-while		Switch - case	Nesting level
			Count	No of conditions	Count	No of conditions	Count	No of conditions	Count	No of conditions		

Previous

Next

Fig. 3 3 User Interface 10

- Decisional statements

ProAnalýza

1

2

3

4

5

### Decisional statements

Package Name	ABCDE							
Class Name	Method Count	Method Name	If	For	While	Do-while	Switch - case	Total
EEE	1	AA	1	2	0	0	0	3
EEE	2	BA	0	1	1	0	2	4
EEE	3	CA	0	2	0	1	1	4
Total decisional statements of class EEE								11
DDD	1	AA	1	2	0	0	0	3
DDD	2	BA	0	0	1	0	1	2
DDD	3	CA	0	1	0	1	1	3
DDD	3	DA	1	1	1	1	1	5
Total decisional statements of class DDD								13
Total decisional statements of package ABCDE								24

Previous

Next

### **3.1.2 Hardware interfaces**

As mentioned in section 2.1.3, no specific hardware interface is required for the analysis of a program by complexity measuring and evaluation apart from a standard PC/Laptop with minimum requirement of Intel Pentium IV with at least 1.6GHz processor or above and hardware devices such as USB dongle/ADSL router to connect to the internet.

### **3.1.3 Software interfaces**

XAMPP is used to connect to MySQL database and to interpret scripts written in PHP5 programming and a latest updated web browser that supports jQuery, AJAX, HTML5, PHP5, CSS3 and Bootstrap technologies.

A precise explanation is discussed in section 2.1.3.

### **3.1.4 Communication interfaces**

All communication interfaces are required to use *ProAnalýza* are discussed in section 2.1.4. Accordingly, an internet connection with high bandwidth and a database connection interface will be required to connect with the database.

### 3.2 Classes/Objects

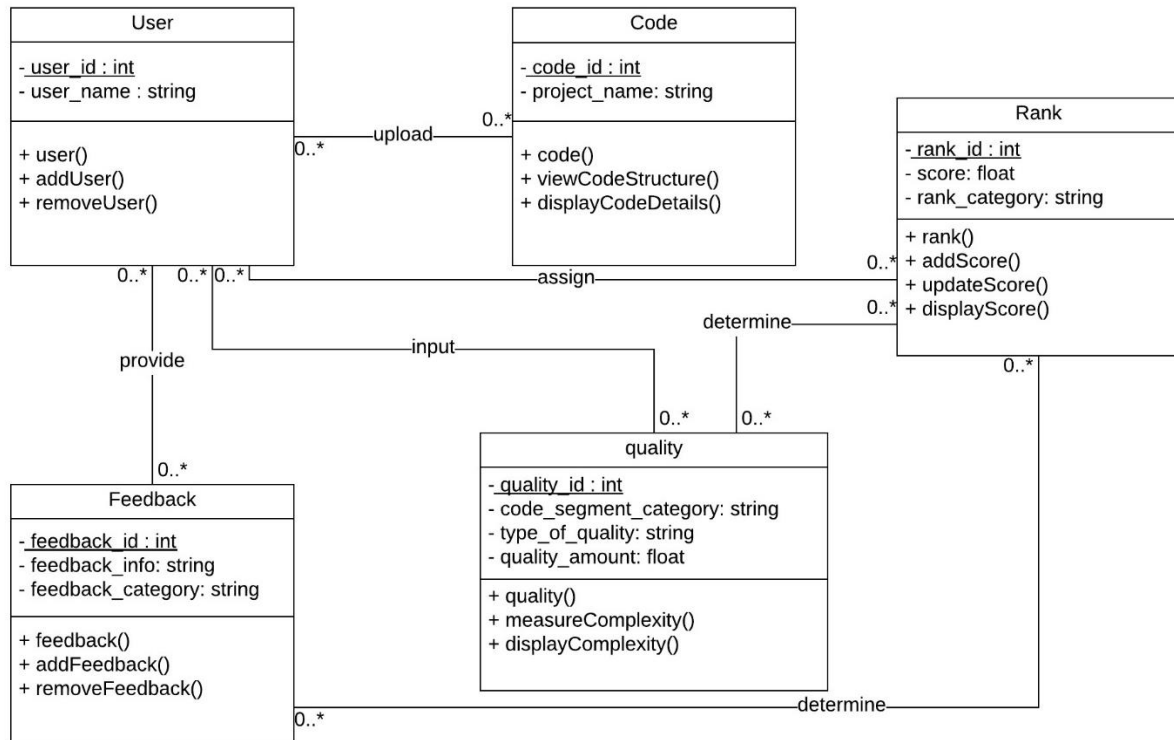


Fig. 3 5 Class diagram of *ProAnalýza*

### 3.3 Performance Requirements

Since this software is going to web – based, it does require a powerful server machine with high band internet access. Server machine should have a powerful CPU and high-speed internet access so that it can handle multiple users at the same time. Another performance requirement is the storage space. Higher storage space means more user and bigger workspace per user so higher the storage, better the performance. Performance requirement by the user side is, web application should be developed as a lightweight web app so that it can work on almost any platform even with slower internet connections. System should be able to deal with multiple users at the same time.

### 3.4 Design Constraints

- The system shall be built using open source web development software/s that conforms to Microsoft's GUI standards.
- The computers must be equipped with web browsers such as Internet explorer.
- The product must be stored in such a way that allows the client easy access to it.
- Response time for loading the product should take no longer than five minutes.
- A general knowledge of basic computer skills is required to use the product
- Data should not become corrupted in case of system crash or power failure (fault tolerance).

### 3.5 Software System Attributes

#### 3.5.1 Reliability

There are no requirements for MTBF or MTTF for this version of the *ProAnalýza* defined in this SRS. However, in accordance with industry recommended practices, the software system should undergo feature testing, load testing, and regression testing prior to release and/or deployment.

#### 3.5.2 Availability

Reasonable efforts should be made to ensure the *ProAnalýza* is available with an uptime of 95%. The uptime is calculated as the percentage of time during the year in which the software system was available to the public. A 95% uptime percentage allows for an average of 18.25 days per year, 36 hours per month, or 8.4 hours per week of downtime.

#### 3.5.3 Security

*ProAnalýza* must follow industry recommended practices for secure software development. At a minimum, the software development must practice the principle of least privilege for defining access-level requirements of the software system and its

associated services. The production-release version of the *ProAnalýza* must pass an automated dynamic application security testing tool.

### **3.5.4 Maintainability**

The architecture, design, implementation, and documentation of the application must minimize the maintenance costs of the software system. The maximum person-time required to fix a security defect (including regression testing and documentation update) must not exceed two person days. Otherwise, the software system must be taken offline or the offending feature disabled. The average person-time required to make a minor enhancement (including testing and documentation update) must not exceed one person week.

### **3.6 Other Requirements**

To improve portability, software should run on variety of platforms and variety of connection speeds. As explained in the performance requirements section, software should be lightweight so that it can run on a machine with slow internet connection. To make the web application lightweight, simple libraries and tools should be used at developing phase.


Portability of *ProAnalýza* can be gain by running on most number of different platform without an additional effort. To achieve this, web application should be developed by using the common technologies and tools which are provided by all common web browsers and operating system such as HTML5, JQuery etc.

*ProAnalýza* must guide users through an interface based on end user concepts. It must be easy to learn and does not obstruct the thematic understanding of the users and must make it easy to correct mistakes. It must be designed so that it can migrate to upgraded hardware or new versions of the operating systems involved. In addition, it must provide solutions/rules regarding data encoding problems such as supporting different character sets, name truncation rules, name matching in case of misspelling etc.



## 4 SUPPORTING INFORMATION

### 4.1 Appendices

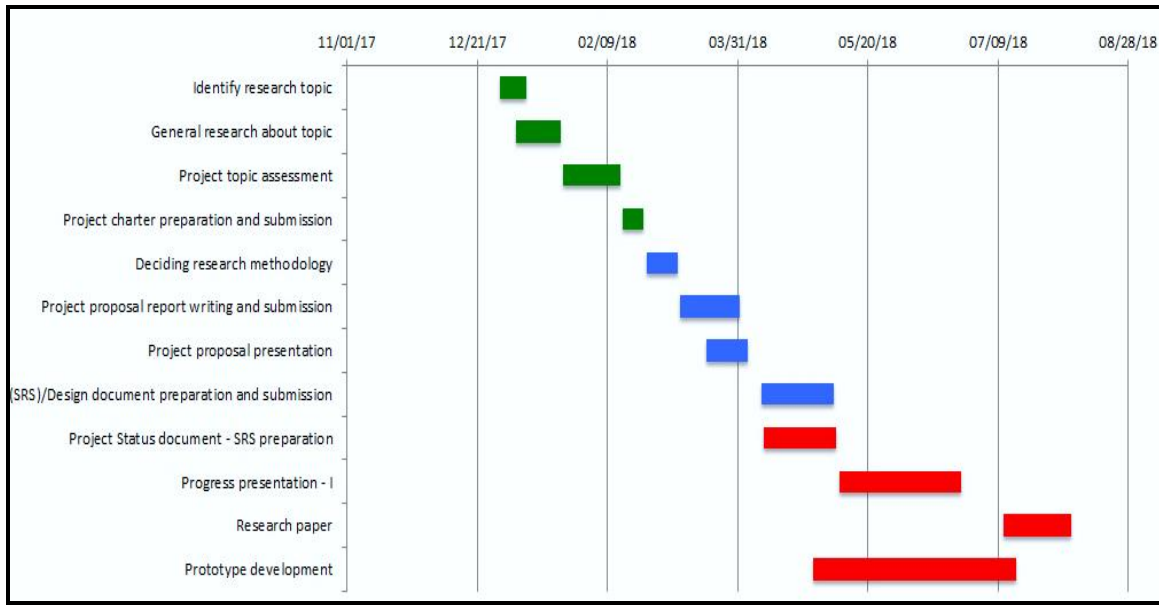


Low Memory Usage	★		★	
Static and Dynamic code analysis	★	★		
Affordable			★	
Accurate				★
Comprehensive rules				★
Quality checking	★	★		★
View program structure		★		
Visual Representation	★	★		★
Rate software developers based on their code quality				

Fig. 4 1 Comparision of existing static code analyzing tool with reference to the features that included in ProAnalýza

**Table Error! No text of specified style in document..1: Tabular representation of Gantt Chart**

Task Name	Start	Finish	Duration (days)
Forming Group	12/22/17	12/29/17	7
Identify research topic	12/30/17	01/09/18	10
General research about topic	01/05/18	01/22/18	17
Project topic assessment	01/23/18	02/14/18	22
Project charter preparation and submission	02/15/18	02/23/18	8
Deciding research methodology	02/24/18	03/08/18	12
Project proposal report writing and submission	03/09/18	04/01/18	23
Project proposal presentation	03/19/18	04/04/18	16
(SRS)/Design document preparation and submission	04/09/18	05/14/18	35
Project Status document - SRS preparation	04/10/18	05/14/18	34
Progress presentation – I	05/15/18	06/25/18	41
Research paper	07/11/18	08/06/18	26
Prototype development	04/29/18	07/16/18	78
Progress presentation – II	07/03/18	08/07/18	35
Final report (Draft)	08/20/18	09/01/18	12
Final report (Draft) feedback submission	09/05/18	09/20/18	15
Website Assessment	09/18/18	10/04/18	16
Final Report (Soft Bound)	09/12/18	10/06/18	24
Final Submissions – CD with deliverables	10/03/18	10/28/18	25
Final Presentation	10/23/18	11/12/18	20
Final Report (Group) Hard Bound	10/30/18	11/21/18	22



**Fig. 4 2 Gantt Chart**

## 5 REFERENCES

- [1] Zitser, M., Lippmann, R. and Leek, T., 2004, October. Testing static analysis tools using exploitable buffer overflows from open source code. In *ACM SIGSOFT Software Engineering Notes* (Vol. 29, No. 6, pp. 97-106). ACM.
- [2] Binkley, D., 2007, May. Source code analysis: A road map. In *2007 Future of Software Engineering* (pp. 104-119). IEEE Computer Society.
- [3] Chowdhury, G.G., 2003. Natural language processing. *Annual review of information science and technology*, 37(1), pp.51-89.
- [4] Witten, I.H., Frank, E., Hall, M.A. and Pal, C.J., 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.
- [5] Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledebor, G., Reynolds, P., Sitaram, P. and Ta, A., 1993, May. Identifying and measuring quality in a software requirements specification. In *Software Metrics Symposium, 1993. Proceedings., First International* (pp. 141-152). IEEE.
- [6] Mekprasertvit, C., 2003. *Software Requirements Specification* (Doctoral dissertation, Kansas State University).
- [7] Voutilainen, A., 1995, March. A syntax-based part-of-speech analyser. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics* (pp. 157-164). Morgan Kaufmann Publishers Inc..
- [8] Neamtiu, I., Foster, J.S. and Hicks, M., 2005. Understanding source code evolution using abstract syntax tree matching. *ACM SIGSOFT Software Engineering Notes*, 30(4), pp.1-5

- [9]Zuse, H., 1991. Software complexity. *NY, USA: Walter de Gruyter*
- [10]Curtis, B., Sheppard, S.B., Milliman, P., Borst, M.A. and Love, T., 1979. Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics. *IEEE Transactions on software engineering*, (2), pp.96-104
- [11]Vanderfeesten, I., Cardoso, J., Mendling, J., Reijers, H.A. and van der Aalst, W., 2007. Quality metrics for business process models. *BPM and Workflow handbook*, 144, pp.179-190

