MINISTRY OF EDUCATION
AND TRAINING VIET NAM

FPT UNIVERSITY

Capstone Project Document

Tobi Automation Testing Tool

TOBI

Tobi Project		
Project members	Len Van Hau	SE04158
	Phan Anh Tu	SE03994
	Ngo Ngoc Lam	SE04185
	Nguyen Thi Giang	SE04376
	Nguyen Dinh Dong	SE04430
Supervisor	**Bui Ngoc Anh MSc.**	
Capstone Project code	**TOBI**	

Hanoi, April 2019

Table of Contents

DEFINITIONS AND ACRONYMS 6
CHAPTER 1 – INTRODUCTION 7
1.1. Purpose 7
1.2. Project Information 7
1.3. Project Members 7
1.3.1. Supervisor 7
1.3.2. Team Members 7
1.4. Background 8
1.5. Problem 8
1.6. Existing systems 9
1.7. Idea of Product 15
1.8. Proposal of System 15
1.8.1. Basic functions 17
1.8.2. Out of Scope Functions 18
CHAPTER 2 – SOFTWARE PROJECT MANAGEMENT PLAN 19
2.1. Project Organization 19
2.1.1 Purpose 19
2.1.2 Software Process Model 19
2.1.3 Organization Structure 20
2.1.4 Roles and Responsibilities 21
2.1.5 Project Team Member 22
2.1.6 Tools and Techniques 22
2.2. Project Management Plan 23
2.3.1. Tasks Schedule 23
2.3.2. Project Milestones 27
2.3.3. Meeting Minutes 27
2.3.4. Coding Conventions 33
2.3.5. Risks Management 33
2.3.6. Communication Plan 34

CHAPTER 3 – SOFTWARE REQUIREMENT SPECIFICATIONS 36
3.1. Purpose 36
3.2. Functional Requirements 37
3.2.1 Use Case Diagram 37
3.2.2 Business Rules 38
3.2.3 Use Cases 39
3.2.3.1 Project 41
3.2.3.2 Record 51
3.3.3.3 Log 64
3.3.3.4 Visualizer 67
3.3.3.5 Image Viewer 75
3.3.3.6 Compare UI 79
3.3.3.7 Report 93
3.3.3.8 App Settings 96
3.3.3.9 Other features 98
3.3. Non-functional Requirements 101
3.6.1. Usability 101
3.6.2. Maintainability and Extensibility 101
3.6.3. Performance 101
3.4. Entity Relationship Diagram 102
CHAPTER 4 – SOFTWARE DESIGN SPECIFICATIONS 103
4.1. Purpose 103
4.2. Overview of System Architecture 103
4.2.1 Diagram 103
4.2.2 System Architecture Explanation 103
4.2.3 Component Explanation 104
4.2.3.1 Interface Components 105
4.2.3.2 Application Programming Interface (API) 106
4.2.3.3 External Libraries 107
4.3. High-level Architectural Design 107
4.3.1. Architecture Layers Design 107
4.3.1.1 Diagram 107
4.3.1.2 Project Structure 108

4.3.3 User Interface Design 109
4.4 Data Structure Design 110
4.4.1 Diagram 110
4.4.2 Data Dictionary 110
4.4.2.1 Entities 110
4.4.2.2 Attributes 110
4.5 Detailed System Design 112
4.5.1 Class Diagram 112
4.5.2 Class Specifications 112
4.5.2.1 Record Test 112
4.5.2.2 Ribbon 115
4.5.2.3 Compare Screen 117
4.5.2.4 Visualizer 120
4.5.2.5 Log Screen 122
4.5.2.6 Report 123
4.5.2.7 Generic Log 123
4.5.2.8 Welcome 124
4.5.2.9 Workspace 125
4.5.3 Detailed Designs 128
4.5.3.1 Project 128
4.5.3.2 Record 137
4.5.3.3 Log 150
4.5.3.4 Test Visualizer 153
4.5.3.5 Image Viewer 160
4.5.3.6 Compare UI 163
4.5.3.7 App Settings 173
4.5.3.8 Report 175
4.5.3.9 Other Features 178
CHAPTER 5 – SOFTWARE TESTING DOCUMENTATION 181
5.1. Purpose 181
5.2. Scope of testing 181
5.2.1. Stages of testing 181
5.2.2. Types of testing 183
5.2.3. Range of testing 185

5.3. Test plan 186
5.3.1. Testing tools and environment 186
5.3.2. Resources and responsibilities 187
5.3.3. Test strategy 187
5.3.4. Features to be tested 191
5.3.5. Features not to be tested 192
5.4. Test approach 193
5.4.1. Unit test 193
5.4.2. Integration and System test 195
5.4.3. Regression test 195
5.4.4. Acceptance test 195
5.5. Test report 199
5.5.1. Iteration 1 199
5.5.2. Iteration 2 200
5.5.3. Iteration 3 201
5.5.4. Iteration 4 202
5.5.5. Iteration 5 203
CHAPTER 6 – SOFTWARE USER MANUAL 204
6.1. User Guidelines 204
6.2.1. Installment 204
6.2.2. Window Application Functions 213
6.2.2.1 Create Test Project 213
6.2.2.2 Create Test Case 214
6.2.2.3 Start Recording 215
6.2.2.4 Run Test Case 215
6.2.2.5 Compare Image 217
6.2.2.6 Report 218
REFERENCES 219

DEFINITIONS AND ACRONYMS

Acronym	Definition	Note
API	Application Programming Interface	
BA Leader	Business Analysis Leader	
BUL	Business Unit Leader	
TOBI	Tobi Automation Test	
FHN.MDN	Fsoft Hanoi – Major Delivery Unit	
JSON	Javascript Object Notation	
MVVM	Model-View-ViewModel	Design Pattern
UC	User Case	
UT	Unit Test	
SDD	System Design Description	
SRS	Software Requirements Specification	
SVN	Subversion	
XAML	Extensible Application Markup Language	
WPF	Window Presentation Framework	

CHAPTER 1 – INTRODUCTION

1.1. Purpose

Introduction chapter introduces the background and initial idea of the Capstone Project. In this chapter, we – the develop team – will provide information about project information, supervisor, project team members, the overview of existing systems, the problems and solutions of the project.

By comparing to the similar existing solutions, we also show advantages, benefits, limitation and risks that our system has to face up to in the future.

1.2. Project Information

Project name: **Tobi Automation Testing Tool**

Project code: **TOBI**

Project group name: **TOBI TEAM**

Product type: **Window Application**

Timeline: **From 09th January to 17th April 2019**

1.3. Project Members

1.3.1. Supervisor

Full name	Phone	E-mail	Title
Bui Ngoc Anh	0915343020	AnhBN@fpt.edu.vn	Lecturer

Table 1. Information of supervisor

1.3.2. Team Members

Full name	Student Code	Phone	E-mail	Role
Len Van Hau	SE04158	0378823196	HauLVSE04158@fpt.edu.vn	Leader
Phan Anh Tu	SE03994	0355713335	TuPASE03994@fpt.edu.vn	Member
Ngo Ngoc Lam	SE04185	0766669357	LamNNSE04185@fpt.edu.vn	Member
Nguyen Thi Giang	SE04376	0965659180	GiangNTSE04376@fpt.edu.vn	Member
Nguyen Dinh Dong	SE04430	0702264596	DongNDSE04430@fpt.edu.vn	Member

Table 2. Information of team members

1.3.3. Stake holder

	Full name	Phone	E-mail	Title
Stakeholder	Nguyen Manh Hieu	0966842415	HieuNM14@fsoft.com.vn	BA Leader, End user
Stakeholder	Nguyen Chanh Tin	0982128766	TinNC@fsoft.com.vn	BUL
Stakeholder	Nguyen Dinh Dan	0977040948	DanND@fsoft.com.vn	Vice BUL

Table 3. Information of stakeholders

1.4. Background

FHN.MDN is a specialized migration agency in FPT Software, and its main task is to convert old languages and technologies into newer languages and technologies. The reason is that the number of supported libraries, resources or developers for new languages and technologies are growing much faster than old languages and technologies. Some software packages need to be converted because older technology is no longer supported by the manufacturer for its errors or vulnerabilities.

In addition, customers always require that their system must have familiar user interfaces, while it needs to precisely meet any business logic of the screen. FHN.MDN works towards migrating one item at a time, and each item will be tested again and again with a certain number of test cases. However, these are done manually. This results in a task being evaluated many times, time consuming to check whether a bug was fixed or not, check if the old bug has resolved or conflicted. Besides, exchanging information between developers and tester takes a lot of time.

The FHN.MDN unit has built a service system for migration program, developed software packages to support members' task estimation, member productivity measurement, automatic migration pattern, software for storing tester data, and so on. Above all, they want to build a software to help their testers save time and effort.

1.5. Problem

TOBI Team has suggested to FHN. MDN to develop a software that helps testers to reduce the repetition of an operation, and testers will have more time to perform other

tasks. In addition, exchanging scripts to reproduce bugs will be a time-saving advantage for both developers and testers in developing and migrating software.

However, automation test is always a new field to many people in the FHN.MDN unit. Currently, they get used to the traditional approach, as they were not inclined to set up a development project for automation test. In fact, they are not sure whether developing automation test software is feasible.

Technology is the biggest problem. Many desired functions that the software can run, while FHN.MDN do not believe these can be done. Therefore, there is not much technology support from FHN.MDN during project development.

1.6. Existing systems

→ Supporting API and services testing,

→ Offering some AI/ML and analytics capabilities,

→ Popularity and maturity,

→References:

- Best Automation Testing Tools for 2019 (Top 10 reviews)

https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018-top-10-reviews-8a4a19f664d2

- Top 10 API Testing Tools (Details & Updates Done for You!)

https://medium.com/@alicealdaine/top-10-api-testing-tools-rest-soap-services-5395cb03cfa9

- Software Testing Trends for 2018 and Beyond

https://dzone.com/articles/software-testing-trends-for-2018-and-beyond

Product	Selenium	Katalon Studio	Unified Functional Testing	TestComplete	SoapUI
Available since	2004	2015	1998	1999	2005
Application Under Test	Web apps	Web (UI & API), Mobile apps	Web (UI &API), Mobile, Desktop, Packaged apps	Web (UI &API), Mobile, Desktop apps	API/Web service
Pricing	Free	Free	$$$$	$$$	$$
Supported Platforms	Windows Linux OS X	Windows Linux OS X	Windows	Windows	Windows Linux OS X
Scripting languages	Java, C#, Perl, Python, JavaScript, Ruby, PHP	Java/Groovy	VBScript	JavaScript, Python, VBScript, JScript, Delphi, C++ and C#	Groovy
Programming skills	Advanced skills needed to integrate various tools	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts
Ease of Installation and Use	Require advanced skills to install and use	Easy to setup and use	Complex in installation. Need training to properly use the tool	Easy to setup. Need training to properly use the tool	Easy to setup and use

Figure 1. Testing Frameworks

1. Selenium (2004)

- Selenium is the household name when it comes to test automation.
- It is considered the industry standard for user interface automation testing of Web applications.
- Almost nine out of ten testers are using or have ever used Selenium in their projects, according to Survey on Test Automation Challenges.
- For developers and testers who have experience and skills in programming and scripting,
- Users can write test scripts in many differents languages (such as Java, Groovy, Python, C#, PHP, Ruby, and Perl)
- To use Selenium effectively, users are required of advanced programming skills
- This is a main disadvantage of Selenium, which is addressed in integrated tools like Katalon Studio.

Website: http://www.seleniumhq.org/

License: Open-source

2. Katalon Studio (2015)

- Katalon Studio is a powerful and comprehensive automation solution for testing API, Web, and mobile apps.
- It has a rich feature set for these types of testing and supports multiple platforms including Windows, macOS, and Linux.
- Leveraging Selenium and Appium engines, Katalon Studio provides a uniquely integrated environment for testers who find difficulties in integrating

  and deploying different frameworks and libraries to use Selenium and Appium, as well as those who are already familiar with these engines.
- Katalon Studio has an intelligent test recording capability that generates resilient and maintainable locators,

  allowing test scripts to work even when the application under test is changed. Highlights of the tool include:

  - A complete feature set for test automation of API/Web services, Web, and mobile applications

- Katalon Studio supports both SOAP and RESTful for API and services testing
- Hundreds of built-in keywords for creating test cases
- Support BDD Cucumber to express test scenario in natural languages
- Can be used for both automated and exploratory testing
- Suitable for both pros and non-techies

Website: https://www.katalon.com/

License: Free

3. UFT (Unified Funtional Testing) (1998)

- UFT is a popular commercial tool to test desktop, Web, and mobile application.
- It has been extended to include a good set of capabilities for API testing.
- By supporting multiple platforms for the target application under test (AUT), UFT provides a convenient choice to test the AUT that operates on desktop, Web, and mobile.

- UFT provides several advanced capabilities for smart object detection, image-based object detection, and correction.
- Several highlights of the tool that support API testing:
  - Intuitive user interface for creating, executing, and reporting API tests
  - Support generating API tests from WADL documents
  - Tests' actions, activities, and parameters can be visualized in diagrams

Website: https://software.microfocus.com/fr-ca/software/uft

License: From $3,200 per year.

4. TestComplete (1999)

- TestComplete continues to be in the list this year for its powerful and comprehensive set of features for Web, mobile, and desktop application testing.
- Testers can use JavaScript, VBScript, Python, or C++Script to write test scripts.
- Like UFT, TestComplete has an object recognition engine that can accurately detect dynamic user interface elements.

■ This engine is especially useful in applications that have dynamic and frequently changing user interfaces.

■ Testers can easily use TestComplete's record and playback feature, like Katalon Studio.

■ They can insert checkpoints into test steps to verify results. As a product of SmartBear, TestComplete can be integrated easily with other products offered by SmartBear.

Website: https://smartbear.com/product/testcomplete/overview/

License: From $2,399 per user per year

5. SoapUI (2005)

■ SoapUI is not a test automation tool for Web or mobile app testing; but it can be a tool of choice to test API and services.

■ It is a headless functional testing tool specifically designed for API testing.

■ SoapUI supports both REST and SOAP services. API automation testers can use either the open-source or pro version.

■ The pro edition has a user-friendly interface and several advanced features such as assertion wizard, form editor, and SQL query builder.

■ SoapUI is a tool of ReadyAPI suite, offered by SmartBear.

■ The tool provides a quite comprehensive feature set for API testing with many advanced capabilities, including:

- Generating tests easily using drag and drop, point-and-click
- Powerful data-driven testing with data from files and databases
- Asynchronous testing
- Scripts can be reused easily
- Creating mock services with RESTful mocking

Website: https://www.soapui.org/

License: SoapUI is packaged and distributed in two editions, open source and Pro. The latter starts at $659 per year.

Note: SoapUI and Postman are two representatives of several top API testing tools in this list only.

To learn more about the most popular and powerful API testing tools for SOAP and REST services, please refer to this list.

https://medium.com/@alicealdaine/top-10-api-testing-tools-rest-soap-services-5395cb03cfa9

	TOBI Tool	Katalon Studio	TestComplete
Show differences between image pair	✔	X	✔
Compare Image Folders	✔	X	X
Project Explorer (IDE-like)	✔	✔	✔
Price (free)	✔	✔	X
Migration Test	✔	X	X
Automatic image accuracy check	✔	X	✔

Table 4. Existing System Comparison

1.7. Idea of Product

Look at the problem at FHN.MDN. Tobi Team found that it was possible to help them save those repetitive steps and help testers in identifying software errors.

When a tester learns the business logic of the screen/item and records the test steps again, then the tester/dev can then use the test script to run and check if the case is running properly.

The test cases only work once: Usually experienced BA/Test will create these scripts with step records or write scripts, then the testers only need to run and check the correctness of the case.

1.8. Proposal of System

After considering all the factors, we have decided to develop a software that aims to minimize testers' efforts for their work.

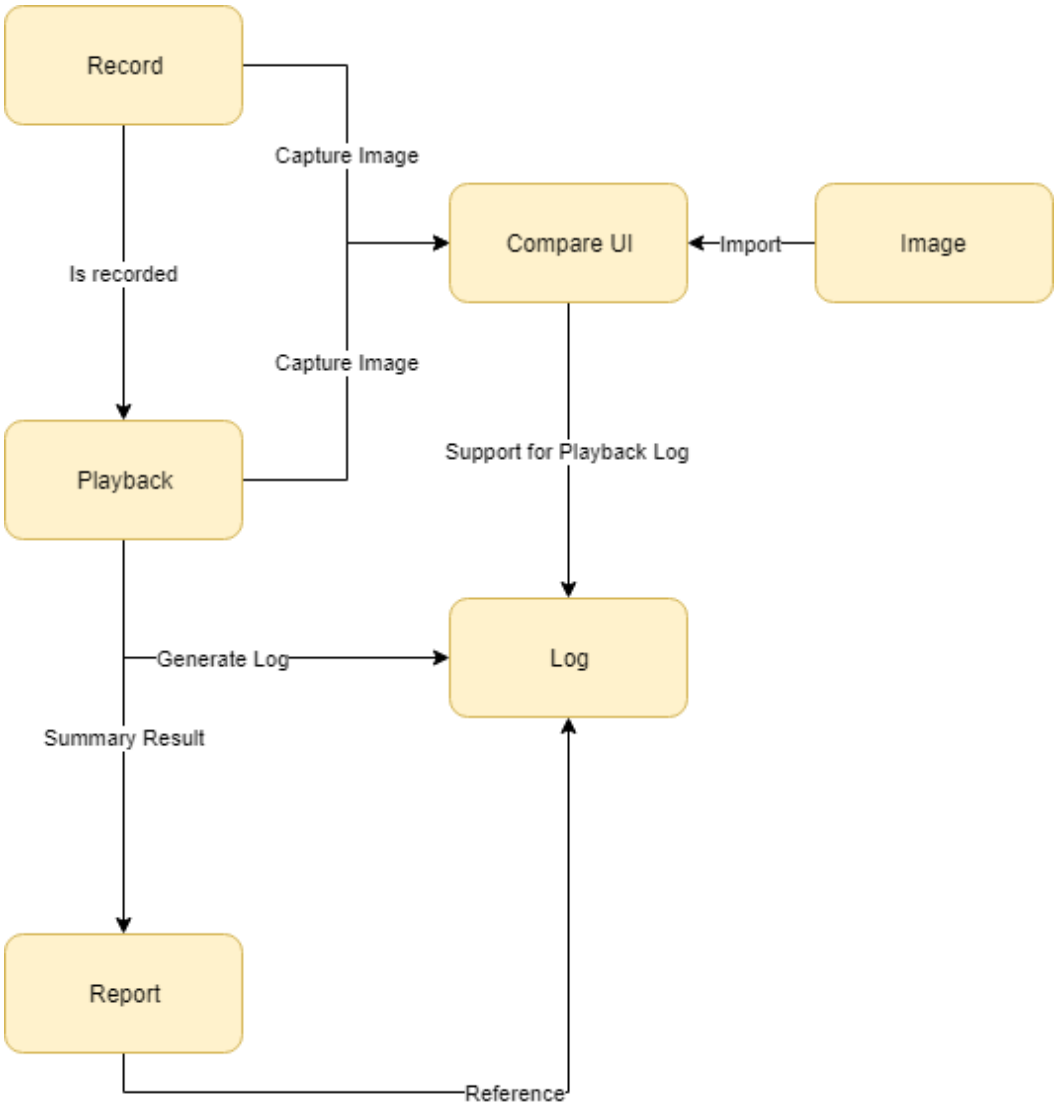


Figure 2. Main flow of TOBI application

(1) *Record:* Running a record will create a script for users, or users can write their own scripts if they have experience with the tool. In script form, users can arbitrarily change test cases to match the screen operation they are doing or respond to system changes without having to record again. Running the script can be transferred to the developer in charge of the screen or a lower-level tester.

(2) *Playback:* The built-in test script can be used to import and run in software. Test script will describe step by step so that the software can understand and simulate the user's testing process, then run again as the user actually uses.

(3) *Report:* Report in the form of sheet, show the steps and evaluate the status of steps to prove the business and the UI according to the steps is correct. The report shows how the UI images are displayed compared to the old version, thereby checking any tested errors. The report chart gives a more intuitive look to the system under testing.

(4) *Compare UI*: Can be checked automatically step by step, or users can import images to compare. After comparing, the user will receive a report about any changes from old to new.

(5) *Log:* The steps needed to save the evidence. After each step, it will automatically capture and save the image and evaluate whether it has passed the test or not. It is represented separately or can be integrated in the report function.

(6) *Image:* Image to import into the software's compare UI function

1.8.1. Basic functions

Users using the function must not be divided into different categories but only one role. This means users always have the right to use the following basic functions:

- Record
- Playback
- Log
- Report
- Compare UI

1.8.2. Out of Scope Functions

Due to the limitation of effort, our software will not contain the functions below, but we believe that these functions could be implemented in the future:

- Add check: The function allows users to predict the results in advance and ask whether the item is correct or not
- Server: The function allows users to interact with server and store test data.

Any functions which are not listed in SRS and SDD will be out of scope functions.

CHAPTER 2 – SOFTWARE PROJECT MANAGEMENT PLAN

2.1. Project Organization

2.1.1 Purpose

This chapter provides an overview of the project plan including project organization and project management plan.

2.1.2 Software Process Model

Figure 3. Iterative and Incremental Software Process Model

Tobi Automation Test project uses the Iterative & Incremental Software Process Model.

The Iterative & Incremental Software Process Model is mostly used when the scope of the project is big, the major requirements are defined clearly, and some more details will be added later in software development. Also, this model has these useful advantages:

- Main functions can be developed early & quickly.
- Testing and debugging during smaller iterations are easier.
- More flexible – easy & less costly to change if any.

- Easier to manage risk because risky pieces are identified and handled during its iteration.
- Iterations may be run simultaneously. A design team starts the next iteration while the current one is under testing.
- The project team learns from the first iteration and may use best practices and experiences to improve in the next iterations.

2.1.3 Organization Structure

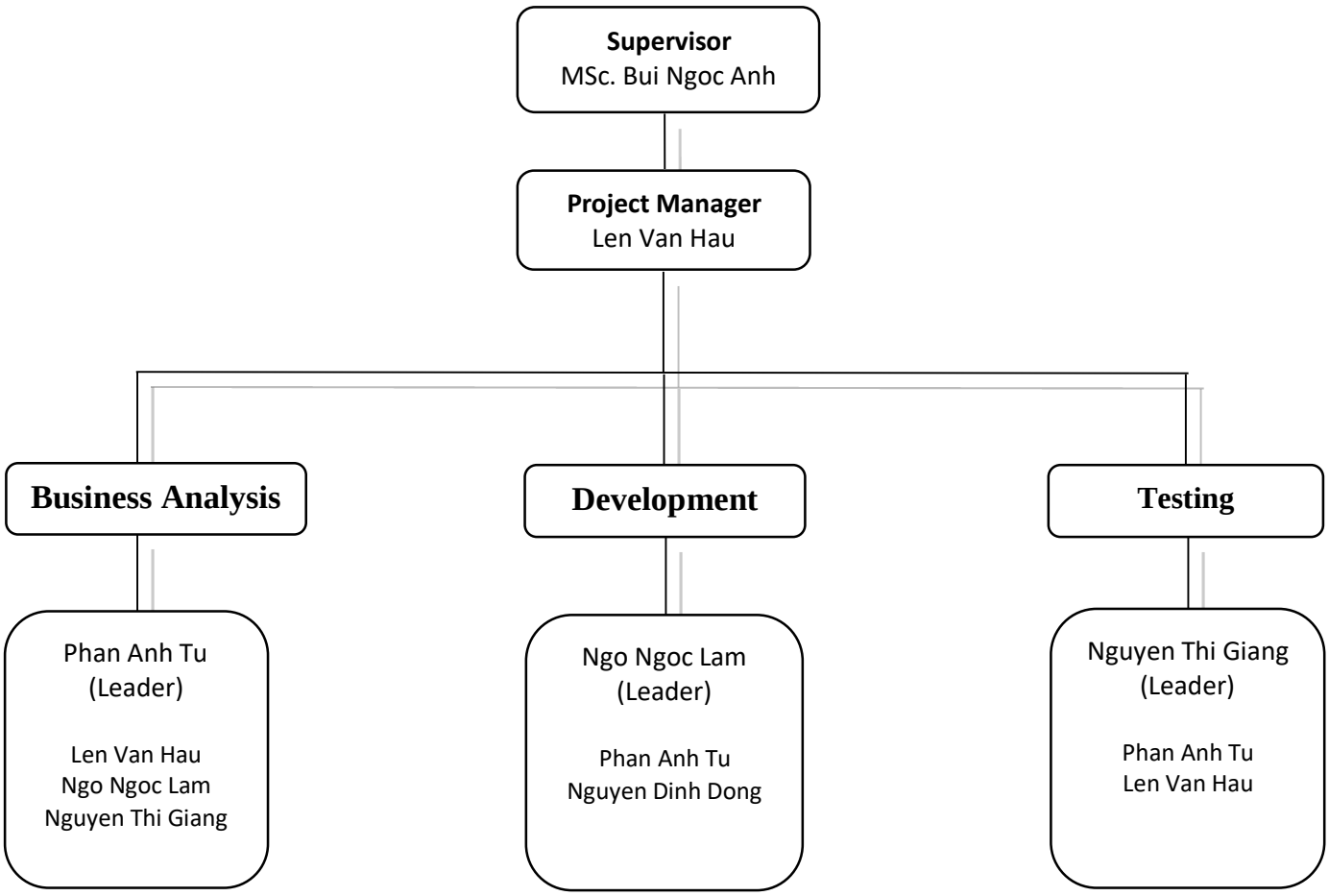


Figure 4. Organization structure

2.1.4 Roles and Responsibilities

Role	Responsibility
Project manager	Planning, developing schedules, coordinating communication, generally responsible for keeping the team's focus on the main goal.
Technical leader	Responsible for choosing and deciding what technologies should be used, as well as for overseeing the work being done by other developers.
Document leader	Document leader has responsibility to prepare all the documents relating to the project.
Quality Assurance Manager	Ensuring the product meets the certain standards of quality from requirements.
Developer	Developers have responsibilities to code product, review other developers' code. They set up, run unit test, receive bugs from testers and fix these bugs.
Test Leader	Responsible for test execution, including test set-up and test run, evaluation of test run and error recovery, defect logging and test results recording.
Tester	Involve in testing the product.
Designer	Designer has responsibilities to design user interface of product, draw mock-up screens for all user cases before implementing.
Business analyst	Business analysts have responsibilities to work with project manager to analyze business domain, define functions and scope of system. They work as a bridge between business problems and technology solutions.

Table 5. Roles and Responsibilities

2.1.5 Project Team Member

Team member	Role
HauLV	Project Manager
LamNN	Technical Leader, Developer
TuPA	Business Analyst, Document Leader, Developer
DongND	Developer, Designer.
GiangNT	Test Leader

Table 6. Project team member

2.1.6 Tools and Techniques

Programming languages	C#, XAML, C++
Framework	WPF, AvalonDock, Reogrid
Software architecture	MVVM pattern
Operating System	Windows 10
Version control	Git 2.9.0, Github, SVN
IDE/Editors	Visual Studio 2017, Visual Studio Code 1.30
UML tools	Astah Professional 8.0, Draw.io
Project Management Tools	Microsoft Project Management, Microsoft Excel
File Management Tools	Google Drive, TortoiseSVN
Document Tools	Microsoft Office 2016
Process model	Iterative and Incremental software process model.
Development process	Test-driven development.
Contact tools	Slack, Skype, Facebook
Testing Tools	MSTest, Microsoft Excel 2016
Libraries	MouseKeyHook, Tesseract, NewtonSoft JSON,..
Theme Resources	FluentWPF, WPF Toolkit, Material Design

Table 7. Tools and Techniques

2.2. Project Management Plan

2.3.1. Tasks Schedule

Iteration 1: Prepare and learn for starting automation project

- Duration: January, 9th 2019 – January, 16th 2019.
- Content:
    - Learn about automation test
    - Learn the available systems, analyze advantages / disadvantages
    - Build small demos to solve automation test problems
    - Build source code structure for phase coding
    - Learn the business logic needed for software Iteration 2: Focusing on developing client-side functions.

Iteration 2: Coding Recorder and development platform for the following functions

- Duration: January, 17th 2019 – January, 25th 2019.
- Content:
    - Building and applying software construction structure to the project
    - Make the demo applications related to Recorder
    - Make the required documents related to Recorder
    - Coding Recorder
    - Conduct tests and fix bugs for Recorder
    - Evaluate Recorder functions

Iteration 3: Coding Playback and development integrated with Recorder

- Duration: January, 28th 2019 – February, 1th 2019.
- Content:
    - Make the demo applications related to Playback
    - Make the required documents related to Playback
    - Coding Playback
    - Conduct tests and fix bugs for Playback
    - Evaluate Playback functions

- Integrate Recorder and Playback into a complete flow

Iteration 4: Coding Log and build structure for the Report function

- Duration: February, 11th 2019 – February, 15th 2019.
- Content:
  - Make the demo applications related to Log
  - Make the required documents related to Log
  - Coding Log
  - Conduct tests and fix bugs for Log
  - Evaluate Log function
  - Integrate log and report to Log and Playback feature
  - Design Log data structure suitable for report

Iteration 5: Coding Compare UI and build the Compare UI module separately for other systems later

- Duration: February, 18th 2019 – March, 1th 2019.
- Content:
  - Make the demo applications related to Compare UI
  - Make the required documents related to Compare UI
  - Coding Compare UI
  - Conduct tests and fix bugs for Compare UI
  - Evaluate Compare UI function
  - Integrate development from Compare UI and Recorder-Playback group
  - Develope external modules for Compare UI (external dll)

Iteration 6: Coding Report supports overview for users

- Duration: March, 4th 2019 – March, 18th 2019.
- Content:
  - Make the demo applications related to Report
  - Make the required documents related to Report

- Coding Report
- Conduct tests and fix bugs for Report
- Evaluate Report function
- Combine all functions into a complete flow

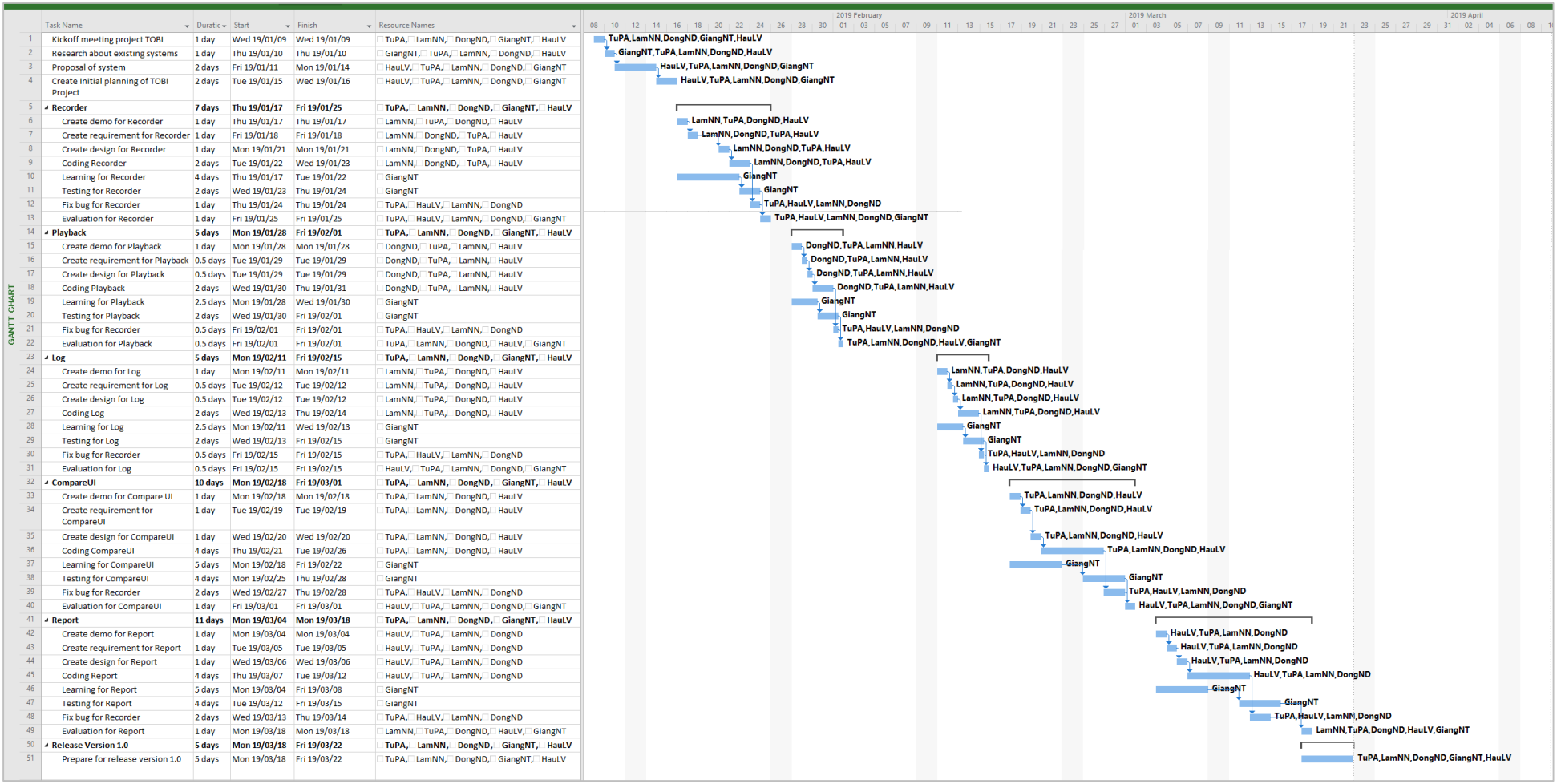


Figure 5. Project management in MS Project 2017

Detail information refers to “TOBI_CapstoneProject.mpp” file.

2.3.2. Project Milestones

No	Milestone	Completion Date	Verification
1	Project start	07/01/2019	Supervisor & Stakeholder
2	Test Project Manager	21/01/2019	Supervisor & Stakeholder
3	Record Step Function	28/01/2019	Supervisor & Stakeholder
4	Playback Function	11/02/2019	Supervisor & Stakeholder
5	Log Function	18/02/2019	Supervisor & Stakeholder
6	Compare UI	04/03/2019	Supervisor & Stakeholder
7	Report Function	18/03/2019	Supervisor & Stakeholder
8	Release version 1.0	25/03/2019	Supervisor & Stakeholder
9	Report from end user	15/04/2019	Supervisor & Stakeholder
10	Final report	22/04/2019	FPT University, Supervisor & Stakeholder
11	Defend project	03/05/2019	FPT University, supervisor

Table 8. Project Milestones

2.3.3. Meeting Minutes

A secretary whom the project manager assigns to will note all the meetings of team. He or she should write down ideas, contents discussion and notation by following the meeting minutes below:

Meeting/Project name	TOBI		
Date of meeting	7/1/2019	**Time/Type**	2 hours/Face-to-face
Meeting is hosted by	HauLV	**Location**	Canteen FPT University
Note Taker	GiangNT	**Time Keeper**	TuPA
1. Meeting Objective - Introducing team members - Setting up working rules, Contact tools - Choosing name and ideas for project - Other information			

2. Attendee

Name	Role	E-mail	Phone
Prof. AnhBN	Supervisor	AnhBN@fpt.edu.vn	0915343020
HauLV	Technical leader	HauLVSE04158@fpt.edu.vn	0378823196
TuPA	Document leader	TuPASE03994@fpt.edu.vn	0355713335
LamNN	Developer	LamNNSE04185@fpt.edu.vn	0766669357
DongND	Developer	DongNDSE04430@fpt.edu.vn	0702264596
GiangNT	Tester	GiangNTSE04376@fpt.edu.vn	0965659180

3. Content

No.	Action item(s)	Detail	Person in charge	Start date	End date
1	Rules	- Late for a meeting (max 5m - 50k) - Report of each day (max is 24h the next day) (20k) - If there is any issue, inform it early to other members. - Using sticker, spam 10k / line (apply to group chat: "Group preparing TOBI project")	All team members	7/1/2019	10/5/2019
2	Contact tools	- Facebook - Slack - Skype	- HauLV - LamNN	7/1/2019	10/5/2019
3	Choosing name and ideas for project	- Tobi Automation Testing Tool - Tool helps tester automatically records steps and play them back	All team members	7/1/2019	10/5/2019

4. Note

Meeting/Project name	TOBI		
Date of meeting	11/1/2019	**Time/Type**	2 hours/Face-to-face
Meeting is hosted by	HauLV	**Location**	Room 104R, FPT University
Note Taker	GiangNT	**Time Keeper**	TuPA

1. Meeting Objective

- Mr. Ngoc Anh: required a demo application for each specific function in the following week, required team to assign tasks to each member to do their own tasks.
- He asked the team to prepare: Access private/protected fields in the application, and make sure that the tool still works fine.

2. Attendee

Name	Role	E-mail	Phone
Prof. AnhBN	Supervisor	AnhBN@fpt.edu.vn	0915343020
HauLV	Technical leader	HauLVSE04158@fpt.edu.vn	0378823196
TuPA	Document leader	TuPASE03994@fpt.edu.vn	0355713335
LamNN	Developer	LamNNSE04185@fpt.edu.vn	0766669357
DongND	Developer	DongNDSE04430@fpt.edu.vn	0702264596
GiangNT	Tester	GiangNTSE04376@fpt.edu.vn	0965659180

3. Content

No.	Action item(s)	Detail	Person in charge	Start date	End date
1	Build a demo system	Lam and Dong will build a demo application (made of XPath)	LamNN, DongND	14/01/2019	18/01/2019
2	Documents	- Making documents at FPT University to protect FSoft projects and documents for Fsoft	TuPA (other members support)	14/01/2019	14/04/2019
3	Define functions	- Define what functions in relation to each other, and standardize the script language	HauLV (other members support)	14/01/2019	18/01/2019
4	Existing system	Collect materials related to existing systems	GiangNT	14/01/2019	16/01/2019

4. Note

Meeting/Project name	TOBI		
Date of meeting	29/3/2019	**Time/Type**	2 hours/Face-to-face
Meeting is hosted by	HauLV	**Location**	Room 104R, FPT University
Note Taker	GiangNT	**Time Keeper**	TuPA

1. Meeting Objective

- Project team wants to choose JSON-based files over traditional DBMS
- Support various target system: Windows 10, Windows 7,.. and programming languages: VB.Net, C#,..
- Fix existing bugs and run the application on actual projects.

2. Attendee

Name	Role	E-mail	Phone
Prof. AnhBN	Supervisor	AnhBN@fpt.edu.vn	0915343020
HauLV	Technical leader	HauLVSE04158@fpt.edu.vn	0378823196
TuPA	Document leader	TuPASE03994@fpt.edu.vn	0355713335
LamNN	Developer	LamNNSE04185@fpt.edu.vn	0766669357
DongND	Developer	DongNDSE04430@fpt.edu.vn	0702264596
GiangNT	Tester	GiangNTSE04376@fpt.edu.vn	0965659180

3. Content

No.	Action item(s)	Detail	Person in charge	Start date	End date
1	Reasons to support JSON-based files	- Prof.AnhBN comments about the advantages and limitations of DBMS/JSON-based files for data storage.	HauLV	30/03/2019	01/04/2019
2	Support Windows 7 (fix bugs related to step recorder)	- Prof.AnhBN: Migration testing will likely include: previous application works fine on older OS (Win 7), while updated version can't on Windows 10	LamNN, DongND	30/03/2019	06/04/2019
3	Test the application on actual projects	- Test the application on various systems and projects. For example: Netbeans projects	GiangNT32	30/03/2019	08/04/2018

4. Note

2.3.4. Coding Conventions

Purpose of coding conventions:

- Coding conventions create a consistent look to the code. They help readers to focus on content, not layout.
- Coding conventions enable readers to quickly understand by making assumptions based on previous experiences.
- Coding conventions facilitate copying, changing and maintaining the code.

Because of these reasons, we strictly follow the Convention Styles Guide below:

- C#: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions.
- XAML: https://github.com/cmaneu/xaml-coding-guidelines

2.3.5. Risks Management

No.	Description	Avoidance plan	Contingency plan	Impact
R1	Absence/illness of team members	Members have inform the project manager the reason why they are absent and build their own plan to keep up with the work process.	The project manager always has back up plan to avoid effect from the absence and ensure that problem will be solved.	High
R2	Conflicts among members	At the beginning, members should clear their roles and responsibilities.	The project manager has to find out the root that caused problem and solve it. Face-to-face meeting is a good method.	Medium
R3	Equipment of members is broken	Members must be careful with their equipment.	Members should use another equipment until the broken one is fixed or buy the new one.	Low
R4	Business problem	Any ideas are welcome but members have to discuss with	Making sure that the business logic of any ideas is carefully analyzed.	High

No.	Description	Avoidance plan	Contingency plan	Impact
		others and always focus on the reality and possibility.		
R5	Change management overload	Analyzing requirements carefully.	If there is a “must be changed” requirement, all team members must join the meeting to decide whether it should be implemented or not.	High
R6	Project team member misunderstands requirements	Members have to read requirement specification and related documents carefully.	Making sure that any miscommunication would be resolved.	Medium
R7	New technologies	Choosing technology based on members’ qualification. All team members must nurture by self-study.	When the technical leader chooses a new technology, he has to explain to all team members about the decision.	High

Table 9. Risks management

Probability			
high		R9	R1, R4
medium	R7	R6	R2, R5, R8
low	R3		
	low	medium	high
		Impact	

Table 10. Probability/impact matrix of risk

2.3.6. Communication Plan

We made plans for three kinds of meeting:

- Weekly meeting.
- Daily discussing.
- Unscheduled meeting.

Weekly meeting: All members will meet the supervisor on Wednesdays at the Alpha building, 104R/HB, FPT University. In every meeting, each member informs other members about what he or she has finished last week, status (fast, in time or slow), issues encountered and how to solve them. If any problem arises and one member cannot find the solution, all members will discuss to solve them. Subsequently, the project manager will assign tasks supposedly done next week and estimate time for doing these tasks. The secretary will write everything that our team discusses down in meeting minutes.

Daily discussing: For every working day, a small review section hosted by the project manager will take space at the working space (FHN.MDN – Fsoft). Each member has to review what he or she did yesterday, the problems remained and what need to be done today. By focusing on what each member has accomplished and will do, team members will know exactly the efforts that they need to spend in general.

Unscheduled meeting: Unscheduled meetings are usually online meetings. It happens when there are important problems involved all members and needed to be resolved immediately.

Communication channels:

- Face-to-face meeting.
- E-mail, Excel Sheets: Tracking and saving important tasks and information.
- Skype, Slack: Online meeting.
- Facebook Messenger, mobile phone: Chatting and informal conversations.

CHAPTER 3 – SOFTWARE REQUIREMENT SPECIFICATIONS

3.1. Purpose

Software Requirements Specifications chapter provides general, as well as specific information of the system. It also shows the describing of functional requirements and non-functional requirements. Following these requirements, our product will perform its intended functionality used in designing developing and testing. All members work based on the information that provided in this document.

3.2. Functional Requirements

3.2.1 Use Case Diagram

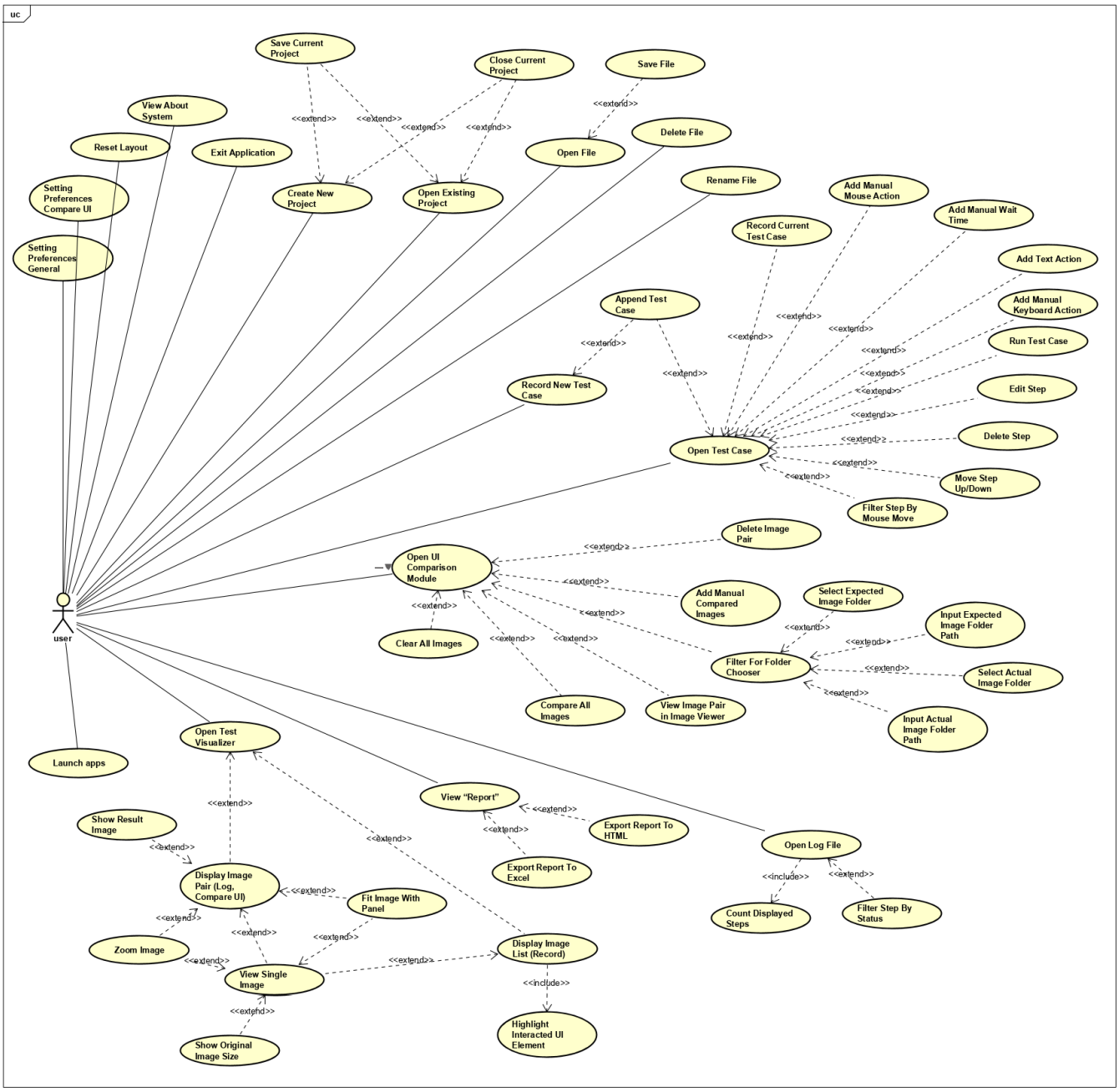


Figure 6. Use case diagram of Tobi Automation Test system

3.2.2 Business Rules

ID	Description
B01	Project name must be valid: starts with uppercase letter, special characters (!@#%^..), and spaces are not allowed.
B02	Test case name must be valid: starts with uppercase letter, special characters (!@#%^..), and spaces are not allowed.
B03	Location (project path) must be valid and exist within user's computer.
B04	When creating new project, user must enter "Project name" and "Location" correctly, if not, prompt user to enter again.
B05	Project name must not exist in the specified location.
B06	Creating new project or opening new one will close all document tabs.
B07	Always create Start Page when open new project.
B08	All test case files must be located in "testcases" folder in the project directory.
B09	All test logs must be located in "logs" folder in the project directory.
B10	User must open a project before using important features (record, compare, report, play all...)
B11	Show captured images in "Test Visualizer" in relation to steps displayed in Record document.
B12	Of all recorded steps, Mouse Move and Drag actions cannot be edited, the rest can be edited based on their types.
B13	Always append new steps to the end of the current test case.
B14	Always show steps with corresponding icon types.
B15	Clicking on a step with image will navigate to the associated image in Test Visualizer.
B16	Capture and highlight active window images for record & playback feature.
B17	Do not show Mouse Move and Drag actions by default.
B18	When move step up/down, images in Test Visualizer will be moved as well.
B19	Mouse move steps will be grouped to one step, if no other step type is available.
B20	Wait time exceeding 250 ms will be regarded as a separated step.
B21	Test logs generated from a particular test case are grouped and shown in hierarchy in Project Explorer.
B22	Only show Report Chart when clicking on Report tab.

B23	Display warning if Expected Image and Actual Image are not matched (based on accuracy check configured in Preferences).
B24	When playing back, if user cannot click on any element during step execution, the step will be considered “Failed”.
B25	User cannot use append feature when no test case tab is activated.
B26	User cannot use “Play All” feature when there is no available test case.
B27	User must save or discard all changes before quitting Tobi Automation Test.
B28	Expected image and Actual image must be consistent in width and height.
B29	Images shown in Image Viewer are adjusted to fit their contained panel by default.
B30	Hide Recorder dialog when capturing images.

Table 11. Business rules of Window application

3.2.3 Use Cases

Actor	Inherits From	Description
User		Everybody who uses Tobi Automation Test system.

Table 12. Actor Description

Section	ID	Actor	Name
Project	UC-1.0	User	Create New Project
	UC-2.0	User	Open Existing Project
	UC-3.0	User	Close Current Project
	UC-4.0	User	Save File
	UC-5.0	User	Save Current Project
	UC-6.0	User	Exit Application
	UC-7.0	User	Open File
	UC-8.0	User	Rename File
	UC-9.0	User	Delete File
Record	UC-10.0	User	Record New Test Case
	UC-11.0	User	Record Current Test Case
	UC-12.0	User	Append Test Case
	UC-13.0	User	Run Test Case

	UC-14.0	User	Play All Test Case
	UC-15.0	User	Add Manual Mouse Action
	UC-16.0	User	Add Manual Keyboard Action
	UC-17.0	User	Add Text Action
	UC-18.0	User	Add Manual Wait Time
	UC-19.0	User	Move Step Up/Down
	UC-20.0	User	Edit Step
	UC-21.0	User	Delete Step
	UC-22.0	User	Open Test Case
	UC-23.0	User	Filter Step By Mouse Move
Log	UC-24.0	User	Open Log File
	UC-25.0	User	Count Displayed Steps
	UC-26.0	User	Filter Step By Status
Visualizer	UC-27.0	User	Open Test Visualizer
	UC-28.0	User	Display Image List (Record)
	UC-29.0	User	Display Image Pair (Log, UI Comparison)
	UC-30.0	User	Fit Image With Panel
	UC-31.0	User	Zoom Image
	UC-32.0	User	Highlight Interacted UI Element
	UC-33.0	User	Show Result Image
Image Viewer	UC-34.0	User	View Single Image
	UC-35.0	User	Show Original Image Size
	UC-36.0	User	View Image Pair
Compare UI	UC-37.0	User	Open UI Comparison Module
	UC-38.0	User	Filter For Folder Chooser
	UC-39.0	User	Select Expected Image Folder
	UC-40.0	User	Input Expected Image Folder Path
	UC-41.0	User	Select Actual Image Folder

	UC-42.0	User	Input Actual Image Folder Path
	UC-43.0	User	Add Manual Compared Images
	UC-44.0	User	Clear All Images
	UC-45.0	User	Compare All Images
	UC-46.0	User	View Image Pair in Image Viewer
	UC-47.0	User	Delete Image Pair
Report	UC-48.0	User	View "Report"
	UC-49.0	User	Export Report To Excel
	UC-50.0	User	Export Report To HTML
Settings	UC-51.0	User	Setting preferences general
	UC-52.0	User	Setting preferences Compare UI
Other	UC-53.0	User	Reset Layout
	UC-54.0	User	View About System
	UC-55.0	User	Launch Apps

Table 13. Use Case List

3.2.3.1 Project

3.2.3.1.1 Create New Project

Use case specification

UC ID and Name:	**UC-1.0 Create New Project**		
Version	**1.1**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Create new project on TOBI system.		
Preconditions:	**PRE-1.1** User has opened TOBI system.		
Post-conditions:	**POST-1.1** When the normal flow completes successfully, a new project is created on TOBI system and become active project.		
Normal Flow:	**1.0 Create New Project** 1. User opens Tobi Automation Test system.		

	2. System displays TOBI main screen. 3. User clicks *FILE* menu on the menu bar, and clicks on New Project. 4. System displays Create New Project dialog box. 5. User enters Project Name in the dialog box. 6. User clicks on Browse button to choose location for the project. 7. User clicks on OK button to confirm project creation. 8. System creates a new basic project with the project name provided.
Alternative Flows:	**1.1 Create New Project** 1. User clicks on New button on toolbar menu 2. Go to step 4 of normal flow **1.2 Create New Project** 1. From Start Page, user clicks on Create Project 2. Go to step 4 of normal flow **1.3 Cancel project creation** 1. When Create New Project dialog shows up, user clicks on Cancel. 2. System displays main screen.
Exceptions:	**1.0-E1 – Cannot create new project with the name specified.** System displays error message (The project name has existed already!) **1.0-E2 – Project name violates business rules.** System displays error message (Project name is not valid). **1.0-E3 –Location violates business rules.** Disable OK button.
Priority:	High
Frequency of Use:	Low
Business Rules:	B01, B03, B04, B05, B06, B07, B10
Other Information:	N/A.
Assumptions:	User has already installed Tobi Automation Test.

Table 14. Use case: Create New Project

3.2.3.1.2 Open Existing Project

Use Case Specification

UC ID and Name:	**UC-2.0 Open Existing Project**		
Version	**1.2**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Open an existing project located in user's computer.		
Preconditions:	**PRE-2.1** User has opened TOBI system.		
Post-conditions:	**POST-2.1** When the normal flow completes successfully, the chosen project will become active project.		
Normal Flow:	**2.0 Open Existing Project** 1. System displays TOBI main screen. 2. User clicks *FILE* menu on the menu bar, and clicks on Open Project. 3. System displays Open dialog to choose Project. 4. User locates to Project folder, and click on file with *.tobi extension (Project file). 5. User clicks on Open button. 6. System creates a fully structured project with the information given by *tobi project file.		
Alternative Flows:	**2.1 Open Existing Project** 1. User clicks on Open button on toolbar menu 2. Go to step 4 of normal flow **2.2 Open Existing Project** 1. From Start Page, user clicks on Open Existing Project 2. Go to step 4 of normal flow **2.3 Stop system from opening project** 1. When Open dialog shows up, user clicks on Cancel. 2. System displays main screen.		
Exceptions:	**2.0-E1 – Project file is corrupted.** System displays error message (Project file is not valid!). **2.0-E2 – Missing some files.** System displays error message (Failed to build project structure!).		

Priority:	Low
Frequency of Use:	Medium
Business Rules:	B03, B07, B08, B09, B10
Other Information:	Default location to open existing project is Documents folder.
Assumptions:	Project has already existed in the specified location.

Table 15. Use case: Open Existing Project

3.2.3.1.3 Close Current Project

Use Case Specification

UC ID and Name:	**UC-3.0 Close Current Project**		
Version	**1.0**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Close the current active project.		
Preconditions:	**PRE-3.1** User has already opened a project.		
Post-conditions:	**POST-3.1** When the normal flow completes successfully, the current active project will be closed.		
Normal Flow:	**3.0 Close Current Project** 1. System displays TOBI main screen with an active project. 2. User clicks *FILE* menu on the menu bar, and clicks on Close Project. 3. System displays confirmation dialog, and clicks on Close Project. 4. System closes the current project and any relevant information (opened active tabs, visualizer,..)		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Low		
Frequency of	Medium		

Use:	
Business Rules:	B10
Other Information:	N/A
Assumptions:	User has opened TOBI system.

Table 16. Use case: Close Current Project

3.2.3.1.4 Save File

Use Case Specification

UC ID and Name:	**UC-4.0 Save File**		
Version	**1.1**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Save the current active tab.		
Preconditions:	**PRE-4.1** User has opened a record tab, and the tab is active.		
Post-conditions:	**POST-4.1** When the normal flow completes successfully, the current active tab will be saved.		
Normal Flow:	**4.0 Save File** 1. System displays TOBI main screen with an active project. 2. User opens a record test file. 3. User changes the content of the said test file. 4. System displays the test file title with an asterisk symbol (*). 5. User clicks *FILE* menu on the menu bar, and clicks on Save. 6. System saves the test file, its title reverts back to normal (no asterisk *)		
Alternative Flows:	**4.1 Save File** 1. Do the same steps from 1-4 of normal flow 2. Press shortcut Ctrl + S. 3. Go to step 6 of normal flow.		
Exceptions	**N/A**		
Priority:	Low		

Frequency of Use:	Medium
Business Rules:	B02, B08, B09
Other Information:	N/A
Assumptions:	User has created one or more test files.

Table 17. Use case: Save File

3.2.3.1.5 Save Current Project

Use Case Specification

UC ID and Name:	**UC-5.0 Save Current Project**		
Version	**1.2**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Save the current active project.		
Preconditions:	**PRE-5.1** User has opened a project.		
Post-conditions:	**POST-5.1** When the normal flow completes successfully, all relevant information about the project will be saved.		
Normal Flow:	**5.0 Save Current Project** 1. System displays TOBI main screen with an active project. 2. User changes some record test files. 3. User clicks *FILE* menu on the menu bar, and clicks on Save All. 4. System saves all the test files, their titles revert back to normal (no asterisk *)		
Alternative Flows:	**5.1 Save Current Project** 1. System displays TOBI main screen with an active project. 2. User changes some record test files. 3. User presses shortcut Ctrl + Shift + S. 4. Go to step 4 of normal flow.		
Exceptions	**N/A**		
Priority:	Low		
Frequency of	Medium		

Use:	
Business Rules:	B02, B08, B09
Other Information:	N/A
Assumptions:	User has created some test files.

Table 18. Use case: Save Current Project

3.2.3.1.6 Exit Application

Use Case Specification

UC ID and Name:	**UC-6.0 Exit Application**		
Version	**1.0**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Exit the application.		
Preconditions:	**PRE-6.1** User has opened TOBI application.		
Post-conditions:	**POST-6.1** When the normal flow completes successfully, all relevant information about the active project will be saved and the application exits.		
Normal Flow:	**6.0 Exit Application** 1. System displays TOBI main screen with an active project. 2. User clicks *FILE* menu on the menu bar, and clicks on Exit. 3. System displays confirmation dialog, asking whether user wants to save all changes. 4. User clicks on Yes. 5. System save all files, and the application exits.		
Alternative Flows:	**6.1 Exit Application** 1. Do the same steps from 1-3. 2. User clicks on No. 4. The application exits **6.2 Exit Application** 1. Do the same steps from 1-3. 2. User clicks on Cancel.		

	4. The application stays opened.
Exceptions	**N/A**
Priority:	Low
Frequency of Use:	Medium
Business Rules:	N/A
Other Information:	N/A
Assumptions:	User has opened TOBI application.

Table 19. Use case: Exit Application

3.2.3.1.7 Open File

Use Case Specification

UC ID and Name:	**UC-7.0 Open File**		
Version	**1.4**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Open the file from Project Explorer using context menu.		
Preconditions:	**PRE-7.1** The file needed to open is available.		
Post-conditions:	**POST-7.1** When the normal flow completes successfully, the file will be opened in new tab and displays its content.		
Normal Flow:	**7.0 Open File** 1. System displays TOBI main screen with an active project. 2. User double clicks on a file from Project Explorer 3. System displays a new tab showing the file content. The layout shows differently depending on the type of file.		

Alternative Flows:	**7.1 Open File** 1. System displays TOBI main screen with an active project. 2. User right clicks on a file from Project Explorer. 3. System displays a dropdown list. 4. User clicks on Open menu. 5. System displays a new tab showing the file content. The file layout shows differently depending on the type of file.
Exceptions	**N/A**
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B03, B08, B09, B10
Other Information:	N/A
Assumptions:	User has opened TOBI application.

Table 20. Use case: Open File

3.2.3.1.8 Rename File

Use Case Specification

UC ID and Name:	**UC-8.0 Rename File**		
Version	**1.2**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Rename the file from Project Explorer using context menu.		
Preconditions:	**PRE-8.1** The file needed to rename is available.		
Post-conditions:	**POST-8.1** When the normal flow completes successfully, the file will be renamed.		
Normal Flow:	**8.0 Rename File** 1. System displays TOBI main screen with an active project. 2. User right clicks on a file from Project Explorer. 3. System displays a dropdown list. 4. User clicks on Rename menu.		

	5. System displays Rename dialog. 6. User enters a new name 7. User clicks OK. 8. System displays the file with the name user has just created.
Alternative Flows:	**8.1 Rename File** 1. Do the same steps from 1-5 of normal flow. 2. User clicks on Cancel 3. System closes the Rename dialog, the file remains the same.
Exceptions	**N/A**
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B02, B03, B07, B08
Other Information:	N/A
Assumptions:	User has opened TOBI application.

Table 21. Use case: Rename File

3.2.3.1.9 Delete File

Use Case Specification

UC ID and Name:	**UC-9.0 Delete File**		
Version	**1.1**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Delete the file from Project Explorer using context menu.		
Preconditions:	**PRE-9.1** The file needed to delete is available.		
Post-conditions:	**POST-9.1** When the normal flow completes successfully, the file will be deleted from project.		

Normal Flow:	**9.0 Delete File** 1. System displays TOBI main screen with an active project. 2. User right clicks on a file from Project Explorer. 3. System displays a dropdown list. 4. User clicks on Delete menu. 5. System displays "Confirm file deletion" dialog. 6. User clicks on OK. 7. System deletes the chosen file, any relevant information and associated files.
Alternative Flows:	**9.1 Open File** 1. Do the same steps from 1-5. 2. User clicks on Cancel. 3. System closes the "Confirm file deletion" dialog, the file remains the same.
Exceptions	**N/A**
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B03, B08, B09
Other Information:	N/A
Assumptions:	Project has opened TOBI application.

Table 22. Use case: Delete File

3.2.3.2 Record

3.2.3.2.1 Record New Test Case

Use Case Specification

UC ID and Name:	**UC-10.0 Record New Test Case**		
Version	**1.5**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to create new test script		
Preconditions:	**PRE-10.1** User has created project		

Post-conditions:	**POST-10.1** Test Case is created
Normal Flow:	**10.0 Record New Test Script** 1. User clicks "Record" button in toolbar 2. System displays "Create test case" dialog 3. User fills in test script name 4. User clicks "OK" button in dialog 5. System opens a popup recorder, minimizes main screen, creates new test case and starts recording 6. User clicks Stop button 7. System maximizes the main screen
Alternative Flows:	**10.1 Record New Test Script** 1. User clicks "Record" button in "Welcome" page 2. Go to step 2 of normal flow
Exceptions	**10.0-E1 – Test script name violates business rules** System displays error message (test script name is not valid)
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B02, B03, B08, B09, B11, B13, B14, B30
Other Information:	N/A
Assumptions:	Project has opened TOBI application.

Table 23. Use case: Record New Test Case

3.2.3.2.2 Record Current Test Case

Use case specification

UC ID and Name:	**UC-11.0 Record Current Test Case**		
Version	**1.4**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to overwrite current test case		
Preconditions:	**PRE-11.1** User has created test case		
Post-conditions:	**POST-11.1** Test script is overwritten		

Normal Flow:	**11.0 Record Current Test Case** 1. User clicks "Record" button in test case toolbar 2. System minimizes and displays "Record popup" window 3. User starts recording 4. When it's done, user clicks "Stop" button 5. System stops recording and maximizes TOBI's main window.
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	B03, B08, B09, B11, B13, B14, B30
Other Information:	N/A.
Assumptions:	N/A

Table 24. Use case: Record Current Test Case

3.2.3.2.3 Append Test Case

Use case specification

UC ID and Name:	**UC-12.0 Append Test Case**		
Version	**1.2**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to append to existing test cases		
Preconditions:	**PRE-12.1** User has created test case		
Post-conditions:	**POST-12.1** Append new record to the end of selected test case		
Normal Flow:	**12.0 Record Current Test Case** 1. User clicks "Append" button in test script's document tab 2. System displays record popup window, and minimizes the main screen		

	3. User starts recording 4. System appends new records to current test case 5. User clicks Stop button 6. System stops recording and maximizes the main screen
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	B03, B08, B09, B11, B13, B14, B30
Other Information:	N/A.
Assumptions:	N/A

Table 25. Use case: Append Test Case

3.2.3.2.4 Run Test Case

Use case specification

UC ID and Name:	**UC-13.0 Run Test Case**		
Version	**1.3**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to run test case		
Preconditions:	**PRE-13.1** User has created test case		
Post-conditions:	**POST-13.1** Run test case and generate test log		
Normal Flow:	**13.0 Run Test Case** 1. User clicks “Run Test” button in test case toolbar 2. System starts running the test case while minimizing the window. 3. System generates test log after the test case finishes running 4. System maximizes the main screen.		
Alternative Flows:	N/A		
Exceptions:	N/A		

Priority:	Medium
Frequency of Use:	Medium
Business Rules:	B03, B08, B09, B11, B13, B14, B17, B19, B21
Other Information:	N/A.
Assumptions:	N/A

Table 26. Use case: Run Test Case

3.2.3.2.5 Play All Test Case

Use case specification

UC ID and Name:	**UC-14.0 Play All Test Case**		
Version	**1.3**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to play all test cases in the project		
Preconditions:	**PRE-14.1** User has created test cases		
Post-conditions:	**POST-14.1** Run test case and generate generic test logs		
Normal Flow:	**14.0 Run Test Case** 1. User clicks “Play All” button in main toolbar 2. System starts running all the test cases while minimizing the window. 3. System generates generic test log after all the test cases finish running 4. System maximizes the main screen.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	B03, B08, B09, B11, B13, B14, B17, B19, B21		
Other Information:	N/A.		
Assumptions:	N/A		

Table 26. Use case: Play All Test Case

14.1.1.1.1 Add Manual Mouse Action

Use case specification

UC ID and Name:	**UC-15.0 Add Manual Mouse Action**		
Version	**1.2**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to add mouse action		
Preconditions:	**PRE-15.1** User has created test case		
Post-conditions:	**POST-15.1** Add mouse action to test case		
Normal Flow:	**15.0 Add Manual Mouse Action** 1. User clicks “Click” button in test case toolbar 2. System displays “Mouse click action” dialog 3. User chooses action, button and coordinates of mouse in popup 4. User clicks “OK” button in dialog 5. System adds new mouse action to test case		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	B03, B08, B09, B13, B14		
Other Information:	N/A.		
Assumptions:	N/A		

Table 27. Use case: Add Manual Mouse Action

14.1.1.1.2 Add Manual Keyboard Action

Use case specification

UC ID and Name:	**UC-16.0 Add Manual Keyboard Action**		
Version	**1.3**		
Created by:	**DongND**	Date Created:	**11/04/2019**

Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to add keyboard action		
Preconditions:	**PRE-16.1** User has created test case		
Post-conditions:	**POST-16.1** Add keyboard action to test case		
Normal Flow:	**16.0 Add Manual Keyboard Action** 1. User clicks "Keys" button in test case toolbar 2. System displays "Key action" dialog 3. User chooses action and key 4. User clicks "OK" button in dialog 5. System adds new keyboard action to test case		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	B03, B08, B09, B13, B14		
Other Information:	N/A.		
Assumptions:	N/A		

Table 28. Use case: Add Manual Keyboard Action

14.1.1.1.3 Add Text Action

Use case specification

UC ID and Name:	**UC-17.0 Add Text Action**		
Version	**1.2**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to add text action		
Preconditions:	**PRE-17.1** User has created test case		
Post-conditions:	**POST-17.1** Add text action to test case		
Normal Flow:	**17.0 Add Manual Keyboard Action** 1. User clicks "Text" button in test case toolbar 2. System displays "Insert text" dialog		

	3. User fills in text to textbox 4. User clicks “OK” button in dialog 5. System adds new text action to test case
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	B03, B08, B09, B13, B14
Other Information:	N/A.
Assumptions:	N/A

Table 29. Use case: Add Text Action

14.1.1.1.4 Add Manual Wait Time

Use case specification

UC ID and Name:	**UC-18.0 Add Manual Wait Time**		
Version	**1.2**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to add wait action		
Preconditions:	**PRE-18.1** User has created test case		
Post-conditions:	**POST-18.1** Add wait time to test case		
Normal Flow:	**18.0 Add Manual Wait Time** 1. User clicks “Wait” button in test case toolbar 2. System displays “Wait time” dialog 3. User fills in time to wait in millisecond to the textbox on “Wait time” dialog 4. User clicks “OK” button in dialog 5. System adds new wait time step to test case		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		

Frequency of Use:	Medium
Business Rules:	B03, B08, B09, B13, B14, B20
Other Information:	N/A.
Assumptions:	N/A

Table 30. Use case: Add Manual Wait Time

14.1.1.1.5 Move Step Up/Down

Use case specification

UC ID and Name:	**UC-19.0 Move Step Up/Down**		
Version	**1.0**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to move step up or down in test case, in order for some steps to be played before other steps.		
Preconditions:	**PRE-19.1** User has created test case		
Post-conditions:	**POST-19.1** Move step up or down		
Normal Flow:	**19.0 Move Step Up/Down** 1. User chooses step that he/she wants to move in test case 2. User clicks “Up”/”Down” button in test script tab 3. System moves step based on the clicked operation		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	B03, B08, B09, B13, B14, B18		
Other Information:	N/A.		
Assumptions:	N/A		

Table 31. Use case: Move Step Up/Down

3.3.3.2.10 Edit Step

Use case specification

UC ID and Name:	**UC-20.0 Edit Step**		
Version	**1.1**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to edit step in test case		
Preconditions:	**PRE-20.1** User has created test case		
Post-conditions:	**POST-20.1** Step is modified		
Normal Flow:	**20.0 Edit Step** 1. User chooses step he/she wants to move in test case 2. User right clicks to step 3. System displays menu 4. User clicks “Edit” in menu 5. System opens Edit dialog 6. User edits information of step in dialog 7. User clicks “OK” button in dialog 8. System edits step successfully.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	B03, B08, B09, B14		
Other Information:	N/A.		
Assumptions:	N/A		

Table 32. Use case: Edit Step

3.3.3.2.11 Delete Step

Use case specification

UC ID and Name:	**UC-21.0 Delete Step**		
Version	**1.2**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to delete step		
Preconditions:	**PRE-21.1** User has created test case		
Post-conditions:	**POST-21.1** Step is deleted		
Normal Flow:	**21.0 Delete Step** 1. User chooses step he/she wants to move in test case 2. User right clicks to step 3. System displays menu 4. User clicks “Delete” in menu 5. System deletes step successfully.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	B03, B08, B09, B14		
Other Information:	N/A.		
Assumptions:	N/A		

Table 33. Use case: Delete Step

3.3.3.2.12 Open Test Case

Use case specification

UC ID and Name:	**UC-22.0 Open Test Case**		
Version	**1.0**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to open test case		
Preconditions:	**PRE-22.1** User has created project		
Post-conditions:	**POST-22.1** Open test case window		
Normal Flow:	**22.0 Open Test Case** 1. User double clicks to test case in Project Explorer 2. System opens test case window		
Alternative Flows:	**21.1 Open Test Case** 1. User right clicks on test case file in Project Explorer 2. System displays a dropdown list 3. User clicks on Open 4. System opens test case window		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	B03, B08, B13, B14, B17, B21, B25		
Other Information:	N/A.		
Assumptions:	N/A		

Table 34. Use case: Open Test Case

3.3.3.2.13 Filter Step By Mouse Move

Use case specification

UC ID and Name:	**UC-23.0 Filter Step by Mouse Move**		
Version	**1.1**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to hide all Mouse move steps		
Preconditions:	**PRE-23.1** User has created test case		
Post-conditions:	**POST-23.1** Hide mouse move steps		
Normal Flow:	**23.0 Filter Step by Mouse Move** 1. User clicks to “Mouse move” checkbox in test case toolbar 2. System filters and hides all mouse move steps		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	B03, B08, B13, B14, B17, B21, B25		
Other Information:	N/A.		
Assumptions:	N/A		

Table 35. Use case: Filter Step By Mouse Move

3.3.3.3 Log

3.3.3.3.1 Open Log File

Use case specification

UC ID and Name:	**UC-24.0 Open Log File**		
Version	**1.2**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to open log file		
Preconditions:	**PRE-24.1** User has run one or more test cases		
Post-conditions:	**POST-24.1** Open log file window		
Normal Flow:	**24.0 Open Log File** 1. User double clicks to test case in Project Explorer 2. System opens test case window		
Alternative Flows:	**24.1 Open Log File** 1. User right clicks on a log file in Project Explorer 2. System displays a dropdown list 3. User clicks on Open 4. System opens log file window		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	B03, B08, B09		
Other Information:	N/A.		
Assumptions:	N/A		

Table 36. Use case: Open Log File

3.3.3.3.2 Count Displayed Steps

Use case specification

UC ID and Name:	**UC-25.0 Count Displayed Steps**		
Version	**1.0**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to count log steps		
Preconditions:	**PRE-25.1** User has generated log file		
Post-conditions:	**POST-25.1** Show number of steps in log window		
Normal Flow:	**25.0 Count Displayed Steps** 1. User opens a log file 2. System opens log file window 3. System displays number of steps at the end of log window		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	N/A		
Other Information:	N/A.		
Assumptions:	Log file is not empty		

Table 37. Use case: Count Displayed Steps

3.3.3.3.3 Filter Steps By Status

Use case specification

UC ID and Name:	**UC-26.0 Filter Steps By Status**		
Version	**1.0**		
Created by:	**LamNN**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	This function allows user to filter steps by status		
Preconditions:	**PRE-26.1** User has created log file		
Post-conditions:	**POST-26.1** Display log steps based on filtered statuses		
Normal Flow:	**26.0 Filter Steps by Mouse Move** 1. User opens a log file 2. System opens log file window 3. User checks on filter boxes (Passed, Failed, Warning) 4. System displays log steps according to the chosen filter statuses		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	N/A		
Other Information:	N/A.		
Assumptions:	N/A		

Table 38. Use case: Filter Steps By Mouse Move

3.3.3.4 Visualizer

3.3.3.4.1 Open Test Visualizer

Use case specification

UC ID and Name:	**UC-27.0 Open Test Visualizer**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allows users to view and manipulate visual images in each type of file. (Image List / Image Pair)		
Preconditions:	**PRE-27.1** User has opened TOBI system.		
Post-conditions:	**POST-27.1** When the normal flow completes successfully, Test Visualizer is displayed on TOBI system.		
Normal Flow:	**27.0 Open Test Visualizer** 1. User do not open any project. 2. User clicks [Test Visualizer] button at the left bottom of the screen. 3. System displays Test Visualizer dock with blank content. **27.1 Open Test Visualizer** 1. User opens an existing project but do not open any files. 2. User clicks [Test Visualizer] button at the left bottom of the screen. 3. System displays Test Visualizer dock with blank content. **27.2 Open Test Visualizer** 1. User opens an existing project. 2. User opens an existing test case file of current project. 2. User clicks [Test Visualizer] button at the left bottom of the screen. 3. System displays Test Visualizer at the bottom showing a horizontally oriented list of associated images with steps containing image in test case file. **27.3 Open Test Visualizer** 1. User opens existing project. 2. User opens existing [Log] file or [UI comparison] module. 2. User clicks [Test Visualizer] button at the left bottom of the screen.		

	3. System displays Test Visualizer at the bottom with two areas showing [Expected Image] and [Actual Image] and some features: - [Fit Image] - [Zoom Image] - [Show Original Image Size] - [Image Viewer]
Alternative Flows:	N/A.
Exceptions:	N/A.
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 39. Use case: Open Test Visualizer

3.3.3.4.2 Display Image List (Record)

Use case specification

UC ID and Name:	**UC-28.0 Display Image List (Record)**		
Version	**1.3**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user to view captured image list in the current test case step by step		
Preconditions:	**PRE-28.1** Project has at least one non-empty test case and visualizer is opened or pinned.		
Post-conditions:	**POST-28.1** N/A.		
Normal Flow:	**28.0 Display Image List (Record)** 1. User double clicks on [Test case] file from Project Explorer 2. System displays a new tab showing the [Test case] file content. 3. [Test Visualizer] loads and displays captured image list and the [Image Viewer] button show captured image in its original size.		
Alternative Flows:	N/A.		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		
Business Rules:	B23, B24, B28, B29		
Other Information:	N/A.		
Assumptions:	N/A.		

Table 40. Use case: Display Image List (Record)

3.3.3.4.3 Display Image Pair (Log, UI Comparison)

Use case specification

UC ID and Name:	**UC-29.0 Display Image Pair (Log, UI Comparison)**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user to view image pair in screen		
Preconditions:	**PRE-29.1** Project has at least one non-empty test case and visualizer is opened or pinned.		
Post-conditions:	**POST-29.1** N/A.		
Normal Flow:	**29.0 Display Image Pair (Log, UI Comparison)** 1. User double clicks on [Log] file or [UI comparison] module from Project Explorer 2. System displays a new tab showing the content of [Log] file or [UI comparison] module. 3. [Test Visualizer] loads and displays captured image pair at the bottom with two areas showing [Expected Image] and [Actual Image] and the following features: - [Fit Image] - [Zoom Image] - [Show Original Image Size] - [Image Viewer]		
Alternative Flows:	N/A.		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		
Business Rules:	B23, B24, B28, B29		
Other Information:	N/A.		
Assumptions:	N/A.		

Table 41. Use case: Display Image Pair

3.3.3.4.4 Fit Image With Panel

Use case specification

UC ID and Name:	**UC-30.0 Fit Image With Panel**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user to reset image size that fits with its containing panel after zooming out/in in [Test visualizer]		
Preconditions:	**PRE-30.1** Active tab showing the [Log] file or [UI comparison] module content and select any step in [Log] file or select any image pair in [UI comparison] module.		
Post-conditions:	**POST-30.1** N/A.		
Normal Flow:	**30.0 Fit Image With Panel** 1. User changes slider value or scroll to zoom out/in an image. 2. User clicks on [Fit Image] in [Expected Image] area or [Actual Image] area. 3. System resizes and displays [Expected Image] and [Actual Image] that fit with the panel of each area.		
Alternative Flows:	N/A.		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		
Business Rules:	B23, B24, B28, B29		
Other Information:	N/A.		
Assumptions:	N/A.		

Table 41. Use case: Display Image Pair

3.3.3.4.5 Zoom Image

Use case specification

UC ID and Name:	**UC-31.0 Zoom Image**		
Version	**1.0**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user to zoom in/out images in [Test Visualizer] with slider or mouse scroll.		
Preconditions:	**PRE-31.1** Active tab showing the [Log] file or [UI comparison] module content and select any step in [Log] file or select any image pair in [UI comparison] module.		
Post-conditions:	**POST-31.1** N/A.		
Normal Flow:	**31.0 Zoom Image** 1. User changes slider value to zoom out/in image. 2. System scales image size and displays [Expected Image] and [Actual Image].		
Alternative Flows:	**31.1 Zoom Image** 1. User scrolls up/down to zooms in/out image. 2. System scales image size and displays [Expected Image] and [Actual Image].		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		
Business Rules:	B23, B24, B28, B29		
Other Information:	N/A.		
Assumptions:	N/A.		

Table 42. Use case: Zoom Image

3.3.3.4.6 Highlight Interacted UI Element

Use case specification

UC ID and Name:	**UC-32.0 Highlight Interacted UI Element**		
Version	**1.4**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Highlight UI Element which user interacted when executing recorded test case		
Preconditions:	**PRE-32.1** Project has at least one non-empty test case and visualizer is opened or pinned.		
Post-conditions:	**POST-32.1** N/A.		
Normal Flow:	**32.0 Highlight Interacted UI Element** 1. User double clicks on [Test case] file from Project Explorer. 2. System displays Test Visualizer at the bottom showing a horizontally oriented list of associated images with steps containing images in test case file. 3. Each image will highlight the interacted UI element with the previously recorded parameters.		
Alternative Flows:	N/A.		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		
Business Rules:	B23, B24, B28, B29		
Other Information:	N/A.		
Assumptions:	N/A.		

Table 43. Use case: Highlight Interacted UI Element

3.3.3.4.7 Show Result Image

Use case specification

UC ID and Name:	**UC-33.0 Show Result Image**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user to switch between actual image and result image after image pair has been compared.		
Preconditions:	**PRE-33.1** Active tab showing the [Log] file or [UI comparison] module content and select any step in [Log] file or select any image pair in [UI comparison] module.		
Post-conditions:	**POST-33.1** N/A.		
Normal Flow:	**33.0 Show Result Image** 1. User clicks on [Show Result Image] button in [Actual Image] area. 2. System displays [Actual Image] area's content by the following steps: - Actual image is hided. - [Show Result Image] button switchs to [Show Actual Image] - Result Image is displayed. If image pair is different size. System displays message: + "Images have different widths." + "Images have different heights." + "Images have different heights. Images have different widths."		
Alternative Flows:	N/A.		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		
Business Rules:	B23, B24, B28, B29		

Other Information:	N/A.
Assumptions:	N/A.

Table 44. Use case: Show Result Image

3.3.3.5 Image Viewer

3.3.3.5.1 View Single Image

Use case specification

UC ID and Name:	**UC-34.0 View Single Image**		
Version	**1.1**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user to view image detail with some default features.		
Preconditions:	**PRE-34.1** File is opened with content containing at least one image and visualizer is opened or pinned.		
Post-conditions:	**POST-34.1** When the normal flow completes successfully, interface of Image Viewer is displayed on TOBI system in new window		
Normal Flow:	**34.0 View Single Image** 1. User clicks on any [Test case] file from Project Explorer. 2. System displays captured image list in Test Visualizer. 3. User clicks on [Image Viewer] button of any image item. 4. System displays [Image Viewer] in new window with single [Result] tab with selected image and some default features as follow: - [Fit Image] - [Show original Image Size] - [Zoom Image] **34.1 View Single Image** 1. User clicks on any [Log] file from Project Explorer, then select any log step contains image. 2. System displays image pair on Test Visualizer. 3. User clicks on [Image Viewer] button of [Expected Image] or [Actual Image]. 4. System displays [Image Viewer] in new window with single [Result] tab with selected image and some default features as follow:		

	- [Fit Image] - [Show original Image Size] - [Zoom Image] - **34.2 View Single Image** 1. User clicks on [UI Comparison module] from Project Explorer, then select any image pair or add new image pair. 2. System displays image pair on Test Visualizer. 3. User clicks on [Image Viewer] button of [Expected Image] or [Actual Image]. 4. System displays [Image Viewer] in new window with single [Result] tab with selected image and some default features as follow: - [Fit Image] - [Show original Image Size] - [Zoom Image]
Alternative Flows:	**34.2 View Single Image** 1. User clicks on [UI Comparison module]. 2. User clicks [Add] button, then input only [Expected Image] or [Actual Image] 3. User clicks on [Image Viewer] button in current row. 4. System displays [Image Viewer] in new window with single [Result] tab with empty content.
Exceptions:	N/A.
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 45. Use case: View Single Image

3.3.3.5.2 Show Original Image Size

Use case specification

UC ID and Name:	**UC-35.0 Show Original Image Size**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user to view image with original size in Image Viewer.		
Preconditions:	**PRE-35.1** Image viewer is opened with image not null.		
Post-conditions:	**POST-35.1** Display image with original size.		
Normal Flow:	**UC-35.0 Show Original Image Size** 1. User changes slider value or scroll to zoom out/in on Image Viewer in order to change current size of image. 2. User clicks on [Show Original Size] in feature bar on Image Viewer. 3. System resizes and displays image in its original size.		
Alternative Flows:	N/A.		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		
Business Rules:	B23, B24, B28, B29		
Other Information:	N/A.		
Assumptions:	N/A.		

Table 46. Use case: Show Original Image Size

3.3.3.5.3 View Image Pair

Use case specification

UC ID and Name:	**UC-36.0 View Image Pair**		
Version	**1.1**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allows users to observe details and visually compare image pair.		
Preconditions:	**PRE-36.1** UI Comparison Module is opened.		
Post-conditions:	**POST-36.1** Image Viewer is displayed.		
Normal Flow:	**36.0 View Image Pair in Image Viewer** 1. User inputs full image pairs then clicks [Compare] button. 2. System fills the result to datagrid. 3. User clicks [Image Viewer] button in any row of datagrid. 4. System displays Image Viewer window with 3 tabs: - [Result] tab: show image result after compared. - [Fade] tab: show 2 original images let user manual compare. - [Slider] tab: show 2 original images let user manual compare. **36.1 View Image Pair in Image Viewer** 1. User inputs full image pairs then do not execute compare. 2. User clicks [Image Viewer] button in any row of datagrid. 3. System displays Image Viewer window with 3 tabs: - [Result] tab: image result is blank. - [Fade] tab: show 2 original images let user manual compare. - [Slider] tab: show 2 original images let user manual compare. **36.2 View Image Pair in Image Viewer** 1. User inputs only [Expected Image] or [Actual Image]. 2. User clicks [Image Viewer] button in any row of datagrid. 3. System displays Image Viewer window with only tab: - [Result] tab: image result is blank.		
Alternative	N/A.		

Flows:	
Exceptions:	N/A.
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 47. Use case: View Image Pair In Image Viewer

3.3.3.6 Compare UI

3.3.3.6.1 Open UI Comparison Module

Use case specification

UC ID and Name:	**UC-37.0 Open UI Comparison Module**		
Version	**1.3**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
scription:	Open UI Comparison Module on TOBI system.		
Preconditions:	**PRE-37.1** User has opened at least one project on TOBI system.		
Post-conditions:	**POST-37.1** When the normal flow completes successfully, interface of UI comparison module is displayed in TOBI system.		
Normal Flow:	**37.0 Open UI Comparison Module** 1. System displays TOBI main screen with an active project. 2. User double clicks on [Compare UI] file on [Project Explorer]. 3. System displays UI comparison module as active tab on [Workspace].		

Alternative Flows:	**37.1 Open UI Comparison Module** 1. System displays TOBI main screen with an active project. 2. User clicks [TOOLS] menu on the menu bar, then clicks on [Compare UI]. 3. Go to step 3 of normal flow
Exceptions:	N/A
Priority:	High
Frequency of Use:	Low
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	User has already installed Tobi Automation Test.

Table 48. Use case: Open UI Comparison Module

3.3.3.6.2 Filter For Folder Chooser

Use case specification

UC ID and Name:	**UC-38.0 Filter For Folder Chooser**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Show/Hide [Folder Chooser] on UI Comparison Module		
Preconditions:	**PRE-38.1** UI Comparison Module is displayed on workspace		
Post-conditions:	**POST-38.1** N/A.		
Normal Flow:	**38.0 Show Folder chooser feature** 1. User checked on checkbox [Folder Chooser]. 2. System shows [Expected folder] & [Actual folder] buttons, [Expected folder] & [Actual folder] textboxs are blank. **38.1 Hide Folder chooser feature** 1. User unchecked on checkbox [Folder Chooser]. 2. System hides [Expected folder] & [Actual folder] buttons, [Expected folder] & [Actual folder] textboxs are blank.		

Alternative Flows:	N/A.
Exceptions:	N/A.
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 49. Use case: Filter For Folder Chooser

3.3.3.6.3 Select Expected Image Folder

Use case specification

UC ID and Name:	**UC-39.0 Select Expected Image Folder**		
Version	**1.1**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user select expected folder contains more than one images.		
Preconditions:	**PRE-39.1** Checkbox [Folder chooser] is checked.		
Post-conditions:	**POST-39.1** N/A.		
Normal Flow:	**39.0 Select Expected Image Folder** 1. User clicks on [Expected Folder] button. 2. System displays [Folder chooser] dialog box. 3. User selects folder contains valid images, then click [Open] button. 4. System fills path of selected images to [Expected Image] column on datagrid and [Expected Folder] textbox displays path of selected folder.		
Alternative Flows:	**39.1 Select Expected Image Folder** 1. Do the same steps from 1-2 of normal flow		

	2. User selects folder contains valid images, then press [Enter] on keyboard. 3. Go to step 4 of normal flow. **39.2 Select Expected Image Folder** 1. Do the same steps from 1-2 of normal flow 2. User selects folder contains valid images, then click [Cancel] button. 3. System closes folder chooser dialog and nothing change.
Exceptions:	**39.0-E1 – Select folder contains invalid images.** System displays error message (Folder selected don't have any images valid or have some images invalid.) **39.0-E2 – Select folder doesn't have access permission.** System displays error message (You don't currently have permission to access this folder.) **39.0-E3 – Select folder contains some images don't have access permission.** System displays warning message (Selected folder have (n) restricted file(s).)
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 50. Use case: Select Expected Image Folder

3.3.3.6.4 Input Expected Image Folder Path

Use case specification

UC ID and Name:	**UC-40.0 Input Expected Image Folder Path**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user input expected folder contains more than one		

	images by path.
Preconditions:	**PRE-40.1** Checkbox [Folder chooser] is checked.
Post-conditions:	**POST-40.1** N/A.
Normal Flow:	**40.0 Input Expected Image Folder Path** 1. User input path of folder into [Expected Folder] textbox, then press [Enter]. 2. System fills path of selected images is filled to [Expected Image] column on datagrid.
Alternative Flows:	N/A.
Exceptions:	**40.0-E1 – Folder's path is invalid.** System displays error message ((path) is not a valid directory) **40.0-E2 – Inputed folder doesn't have access permission.** System displays error message (You don't currently have permission to access this folder.) **40.0-E3 – Inputed folder contains some images don't have access permission.** System displays warning message (Selected folder have (n) restricted file(s).)
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 51. Use case: Input Expected Image Folder Path

3.3.3.6.5 Select Actual Image Folder

Use case specification

UC ID and Name:	**UC-41.0 Select Actual Image Folder**		
Version	**1.1**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user select actual folder contains more than one images.		
Preconditions:	**PRE-41.1** Checkbox [Folder chooser] is checked.		
Post-conditions:	**POST-41.1** N/A.		
Normal Flow:	**41.0 Select Actual Image Folder** 1. User clicks on [Actual Folder] button. 2. System displays [Folder chooser] dialog box. 3. User selects folder contains valid images, then click [Open] button. 4. System fills path of selected images is filled to [Actual Image] column on datagrid and [Actual Folder] textbox displays path of selected folder.		
Alternative Flows:	**41.1 Select Actual Image Folder** 1. Do the same steps from 1-2 of normal flow 2. User selects folder contains valid images, then press [Enter] on keyboard. 3. Go to step 4 of normal flow. **41.2 Select Actual Image Folder** 1. Do the same steps from 1-2 of normal flow 2. User selects folder contains valid images, then click [Cancel] button. 3. System closes folder chooser dialog and nothing change.		
Exceptions:	**41.0-E1 – Select folder contains invalid images.** System displays error message (Folder selected don't have any images valid or have some images invalid.) **41.0-E2 – Select folder doesn't have access permission.** System displays error message (You don't currently have permission to access this folder.) **40.0-E3 – Select folder contains some images don't have access permission.**		

	System displays warning message (Selected folder have (n) restricted file(s).)
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 52. Use case: Select Actual Image Folder

3.3.3.6.6 Input Actual Image Folder Path

Use case specification

UC ID and Name:	**UC-42.0 Input Actual Image Folder Path**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user input actual folder contains more than one images by path.		
Preconditions:	**PRE-42.1** Checkbox [Folder chooser] is checked.		
Post-conditions:	**POST-42.1** N/A.		
Normal Flow:	**42.0 Input Actual Image Folder Path** 1. User input path of folder into [Actual Folder] textbox, then press [Enter]. 2. System fills path of selected images is filled to [Actual Image] column on datagrid.		
Alternative Flows:	N/A.		
Exceptions:	**42.0-E1 – Folder's path is invalid.** System displays error message ((path) is not a valid directory)		

	42.0-E2 – Inputed folder doesn't have access permission. System displays error message (You don't currently have permission to access this folder.) **42.0-E3 – Inputed folder contains some images don't have access permission.** System displays warning message (Selected folder have (n) restricted file(s).)
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 53. Use case: Input Actual Image Folder Path

3.3.3.6.7 Add Manual Compared Images

Use case specification

UC ID and Name:	**UC-43.0 Add Manual Compared Images**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user input single image pair.		
Preconditions:	**PRE-43.1** UI Comparison Module is opened.		
Post-conditions:	**POST-43.1** N/A.		
Normal Flow:	**43.0 Add Manual Compared Images** 1. User clicks [Add] button in datagrid. 2. System appends a new row to datagrid with [Expected Image] button , [Actual Image] button, [Image Viewer] button and [Delete] button. 3. User clicks [Expected Image] button, then dialog file chooser is showed. 4. User selects valid image, then clicks [Open] button. 5. System fills path of selected image is filled to [Expected		

	Image] column of current row. 6. User clicks [Actual Image] button, then dialog file chooser is showed. 7. User selects valid image, then clicks [Open] button. 8. System fills path of selected image is filled to [Actual Image] column of current row.
Alternative Flows:	**43.1 Add Manual Compared Images** 1. Do the same steps from 1-2 of normal flow. 2. User do nothing after that.
Exceptions:	**43.0-E1 – Select image doesn't have access permission.** System displays error message (You don't have permission to open this file. Contact the file owner or an administrator to obtain permission.)
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 54. Use case: Add Manual Compared Images

3.3.3.6.8 Clear All Images

Use case specification

UC ID and Name:	**UC-44.0 Clear All Images**		
Version	**1.1**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user clear all image pairs are inputted.		
Preconditions:	**PRE-44.1** UI Comparison Module is opened and datagrid contains at least one image pair.		
Post-conditions:	**POST-44.1** N/A.		
Normal Flow:	**44.0 Clear All Images** 1. User clicks [Clear All] button. 2. System displays confirm dialog with message [Are you sure you want to clear all data inputted?], [Yes] button and [No] button. 3. User clicks [Yes] button. 4. System deletes all image pairs in datagrid and disables [Compare] button, [Clear All] button.		
Alternative Flows:	**44.1 Clear All Images** 1. Do the same steps from 1-2 of normal flow 2. User clicks [No] button. 3. System closes confirm dialog and nothing change.		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		
Business Rules:	B23, B24, B28, B29		
Other Information:	N/A.		
Assumptions:	N/A.		

Table 55. Use case: Clear All Images

3.3.3.6.9 Compare All Images

Use case specification

UC ID and Name:	**UC-45.0 Compare All Images**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user compares all image pairs inputted.		
Preconditions:	**PRE-45.1** UI Comparison Module is opened and datagrid contains at least one image pair.		
Post-conditions:	**POST-45.1** N/A.		
Normal Flow:	**45.0 Compare All Images** 1. User clicks [Compare] button. 2. System displays the processing dialog indicating the image handling process. 3. System fills the results after comparison in columns of datagrid as follows: - Log is filled to [Result log] column with messages: + "Images are matched." + "Images are not matched, please review result image." + "Images have different widths." + "Images have different heights." + "Images have different heights. Images have different widths." - Path of image result is filled to [Result Image] column.		
Alternative Flows:	N/A		
Exceptions:	**45.0-E1 – Expected image or Actual image is blank.** System fills error message to [Result Log] column: - "Expected image not found." - "Actual image not found." **45.0-E2 – Expected image and Actual image are blank.** System fills error message to [Result Log] column: - "Expected image not found. Actual image not found."		

Priority:	High
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 56. Use case: Compare All Images

3.3.3.6.10 View Image Pair In Image Viewer

Use case specification

UC ID and Name:	**UC-46.0 View Image Pair In Image Viewer**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allows users to observe details and visually compare image pair.		
Preconditions:	**PRE-46.1** UI Comparison Module is opened.		
Post-conditions:	**POST-46.1** Image Viewer is displayed.		
Normal Flow:	**46.0 View Image Pair in Image Viewer** 1. User inputs full image pairs then clicks [Compare] button. 2. System fills the result to datagrid. 3. User clicks [Image Viewer] button in any row of datagrid. 4. System displays Image Viewer window with 3 tabs: - [Result] tab: show image result after compared. - [Fade] tab: show 2 original images let user manual compare. - [Slider] tab: show 2 original images let user manual compare. **46.1 View Image Pair in Image Viewer** 1. User inputs full image pairs then do not execute compare. 2. User clicks [Image Viewer] button in any row of datagrid.		

	3. System displays Image Viewer window with 3 tabs: - [Result] tab: image result is blank. - [Fade] tab: show 2 original images let user manual compare. - [Slider] tab: show 2 original images let user manual compare. **46.2 View Image Pair in Image Viewer** 1. User inputs only [Expected Image] or [Actual Image]. 2. User clicks [Image Viewer] button in any row of datagrid. 3. System displays Image Viewer window with only tab: - [Result] tab: image result is blank.
Alternative Flows:	N/A.
Exceptions:	N/A.
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 57. Use case: View Image Pair In Image Viewer

3.3.3.6.11 Delete Image Pair

Use case specification

UC ID and Name:	**UC-47.0 Delete Image Pair**		
Version	**1.1**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allow user delete one or more image pair(s) inputted.		
Preconditions:	**PRE-47.1** UI Comparison Module is opened and datagrid contains at least one image pair.		
Post-conditions:	**POST-47.1** N/A.		

Normal Flow:	**47.0 Delete Image Pair** 1. User clicks [Delete] button. 2. System displays confirm dialog with message [Are you sure you want to delete row(s)?], [Yes] button and [No] button. 3. User clicks [Yes] button. 4. System deletes row contains image pairs in datagrid, if datagrid empty after delete image pairs, [Compare] button and [Clear All] button are disabled. **47.1 Delete multiple Image Pair(s)** 1. User press and hold [Ctrl] then clicks on more than one row in datagrid. 1. User clicks [Delete] button. 2. System displays confirm dialog with message [Are you sure you want to delete row(s)?], [Yes] button and [No] button. 3. User clicks [Yes] button. 4. System deletes row(s) contains image pairs in datagrid, if datagrid empty after delete image pairs, [Compare] button and [Clear All] button are disabled.
Alternative Flows:	**47.2 Delete Image Pair** 1. Do the same steps from 1-2 of normal flow 2. User clicks [No] button. 3. System displays confirm dialog and nothing change. **47.3 Delete multiple Image Pair(s)** 1. Do the same steps from 1-2 of normal flow 2. User clicks [No] button. 3. System displays confirm dialog and nothing change.
Exceptions:	N/A.
Priority:	Low
Frequency of Use:	Medium
Business Rules:	B23, B24, B28, B29
Other Information:	N/A.
Assumptions:	N/A.

Table 58. Use case: Delete Image Pair

3.3.3.7 Report

3.3.3.7.1 View Report

Use case specification

Use case ID	UC-48	**Version**	1.1
Use case name	View "Report"		
Created by	HauLV	**Date created**	24/03/2019
Actor	User		
Description	This function allows user to view "Report" screen		
Preconditions	**PRE-48.1** User has opened a project.		
Trigger	N/A.		
Post conditions	Open and active "Report" screen		
Main flows			
Step	**Actor**	**Action**	
1	User	Click "Report" button in toolbar	
2	System	Display "Report" screen	
Alternative flows			
Step	**Actor**	**Action**	
1.a	User	Click "Report" button in "Project Explorer"	
Exceptions	N/A.		
Priority	Low.		
Frequency of use	Medium.		
Business rules	B03, B10, B22		
Other inf.	N/A.		
Assumptions	N/A.		

Table 59. Use case: View Report

3.3.3.7.2 Export Report To HTML

Use case specification

Use case ID	UC-49	**Version**	1.0
Use case name	Export Report To HTML		
Created by	HauLV	**Date created**	24/03/2019
Actor	User.		
Description	This function allows user to export report table to HTML.		
Preconditions	User has opened “Report” screen		
Trigger	N/A.		
Post conditions	Export report and allow to save HTML file.		
Main flows			
Step	**Actor**	**Action**	
1	User	Click on “Export to HTML”.	
2	System	Allow to save HTML file	
Alternative flows			
Exceptions	N/A.		
Step	**Actor**	**Action**	
N/A.	N/A.	N/A.	
Exceptions	N/A.		
Priority	Low.		
Frequency of use	Medium.		
Business rules	B03, B10, B22		
Other inf.	N/A.		
Assumptions	N/A.		

Table 60. Use case: Export Report To HTML

3.3.3.7.3 Export Report To Excel

Use case specification

Use case ID	UC-50	**Version**	1.1
Use case name	Export Report To Excel		
Created by	HauLV	**Date created**	24/03/2019
Actor	User.		
Description	This function allows user to export report table to Excel.		
Preconditions	User has opened “Report” screen		
Trigger	N/A.		
Post conditions	Export report and allow to save Excel file.		
Main flows			
Step	**Actor**	**Action**	
1	User	Click on “Export to Excel”.	
2	System	Allow to save Excel file	
Alternative flows			
Step	**Actor**	**Action**	
N/A.	N/A.	N/A.	
Exceptions	N/A.		
Priority	Low.		
Frequency of use	Medium.		
Business rules	B03, B10, B22		
Other information	N/A.		
Assumptions	N/A.		

Table 61. Use case: Export Report To Excel

3.3.3.8 App Settings

3.3.3.8.1 Setting General Preferences

Use case specification

UC ID and Name:	**UC-51.0 Setting General Preferences**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allows users to customize the waiting time value and speed when run test cases to match the software tested		
Preconditions:	**PRE-51.1 User has opened TOBI system.**		
Post-conditions:	**POST-51.1** N/A.		
Normal Flow:	**51.0 Setting General Preferences** 1. System displays TOBI main screen. 2. User clicks SETTINGS menu on the menu bar, and clicks on Preferences. 3. System displays [System Preferences] dialog. 4. User clicks on tab [General] to setting value for wait time, speed when run test. 5. User can choose to change the waiting time and speed values. 6. User clicks [OK] button to save changed value to system.		
Alternative Flows:	**51.1 Setting General Preferences** 1. System displays TOBI main screen. 2. User clicks SETTINGS menu on the menu bar, and clicks on Preferences. 3. System displays [System Preferences] dialog. 4. User clicks on tab [General] to setting value for wait time, speed when run test. 5. User can choose to change the waiting time and speed values. 6. User clicks [Cancel] button to change nothing.		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		

Business Rules:	N/A.
Other Information:	N/A.
Assumptions:	N/A.

Table 62. Use case: Export Report To Excel

3.3.3.8.2 Setting Compare UI Preferences

Use case specification

UC ID and Name:	**UC-52.0 Setting Compare UI Preferences**		
Version	**1.1**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allows users to customize accuracy of UI comparison when show log after run test.		
Preconditions:	**PRE-52.1** User has already opened TOBI system.		
Post-conditions:	**POST-52.1** Accuracy is configured based on user preference.		
Normal Flow:	**52.0 Setting preferences Compare UI** 1. System displays TOBI main screen. 2. User clicks SETTINGS menu on the menu bar, and clicks on Preferences. 3. System displays [System Preferences] dialog. 4. User clicks on tab [General] to setting value for accuracy, optional values allow percentage after executing the UI comparison is accepted 5. User changes accuracy of UI comparison. 6. User clicks [OK] button to save changed value.		
Alternative Flows:	**52.1 Setting preferences Compare UI** 1. System displays TOBI main screen. 2. User clicks SETTINGS menu on the menu bar, and clicks on Preferences. 3. System displays [System Preferences] dialog. 4. User clicks on tab [General] to setting value for accuracy, optional values allow percentage after executing the UI comparison is accepted		

	5. User changes accuracy of UI comparison. 6. User clicks [Cancel] button to change nothing.
Exceptions:	N/A.
Priority:	Low
Frequency of Use:	Medium
Business Rules:	N/A.
Other Information:	N/A.
Assumptions:	N/A.

Table 63. Use case: Setting Compare UI Preferences

3.3.3.9 Other features

3.3.3.9.1 Reset Layout

Use case specification

UC ID and Name:	**UC-53.0 Reset Layout**		
Version	**1.2**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allows users to reset to tool's original layout		
Preconditions:	**PRE-53.1** User has already opened TOBI system.		
Post-conditions:	**POST-53.1** Tool's layout will be reset to normal.		
Normal Flow:	**53.0 Reset Layout** 1. System displays TOBI main screen. 2. User clicks SETTINGS menu on the menu bar, and clicks on Reset Layout. 3. System displays TOBI with original layout		
Alternative Flows:	N/A		
Exceptions:	N/A.		
Priority:	Low		

Frequency of Use:	Medium
Business Rules:	N/A.
Other Information:	N/A.
Assumptions:	User has messed up with tool's layout.

Table 64. Use case: Reset Layout

3.3.3.9.2 View About System

Use case specification

UC ID and Name:	**UC-54. View About System**		
Version	**1.1**		
Created by:	**TuPA**	Date Created:	**04/11/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Allows users to view about system.		
Preconditions:	**PRE-54.1** User has already opened TOBI system.		
Post-conditions:	**POST-54.1** About system dialog shows up.		
Normal Flow:	**54.0 View About System** 1. System displays TOBI main screen. 2. User clicks ABOUT menu on the menu bar. 3. System displays About system dialog.		
Alternative Flows:	N/A		
Exceptions:	N/A.		
Priority:	Low		
Frequency of Use:	Medium		
Business Rules:	N/A.		
Other Information:	N/A.		
Assumptions:	N/A.		

Table 65. Use case: View About System

3.3.3.9.3 Launch Apps

UC ID and Name:	**UC-55.0 Launch Apps**		
Version	**1.0**		
Created by:	**DongND**	Date Created:	**11/04/2019**
Primary Actor:	**User**	Secondary Actors:	N/A.
Description:	Input old and new tested application to launch when record and run test.		
Preconditions:	**PRE-55.1** User has opened TOBI system.		
Post-conditions:	**POST-55.1** When the normal flow completes successfully, user can input old and new tested application.		
Normal Flow:	**55.0 Launch Apps** 1. User click on “Launch App” button in toolbar 2. System displays a dialog 3. User browse old and new file path by click “Browse” button 4. User click “OK” button to confirm		
Alternative Flows:	**N/A**		
Exceptions	**N/A**		
Priority:	Low		
Frequency of Use:	Low		
Business Rules:	N/A		
Other Information:	N/A		
Assumptions:	Project has opened TOBI application.		

3.3. Non-functional Requirements

3.6.1. Usability

- User interface should be friendly, simple, logical and easy to use.
- All the texts and messages should be informed exactly and clearly.

3.6.2. Maintainability and Extensibility

- Source code must follow the coding convention in order to help developers to read, understand and find it easy to maintain or modify the code.
- Window application should be developed based on Object Oriented Programming paradigm in order to increase maintainability and extensibility.
- Window application uses MVVM model, modular software and component-based development.

3.6.3. Performance

- Delay time of CPU (from the moment that the screen is captured to the moment that censor boxes displays): 0.1 – 1 seconds. It is calculated by the function stopWatch() on Visual Studio.
- Benchmark: Base on delay time of CPU: x (seconds). If $x < 0.25$, using real-time. If $0.25 \leq x \leq 0.8$, using real-time and object tracking. If $x > 0.8$, using scan interval.
- Memory: Use 600MB – 700MB RAM when running.
- Hard disks: 300MB – 400MB (to store executed, config and library files)

3.4. Entity Relationship Diagram

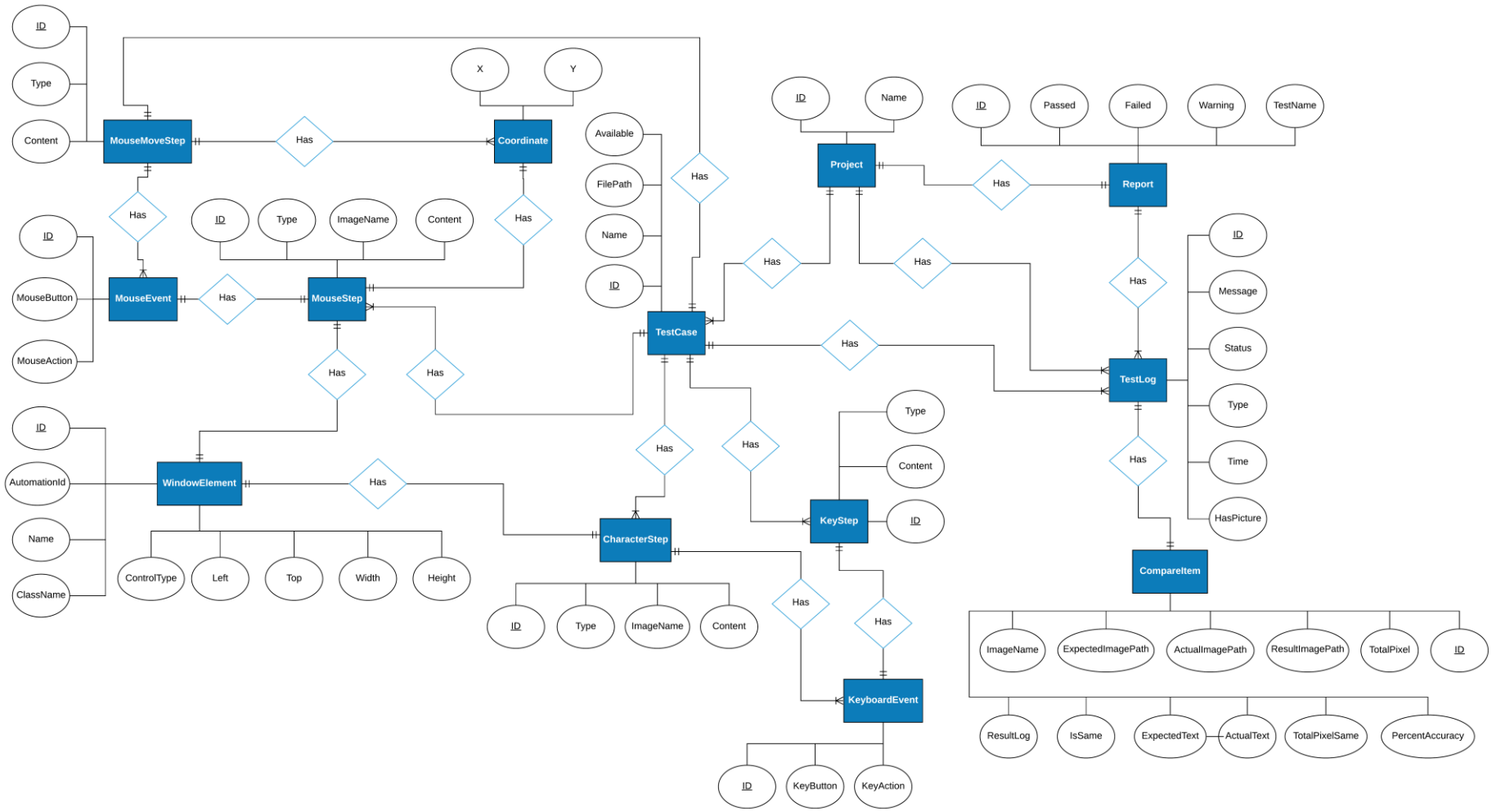


Figure 7. Entity Relationship Diagram

CHAPTER 4 – SOFTWARE DESIGN SPECIFICATIONS

4.1. Purpose

This chapter is to give the developer team an overview of what the system's architecture is, and how they should be implemented. This chapter consists of:

- ✓ Overview of system architecture
- ✓ Architecture layer design
- ✓ Data Structure Design
- ✓ Application of Tobi Automation Test.
- ✓ Detailed design

4.2. Overview of System Architecture

4.2.1 Diagram

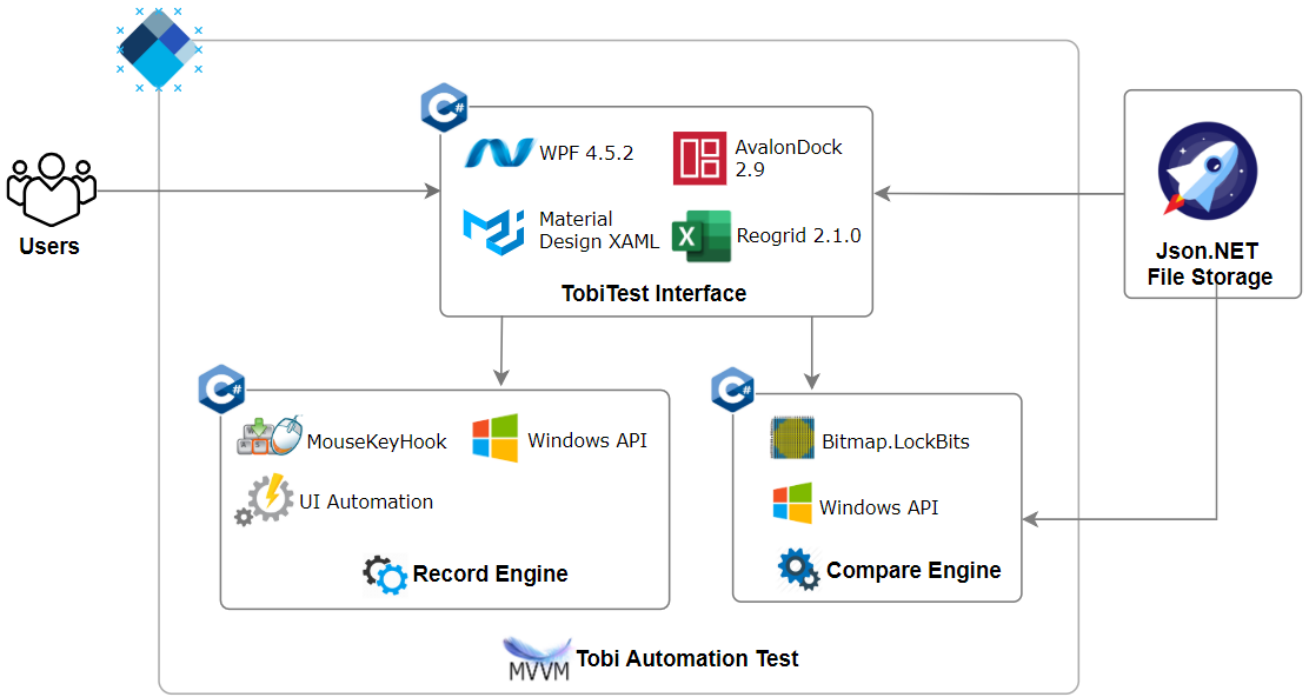


Figure 8. System architecture

4.2.2 System Architecture Explanation

<u>Interface:</u>

- Build user interfaces for Windows-based application using WPF 4.5.2, which takes advantage of WPF components and external UI Libraries & Toolkits (e.g AvalonDock, Reogrid, Material Design).

- Use XAML components as building blocks to create a consistent yet modern look-and-feel for the application.

File Storage:

- Storage: Use *.json format to store data, and NewtonSoft JSON to serialize/deserialize various data structures to files.

Record Engine:

- UI automation: Manipulate and capture detailed information about UI elements displayed on screen or in tested apps.
- Hooking: Capture low-level mouse and keyboard events sent from user inputs with the help of Windows API.

Compare Engine:

- Image Processing: Bitmaps.LockBits to lock rectangular portions of bitmaps and provide a temporary buffer and can be used to read or write pixel.

API:

- Microsoft UI Automation: Allows one to access, identify, and manipulate the user interface (UI) elements of another application.
- Windows API: Microsoft's core set of APIs to manipulate low-level hooking operations and bitmap functions.

4.2.3 Component Explanation

Tobi Automation Test is a useful tool built entirely on Windows Presentation Foundation framework and MVVM design pattern. It aims to provide an easily managed, highly customized and scalable system. This section will explain the function and mechanism of each unit in the system architecture design.

4.2.3.1 Interface Components

4.2.3.1.1 Windows Presentation Foundation

Windows Presentation Foundation (WPF) is a next-generation Presentation system for building Windows client applications with visually stunning user experiences. **We use WPF to build UI components, utilize its rich Windows API, and make highly scalable application based on its powerful framework.**

4.2.3.1.2 Material Design In XAML Toolkit

Material Design In XAML Toolkit can easily bring beautiful desktop applications to life, using a modern and popular design language. It is fully open source and one of the most popular GUI libraries for WPF. **We use this toolkit for building stunning UI components suitable for modern applications.**

4.2.3.1.3 AvalonDock

AvalonDock is a docking windows control for WPF that lets one create customizable layouts using a full featured window docking system similar to what is found in many popular integrated development environments (IDEs) such as Visual Studio. **We use AvalonDock to utilize its various themes and make IDE-like look and feel for Tobi Automation Test system.**

4.2.3.1.4 Reogrid

Reogrid is fast and powerful .NET spreadsheet component, and it supports data format, freeze, outline, formula calculation, chart, script execution and etc. It's compatible with Excel 2007 (.xlsx) format and working on .NET 3.5 (or client profile), WPF and Android platform. **We use Reogrid to create the report's spreadsheet section.**

4.2.3.2 Application Programming Interface (API)

4.2.3.2.1 Windows UI Automation

Windows UI Automation enables Windows applications to provide and consume programmatic information about user interfaces (UIs). It provides programmatic access to most UI elements on the desktop. It enables assistive technology products, such as screen readers, to provide information about the UI to end users and to manipulate the UI by means other than standard input.

4.2.3.2.2 Windows API

The **Windows API**, informally WinAPI, is Microsoft's core set of application programming interfaces (APIs) available in the Microsoft Windows operating systems. Almost all Windows programs interact with the Windows API. **We use Windows API to access low-level hook functionalities in order to handle mouse and keyboard events.**

4.2.3.3External Libraries

4.2.3.3.1 JSON.NET

JSON.NET is a popular high-performance JSON framework for .NET, which has flexible JSON serializer for converting between .NET objects and JSON. **We use JSON.NET in order to convert complex data structures to JSON files and vice versa.**

4.2.3.3.2 MouseKeyHook

MouseKeyHook allows you to tap keyboard and mouse, to detect and record their activity even when an application is inactive and runs in background. **We use MouseKeyHook** to capture low-level mouse and keyboard events sent from user inputs with the help of Windows API.

4.3. High-level Architectural Design

4.3.1. Architecture Layers Design

4.3.1.1 Diagram

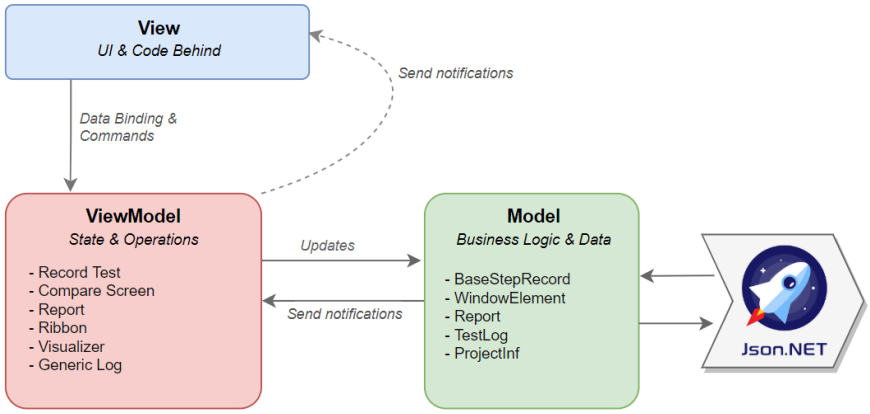


Figure 9. TobiTest Layer Design

4.3.1.2 Project Structure

No	Directory/File	Description
1	Assets/	Contain images and other important assets for building user interfaces.
2	Config/	Contain project-level settings for preferences.
3	Converter/	Contain mappings to/from logical values in the view models and presented values in the user interfaces.
4	CustomDialog/	Contain configuration classes to create and access Dialog from view models.
5	Helper/	Contain helper functions for easy manipulation of data and user inputs.
6	Hook/	Contain native hooking functions, structs, enums to handle mouse and keyboard inputs.
7	Library/	Contain hooking library which utilizes Windows API.
8	Model/	Contain data structure models.
9	ProjectViewModel/	Contain view models for project manipulation features.
10	Selector/	Contain selectors to display template based on tab being clicked.
11	UIAccess/	Contain functions to access and manipulate UI elements .
12	UserControls/	Contain user controls that can be used as templates in various places.
13	View/	Contain UI elements and building blocks for interfaces of TobiTest application.
14	ViewModel/	Contain view models that store states and operations of UI elements.
15	ViewModelDialog/	Contain view models storing states and operations of Dialog UI Elements.
16	CompareEngine/Engine/	Contain functions to handle image comparison.
17	App.xaml	Contain instructions for starting the desired Window & Page with specified resources.
18	App.config	Contains all environment variables and configurations.
19	app.ico	Contain application icon.
20	packages.config	Contain id, version, and target framework of all necessary packages.
21	Settings.cs	Contain functions to handle specific events on the Settings class.

Table 66. Application Layers Design

4.3.3 User Interface Design

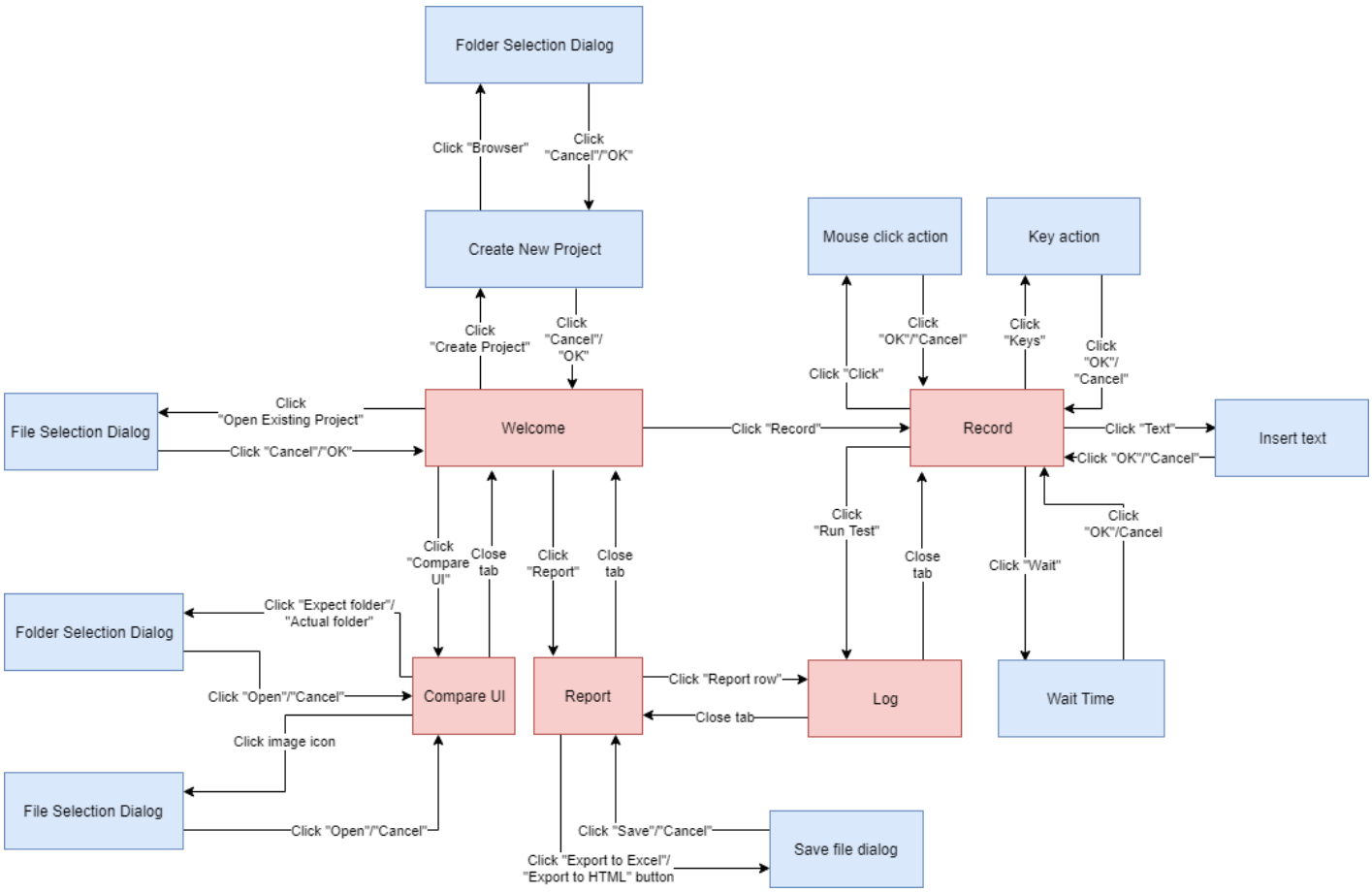
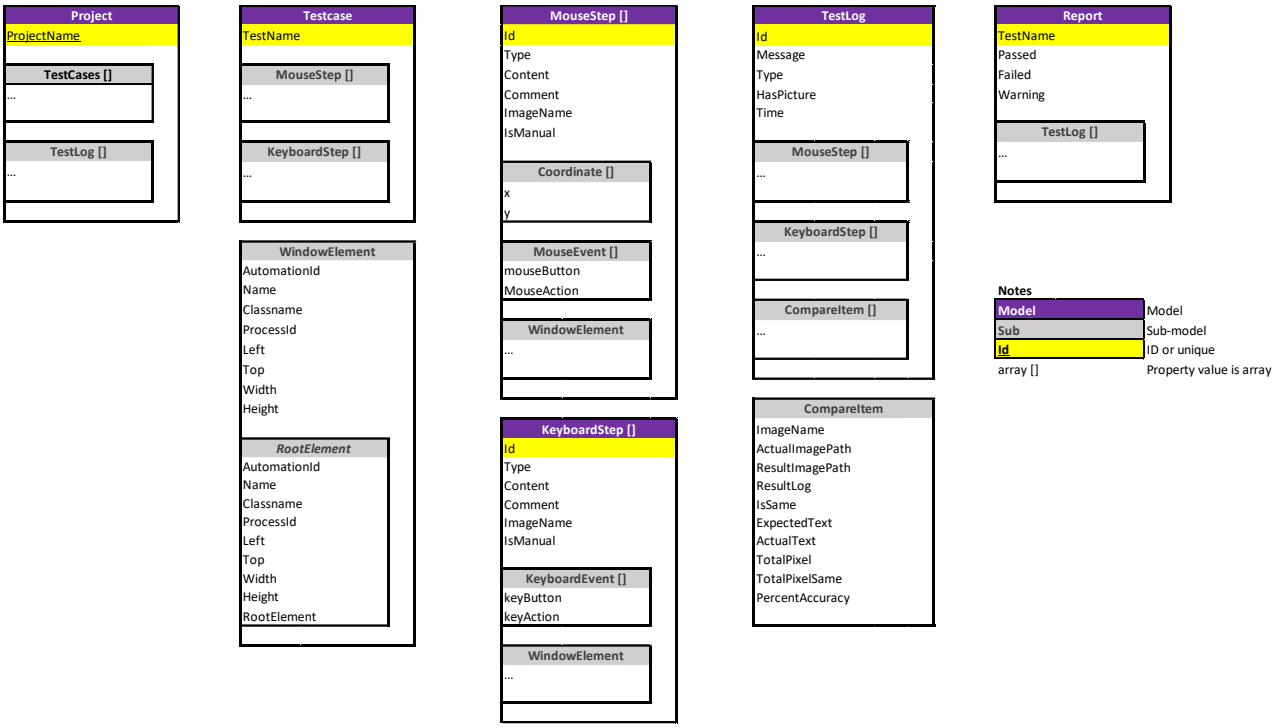


Figure 10. User Interface Design

4.4 Data Structure Design

4.4.1 Diagram

Figure 11. Data Structure Design



4.4.2 Data Dictionary

4.4.2.1Entities

No	Model Name	Description
1	Project	Information of test project user create
2	Testcase	Information of recorded test
3	MouseStep	Information of recorded mouse step
4	KeyboardStep	Information of recorded keyboard step
5	WindowElement	Information of window element
6	TestLog	Information of test log after run test
7	CompareItem	Information of compare image
8	Report	Information of project report

Table 67. Entity Table

4.4.2.2 Attributes

Entity Name	Field Name	Data Type	Description	Not Null ?
Project	ProjectName	String	Name of test project	Yes
Testcase	TestName	Integer	Name of test case	Yes

MouseStep	Id		Integer	Id of mouse step	Yes
	Type		String	Type of step	Yes
	Content		String	Description about step	Yes
	Comment		String	Comment about step	No
	ImageName		String	Name of Image	No
	IsManual		Boolean	Indicate step is record or manual	Yes
	Coordinate	x	Integer	x coordinate of mouse step	Yes
		y	Integer	y coordinate of mouse step	Yes
	MouseEvent	MouseButton	String	Mouse button of event	No
		MouseAction	String	Mouse action of event	No
Keyboard Step	Id		Integer	Id of mouse step	Yes
	Type		String	Type of step	Yes
	Content		String	Description about step	Yes
	Comment		String	Comment about step	No
	ImageName		String	Name of Image	No
	IsManual		Boolean	Indicate step is record or manual	Yes
	KeyboardEvent	KeyButton	String	Key button of event	Yes
		KeyAction	String	Key action of event	Yes
WindowElement	AutomationId		String	Id of UI element	No
	Name		String	Name of UI element	Yes
	Classname		String	Class name of UI element	Yes
	ProcessId		Integer	Id of process	No
	x		Integer	x coordinate of UI element	Yes
	y		Integer	y coordinate of UI element	Yes
	Width		Integer	Width of UI element	Yes
	Height		Integer	Height of UI element	Yes
TestLog	Id		Integer	Id of log	Yes
	Message		String	Message of log	Yes
	Type		String	Type of log	Yes
	HasPicture		Boolean	Indicate log has picture or not	Yes
	Time		DateTime	Time when create log	Yes
Compare Item	ImageName		String	Name of image	Yes
	ActualImagePath		String	Path of actual image	Yes
	ResultImagePath		String	Path of result image	Yes
	Resultlog		String	Result log of compare image	Yes
	IsSame		Boolean	Compare result	Yes
	ExpectedText		String	Path of image when record	No
	ActualText		String	Actual text of compare item	No
	TotalPixel		Integer	Total pixel of image	Yes
	TotalPixelSame		Integer	Total same pixel when compare image	Yes
	PercentAccuracy		Double	Accuracy percentage when comparing image	Yes

Table 68.Attributes Table

4.5 Detailed System Design

4.5.1 Class Diagram

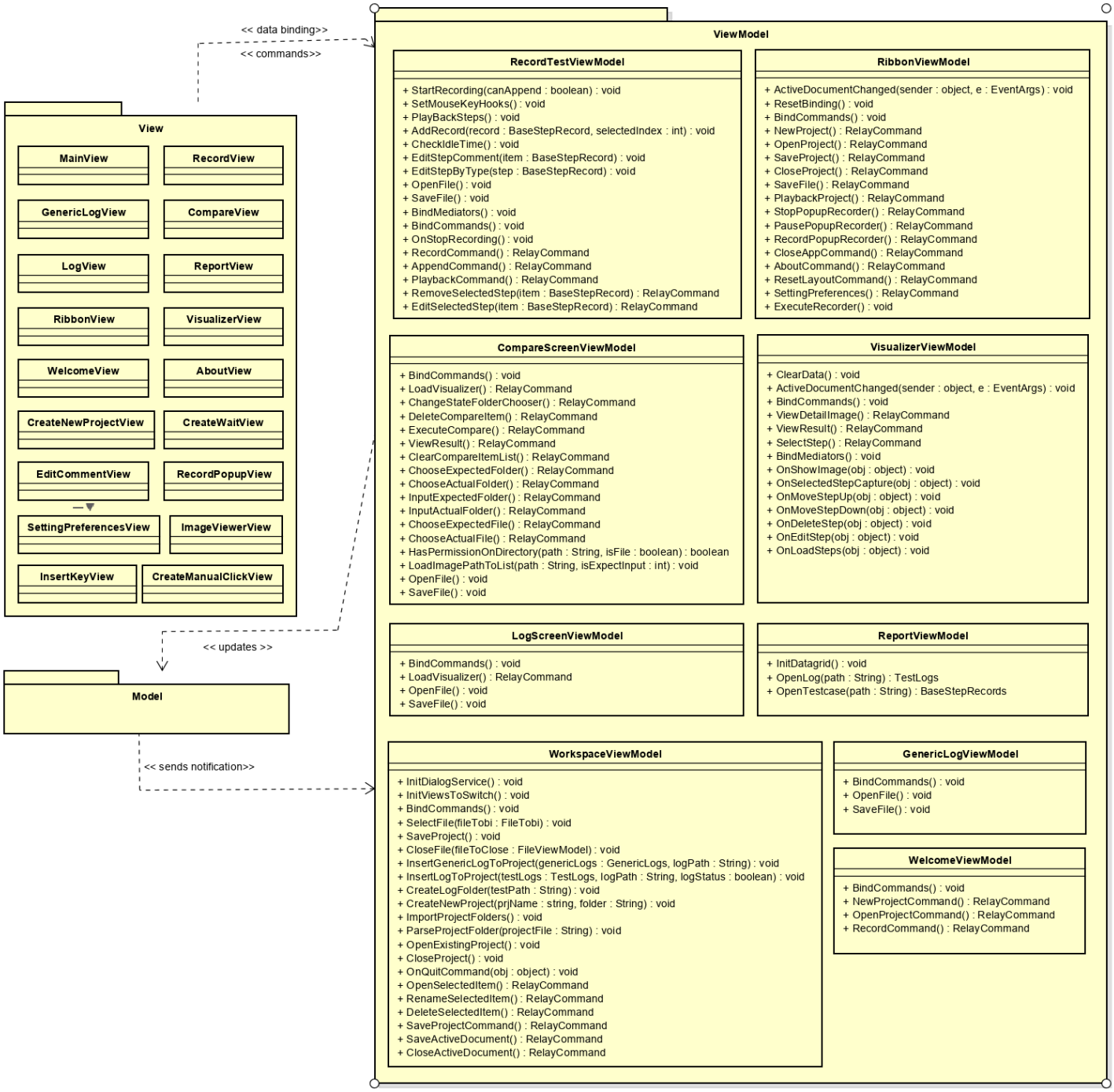


Figure 12. Class Diagram for TOBI

4.5.2 Class Specifications

4.5.2.1 Record Test

RecordTestViewModel			
Attributes			
No	**Name**	**Type**	**Description**
1	*StepList*	List<BaseStepRecord>	*Contain all steps for record file*
2	*SelectedStep*	BaseStepRecord	*The step being clicked on*
3	*highlightRectangle*	HighLightRectangle	*The captured portion of screen*

Operation				
No	**Methods**			
1	**StartRecording()**			
	Return type	void		
	Description	Start capturing user inputs and add steps to test case		
	Parameters	*Parameter*	*Type*	*Description*
		canAppend	Boolean	Add new steps to the end of current test case (canAppend=true)
2	**SetMouseKeyHooks()**			
	Return type	void		
	Description	Subscribe to mouse and keyboard event handlers		
3	**PlaybackSteps()**			
	Return type	void		
	Description	Execute steps in current testcase		
	Parameters	*Parameter*	*Type*	*Description*
		StepList	List<BaseStepRecord>	Steps to playback
4	**AddRecord()**			
	Return type	void		
	Description	Add recorded step to the test case		
	Parameters	*Parameter*	*Type*	*Description*
		record	BaseStepRecord	The record to be added
		selectedIndex	Int	The index where record will be added
5	**CheckIdleTime()**			
	Return type	Void		
	Description	Check idle time to automatically append wait step or highlight UI element		
	Parameters	*Parameters*	*Type*	*Description*
		timer	DispatcherTimer	Check idle time to add wait step
6	**EditStepComment ()**			
	Return type	void		
	Description	Edit step comment		
	Parameters	*Parameters*	*Type*	*Description*
		item	BaseStepRecord	The selected step to edit comment
7	**EditStepByType** ()			
	Return type	void		
	Description	Edit step by type		
	Parameters	*Parameters*	*Type*	*Description*
		step	BaseStepRecord	The selected step to edit by type
8	**OpenFile** ()			
	Return type	void		
	Description	Parse test case data from file		

	Parameters	*Parameters*	*Type*	*Description*
		file	FileViewModel	File to be opened
9	**SaveFile**()			
	Return type	Void		
	Description	Save test case data to file		
	Parameters	*Parameters*	*Type*	*Description*
		file	FileViewModel	File to be saved
10	**BindMediators** ()			
	Return type	void		
	Description	Bind all mediators		
11	**BindCommands** ()			
	Return type	void		
	Description	Bind all commands		
12	**OnStopRecording** ()			
	Return type	void		
	Description	Stop recording function		
13	**RecordCommand** ()			
	Return type	RelayCommand		
	Description	Start recording function (minimize popup recorder)		
	Parameters	*Parameters*	*Type*	*Description*
		canAppend	Boolean	Add new steps to the end of current test case (canAppend=false)
14	**AppendCommand** ()			
	Return type	RelayCommand		
	Description	Start appending function (minimize popup recorder)		
	Parameters	*Parameters*	*Type*	*Description*
		canAppend	Boolean	Add new steps to the end of current test case (canAppend=true)
15	**PlaybackCommand** ()			
	Return type	RelayCommand		
	Description	Playback all steps		
	Parameters	*Parameters*	*Type*	*Description*
		StepList	List<BaseStepRecord>	Contains steps to playback
16	**RemoveSelectedStep** ()			
	Return type	RelayCommand		
	Description	Remove selected step		
	Parameters	*Parameters*	*Type*	*Description*
		item	BaseStepRecord	The selected item to be removed
17	**EditSelectedStep** ()			
	Return type	Void		
	Description	Edit selected step		
	Parameters	*Parameters*	*Type*	*Description*

		item	BaseStepRecord	The selected item to be edited

Table 69. Record Test

4.5.2.2 Ribbon

RibbonViewModel				
Attributes				
No	**Name**	**Type**	**Description**	
1	*CanPause*	Boolean	*Set state for pause button (popup recorder)*	
2	*CanRecord*	Boolean	*Set state for record button (popup recorder)*	
3	*ActiveRecorder*	Boolean	*Set state for append button on toolbar*	
4	*ProjectAvailable*	Boolean	*Disable relevant buttons when there's no project.*	
Operation				
No	**Methods**			
1	**ActiveDocumentChanged ()**			
	Return type	void		
	Description	Set state for active recorder		
	Parameters	*Parameter*	*Type*	*Description*
		sender	object	Event handler
		e	EventArgs	Actual event being sent
2	**ResetBinding()**			
	Return type	void		
	Description	Reset all binding variables		
	Parameters	*Parameter*	*Type*	*Description*
		canPause	Boolean	Reset pause state to true
		canRecord	Boolean	Reset record state to false
3	**BindCommands()**			
	Return type	void		
	Description	Bind all commands		
4	**NewProject()**			
	Return type	RelayCommand		
	Description	Create new project		
	Parameters	*Parameter*	*Type*	*Description*
		location	String	Location to create project
		projectName	String	Name of the project
5	**OpenProject ()**			
	Return type	RelayCommand		
	Description	Open existing project		
	Parameters	*Parameter*	*Type*	*Description*
		location	String	Location to open project
6	**SaveProject ()**			

	Return type	RelayCommand		
	Description	Save current project		
	Parameters	*Parameter*	*Type*	*Description*
		location	String	Location to save project
7	**CloseProject** ()			
	Return type	RelayCommand		
	Description	Close current project		
	Parameters	*Parameter*	*Type*	*Description*
		Files	List<FileViewModel>	Contains all opened files
9	**SaveFile**()			
	Return type	RelayCommand		
	Description	Save current file		
	Parameters	*Parameter*	*Type*	*Description*
		File	FileViewModel	File to be saved
10	**PlaybackProject** ()			
	Return type	RelayCommand		
	Description	Play back all testcases		
	Parameters	*Parameter*	*Type*	*Description*
		logPath	String	Location to save logs
		playAll	Boolean	Indicates whether to playback project or normal testcases
11	**StopPopupRecorder** ()			
	Return type	RelayCommand		
	Description	Stop popup recorder		
	Parameters	*Parameter*	*Type*	*Description*
		win	Window	Close popup recorder
		recorder	RecordTestViewModel	Recorder of current selected test case
12	**PausePopupRecorder** ()			
	Return type	RelayCommand		
	Description	Pause popup recorder		
	Parameters	*Parameter*	*Type*	*Description*
		canPause	Boolean	Set state for pause button
		canRecord	Boolean	Set state for record button
13	**RecordPopupRecorder** ()			
	Return type	RelayCommand		
	Description	Start recording from popup recorder		
14	**CloseAppCommand** ()			
	Return type	RelayCommand		
	Description	Close app		
	Parameters	*Parameter*	*Type*	*Description*

		win	Window	Window instance to close current app.
15	**AboutCommand** ()			
	Return type	RelayCommand		
	Description	Open about system		
	Parameters	*Parameter*	*Type*	*Description*
		N/A.	N/A.	N/A.
16	**ResetLayoutCommand** ()			
	Return type	RelayCommand		
	Description	Reset to original layout		
	Parameters	*Parameter*	*Type*	*Description*
		N/A.	N/A.	N/A.
17	**SettingPreferences** ()			
	Return type	RelayCommand		
	Description	Open setting preferences dialog		
	Parameters	*Parameter*	*Type*	*Description*
		result	boolean	Check whether to save setting or not.
18	**ExecuteRecorder** ()			
	Return type	Void		
	Description	Execute recorder function		
	Parameters	*Parameter*	*Type*	*Description*
		Recorder	RecordTestViewModel	Recorder of current selected test case

Table 70. Ribbon

4.5.2.3 Compare Screen

CompareScreenViewModel			
Attributes			
No	**Name**	**Type**	**Description**
1	*SelectedItem*	CompareItem	*Selected item in a row in compare screen*
2	*CompareItemList*	List<CompareItem>	*List of all compare items (pair of images)*
3	*ExpectedFolder*	String	*Folder contains expected images*
4	*ActualFolder*	String	*Folder contains actual captured images*
Operation			
No	**Methods**		
1	**BindCommands()**		
	Return type	void	
	Description	Bind all commands	
	Parameters	*Parameters*	*Type*	*Description*
		N/A.	N/A.	N/A.
2	**LoadVisualizer()**		
	Return type	RelayCommand	
	Description	Show images in visualizer	
	Parameters	*Parameter*	*Type*	*Description*
		gridView	DataGrid	Table contains all image pair

3	**ChangeStateFolderChooser()**			
	Return type	RelayCommand		
	Description	Reset folder chooser state		
	Parameters	*Parameters*	*Type*	*Description*
		ExpectedFolder	String	Path to folder containing expected images
		ActualFolder	String	Path to folder containing actual images
4	**DeleteCompareItem()**			
	Return type	RelayCommand		
	Description	Delete compare item (image pair)		
	Parameters	*Parameters*	*Type*	*Description*
		selectedList	List<CompareItem>	List of compare item to be deleted
5	**ExecuteCompare()**			
	Return type	RelayCommand		
	Description	Start executing compare all items		
	Parameters	*Parameters*	*Type*	*Description*
		outputPath	String	Folder contains output images
6	**ViewResult()**			
	Return type	RelayCommand		
	Description	View result		
	Parameters	*Parameters*	*Type*	*Description*
		outputPath	String	Folder contains output images
7	**ClearCompareItemList**()			
	Return type	RelayCommand		
	Description	Clear all compare items		
	Parameters	*Parameters*	*Type*	*Description*
		deletePath	String	Path to the comparison item to be deleted
9	**ChooseExpectedFolder** ()			
	Return type	RelayCommand		
	Description	Choose expected folder		
	Parameters	*Parameters*	*Type*	*Description*
		ExpectedFolder	String	Path to expected images
10	**ChooseActualFolder** ()			
	Return type	RelayCommand		
	Description	Choose actual folder		
	Parameters	*Parameters*	*Type*	*Description*
		ActualFolder	String	Path to actual images
11	**InputExpectedFolder**()			
	Return type	RelayCommand		

	Description	Input expected folder		
	Parameters	*Parameters*	*Type*	*Description*
		path	String	Path to expected images
12	**InputActualFolder**()			
	Return type	RelayCommand		
	Description	Input actual folder		
	Parameters	*Parameters*	*Type*	*Description*
		path	String	Path to actual folder
13	**ChooseExpectedFile**()			
	Return type	RelayCommand		
	Description	Choose expected file		
	Parameters	*Parameters*	*Type*	*Description*
		path	String	Path to expected image
14	**ChooseActualFile**()			
	Return type	RelayCommand		
	Description	Choose actual file		
	Parameters	*Parameters*	*Type*	*Description*
		path	String	Path to actual image
15	**HasPermissionOnDirectory**()			
	Return type	Boolean		
	Description	Check directory permission		
	Parameters	*Parameter*	*Type*	*Description*
		path	String	Path to the selected folder
		isValid	Boolean	Check if the folder is accessible
16	**LoadImagePathToList**()			
	Return type	void		
	Description	Load image path to list		
	Parameters	*Parameter*	*Type*	*Description*
		path	String	Path to the selected image
		isExpectInput	Boolean	Check if the image is expected input
17	**OpenFile** ()			
	Return type	void		
	Description	Parse compare data from file		
	Parameters	*Parameters*	*Type*	*Description*
		file	FileViewModel	File to be opened
18	**SaveFile** ()			
	Return type	Void		
	Description	Save compare data to file		
	Parameters	*Parameters*	*Type*	*Description*
		file	FileViewModel	File to be saved

Table 71. Compare Screen

4.5.2.4 Visualizer

VisualizerViewModel			
Attributes			
No	**Name**	**Type**	**Description**
1	*ActualImageVisibility*	Visibility	*Show actual image exist.*
2	*ExpectedImageSource*	String	*Store path of expected image of selected image pair*
3	*ActualImageSource*	String	*Store path of actual image of selected image pair*
4	*ResultImageSource*	String	*Store path of result image of selected image pair*
5	*FilePathResultImage*	String	*Store path of result image of selected image pair*
6	*ResultLog*	String	*Show log warning when actual image or result image not exist.*
7	*SelectedStepCapture*	BaseStepRecord	*Store current selected step.*
8	*FilePathStepCapture*	String	*Store file path of list image captured*
9	*StepCaptures*	List<BaseStepRecord>	*Store and highlight Interacted UI Element recorded.*
10	*VisualizerCompare*	Visibility	*Show visualizer with image pair mode*
11	*VisualizerRecord*	Visibility	*Show visualizer with list images mode*

Operation				
No	**Methods**			
1	**ClearData ()**			
	Return type	void		
	Description	Set all variables to their original values		
	Parameters	*Parameter*	*Type*	*Description*
		N/A.	N/A.	N/A
2	**ActiveDocumentChanged()**			
	Return type	void		
	Description	Set visualizer layout depending on active document		
	Parameters	*Parameters*	*Type*	*Description*
		sender	object	N/A.
		e	EventArgs	N/A
3	**BindCommands()**			
	Return type	void		
	Description	Bind all commands		
	Parameters	*Parameters*	*Type*	*Description*
		N/A.	N/A.	N/A.
4	**ViewDetailImage()**			
	Return type	RelayCommand		
	Description	View image detail		
	Parameters	*Parameters*	*Type*	*Description*
		imagePath	BitmapImage	Get source of image and show on imageViewer
5	**ViewResult()**			

	Return type	RelayCommand		
	Description	View result		
	Parameters	*Parameters*	*Type*	*Description*
		imageSource	String	Path of image to show/hide
6	**SelectStep()**			
	Return type	RelayCommand		
	Description	Select step		
	Parameters	*Parameters*	*Type*	*Description*
		P	ListBox	Listbox that has been setting scroll by selected item
7	**BindMediators** ()			
	Return type	void		
	Description	Bind all mediators		
	Parameters	*Parameters*	*Type*	*Description*
		N/A.	N/A.	N/A.
8	**OnShowImage** ()			
	Return type	void		
	Description			
	Parameters	*Parameters*	*Type*	*Description*
		Obj	Object	Selected item that display image in visualizer on CompareUI or Log mode.
9	**OnSelectedStepCapture** ()			
	Return type	void		
	Parameters	*Parameters*	*Type*	*Description*
		Obj	Object	Set selected image item by selected BaseStepRecord
10	**OnMoveStepUp** ()			
	Return type	void		
	Parameters	*Parameters*	*Type*	*Description*
		Obj	Object	Move image item up by index of BaseStepRecord
11	**OnMoveStepDown** ()			
	Return type	void		
	Parameters	*Parameters*	*Type*	*Description*
		Obj	Object	Move image item down by index of BaseStepRecord
12	**OnDeleteStep** ()			
	Return type	void		
	Parameters	*Parameters*	*Type*	*Description*

		Obj	Object	Delete image item by BaseStepRecord
13	**OnEditStep** ()			
	Return type	void		
	Parameters	*Parameters*	*Type*	*Description*
		Obj	Object	Get edited and rerender BaseStepRecord
14	**OnLoadSteps** ()			
	Return type	void		
	Parameters	*Parameters*	*Type*	*Description*
		Obj	Object	Get list BaseStepRecord from Record Document

Table 72. Visualizer

4.5.2.5 Log Screen

LogScreenViewModel				
Attributes				
No	**Name**	**Type**	**Description**	
1	*LogList*	*List<TestLog>*	*List of steps in log file*	
2	*SelectedLog*	*TestLog*	*Log step being selected*	
Operation				
No	**Methods**			
1	**BindCommands()**			
	Return type	void		
	Description	Bind all commands		
	Parameters	*Parameters*	*Type*	*Description*
		N/A.	N/A.	N/A.
2	**LoadVisualizer()**			
	Return type	RelayCommand		
	Description	Show image in visualizer		
	Parameters	*Parameters*	*Type*	*Description*
		SelectedLog	TestLog	The log step being selected
3	**OpenFile()**			
	Return type	void		
	Description	Parse log data from file		
	Parameters	*Parameters*	*Type*	*Description*
		file	FileViewModel	Log file to be opened
4	**SaveFile()**			
	Return type	Void		
	Description	Save log data to file		
	Parameters	*Parameters*	*Type*	*Description*
		file	FileViewModel	Log file to be saved

Table 73. Log Screen

4.5.2.6 Report

ReportViewModel				
Attributes				
No	**Name**	**Type**	**Description**	
1	*DataReport*	*List<Report>*	*List of report data*	
Operation				
No	**Methods**			
1	**InitDatagrid ()**			
	Return type	void		
	Description	Init data grid		
	Parameters	*Parameters*	*Type*	*Description*
		reports	List<Report>	List of reports to be inited
2	**OpenLog ()**			
	Return type	List<TestLog>		
	Description	Open log file		
	Parameters	*Parameters*	*Type*	*Description*
		path	String	Path to the log file
3	**OpenTestcase ()**			
	Return type	List<BaseStepRecord>		
	Description	Open test case		
	Parameters	*Parameter*	*Type*	*Description*
		path	String	Path to open test case
4	**SaveFile ()**			
	Return type	void		
	Description	Save report data to file		
	Parameters	*Parameters*	*Type*	*Description*
		file	FileViewModel	Report file to be opened

Table 74. Report

4.5.2.7 Generic Log

GenericLogViewModel				
Attributes				
No	**Name**	**Type**	**Description**	
1	*GenericLogList*	List<GenericLog>	*List of generic logs*	
Operation				
No	**Methods**			
1	**BindCommands ()**			
	Return type	void		
	Description	Bind all commands		
	Parameters	*Parameters*	*Type*	*Description*
		N/A.	N/A.	N/A.
2	**OpenFile()**			

	Return type	void		
	Description	Parse generic log data from file		
	Parameters	*Parameters*	*Type*	*Description*
		file	FileViewModel	Generic log file to be opened
3	**SaveFile ()**			
	Return type	void		
	Description	Save generic log data to file		
	Parameters	*Parameters*	*Type*	*Description*
		file	FileViewModel	Generic log file to be saved

Table 75. Generic Log

4.5.2.8 Welcome

WelcomeViewModel				
Attributes				
No	**Name**	**Type**	**Description**	
1	*Version*	String	*Version to display on welcome screen*	
Operation				
No	**Methods**			
1	**BindCommands ()**			
	Return type	void		
	Description	Bind all commands		
	Parameters	*Parameters*	*Type*	*Description*
		N/A.	N/A.	N/A.
2	**NewProjectCommand ()**			
	Return type	RelayCommand		
	Description	Create new project		
	Parameters	*Parameters*	*Type*	*Description*
		projectName	String	Name of the project
		location	String	Location to create project
3	**OpenProjectCommand ()**			
	Return type	RelayCommand		
	Description	Open existing project		
	Parameters	*Parameters*	*Type*	*Description*
		location	String	Location to open project
3	**RecordCommand ()**			
	Return type	RelayCommand		
	Description	Start recording feature		
	Parameters	*Parameters*	*Type*	*Description*
		recorder	RecordTestViewModel	Recorder of the current test case

Table 76. Welcome

4.5.2.9 Workspace

WorkspaceViewModel			
Attributes			
No	**Name**	**Type**	**Description**
1	*Series*	SeriesCollection	*Series to show chart*
2	*RibbonBar*	BaseViewModel	*Data Context to show toolbar*
3	*VisualizerPanel*	BaseViewModel	*Data Context to handle test visualizer*
4	*MainWindowState*	WindowState	*Set state of window (Maximized, Minimized)*
5	*Project*	List<FolderTobi>	*Store all relevant information about current project*
6	*ActiveDocument*	FileViewModel	*The current selected document tab*
7	*Files*	List<FileViewModel>	*Contains all opened files*
8	*SelectedTreeItem*	object	*Selected item in project explorer tree view*
9	*PlayAll*	Boolean	*Check whether to play all test cases*
10	*ribbonVM*	RibbonViewModel	*ViewModel to handle toolbar operation*
11	*visualizerVM*	VisualizerViewModel	*ViewModel to handle operations relevant to test visualizer*

Operation				
No	**Methods**			
1	**InitDialogService ()**			
	Return type	void		
	Description	Init dialog service		
	Parameters	*Parameter*	*Type*	*Description*
		N/A.	N/A.	N/A.
2	**InitViewsToSwitch ()**			
	Return type	void		
	Description	Init default views to switch later		
	Parameters	*Parameter*	*Type*	*Description*
		ribbonBar	BaseViewModel	Initialize ribbon view model
		visualizerPanel	BaseViewModel	Initialize visualizer view model
3	**BindCommands ()**			
	Return type	void		
	Description	Bind all commands		
	Parameters	*Parameter*	*Type*	*Description*
		N/A.	N/A.	N/A.
4	**SelectFile ()**			
	Return type	void		
	Description	Select file to open based on its type		
	Parameters	*Parameter*	*Type*	*Description*
		fileTobi	FileTobi	File from project explorer tree to open
5	**SaveProject()**			
	Return type	void		
	Description	Save the project		
	Parameters	*Parameter*	*Type*	*Description*

		files	List<FileViewModel>	List of opened files to save
6	**CloseFile ()**			
	Return type	void		
	Description	Close current file		
	Parameters	*Parameter*	*Type*	*Description*
		fileToClose	FileViewModel	File to close
7	**InsertGenericLogToProject** ()			
	Return type	void		
	Description	Insert generic log to project		
	Parameters	*Parameter*	*Type*	*Description*
		genericLogs	List<GenericLog>	List of project logs to add to project
		logPath	String	Path to log file
8	**InsertLogToProject** ()			
	Return type	void		
	Description	Insert log file to project		
	Parameters	*Parameter*	*Type*	*Description*
		testLogs	List<TestLog>	List of test logs to insert to projecct
9	**CreateLogFolder** ()			
	Return type	void		
	Description	Create log folder		
	Parameters	*Parameter*	*Type*	*Description*
		testPath	String	Path to create log folder
10	**CreateNewProject** ()			
	Return type	void		
	Description	Create new project		
	Parameters	*Parameter*	*Type*	*Description*
		prjName	String	Name of the project
		folder	String	Path to the project
11	**ImportProjectFolders**()			
	Return type	void		
	Description	Import all project folders		
	Parameters	*Parameter*	*Type*	*Description*
		Project	List<FolderTobi>	Information about project items
12	**ParseProjectFolder** ()			
	Return type	void		
	Description	Parse project folder from project file		
	Parameters	*Parameter*	*Type*	*Description*
		projectFile	String	Path to project file
13	**OpenExistingProject** ()			
	Return type	Void		
	Description	Open existing project		
	Parameters	*Parameter*	*Type*	*Description*

		projectFile	String	Path to project file
14	**CloseProject** ()			
	Return type	void		
	Description	Close project		
	Parameters	*Parameter*	*Type*	*Description*
		N/A.	N/A.	N/A.
15	**OnQuitCommand** ()			
	Return type	void		
	Description	Display confirmation dialog when user wants to exit application		
	Parameters	*Parameter*	*Type*	*Description*
		obj	object	Contains information about quit event
16	**OpenSelectedItem**()			
	Return type	RelayCommand		
	Description	Open selected file		
	Parameters	*Parameter*	*Type*	*Description*
		SelectedTreeItem	FileTobi	Selected item from project explorer tree
17	**RenameSelectedItem** ()			
	Return type	RelayCommand		
	Description	Rename selected file		
	Parameters	*Parameter*	*Type*	*Description*
		SelectedTreeItem	FileTobi	Selected item from project explorer tree
18	**DeleteSelectedItem** ()			
	Return type	RelayCommand		
	Description	Delete selected file		
	Parameters	*Parameters*	*Type*	*Description*
		SelectedTreeItem	FileTobi	Selected item from project explorer tree
19	**SaveProjectCommand** ()			
	Return type	RelayCommand		
	Description	Save current project		
	Parameters	*Parameters*	*Type*	*Description*
		files	List<FileViewModel>	List of opened files to save
20	**SaveActiveDocument** ()			
	Return type	RelayCommand		
	Description	Save active document		
	Parameters	*Parameters*	*Type*	*Description*
		ActiveDocument	FileViewModel	Selected file from document tabs
21	**CloseActiveDocument** ()			
	Return type	void		
	Description	Close active document		
	Parameters	*Parameters*	*Type*	*Description*

		ActiveDocument	FileViewModel	Selected file from document tabs

Table 77. Workspace

4.5.3 Detailed Designs

4.5.3.1 Project

4.5.3.1.1 Create Project

Screen Design

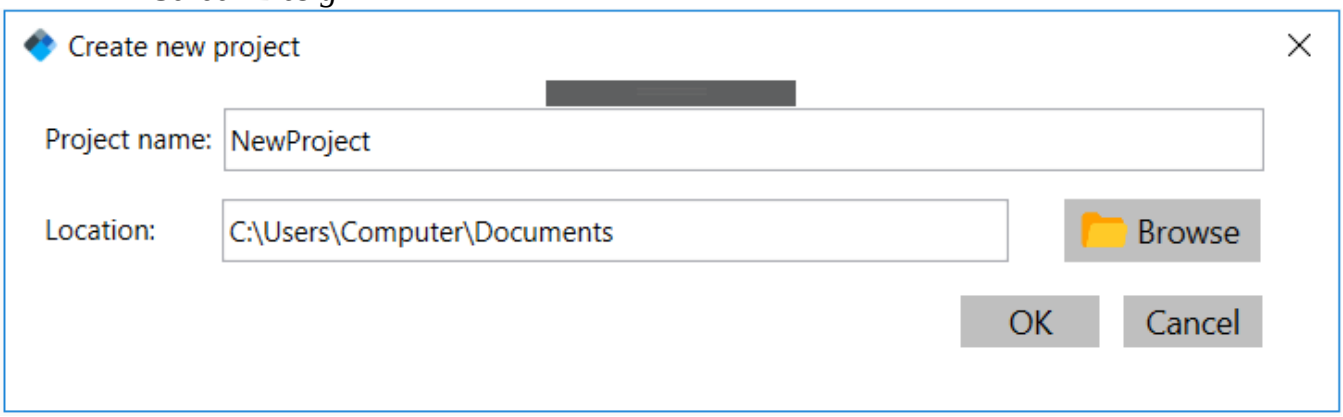


Figure 13. Create Project screen design

Sequence Diagram

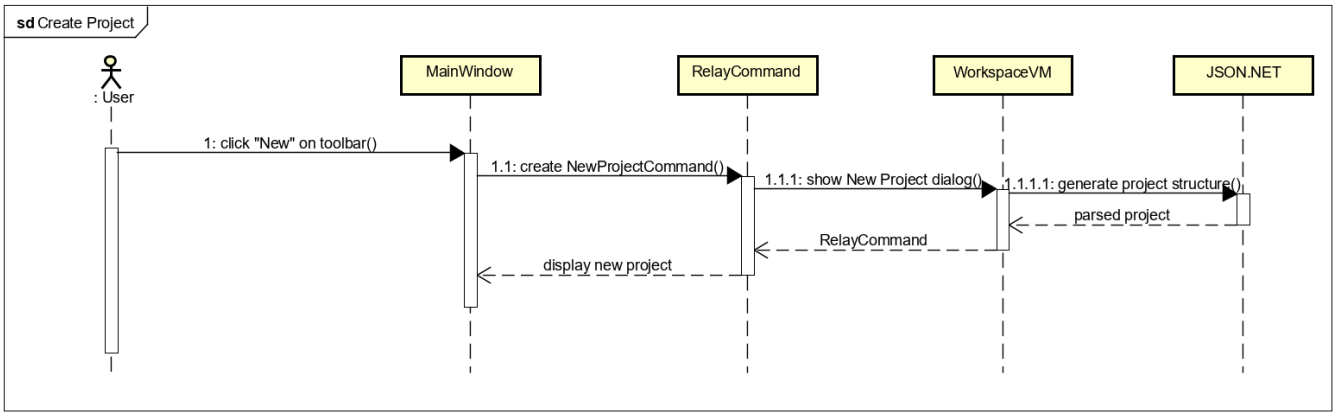


Figure 14. Create Project sequence diagram

4.5.3.1.2 Open Project

Screen Design

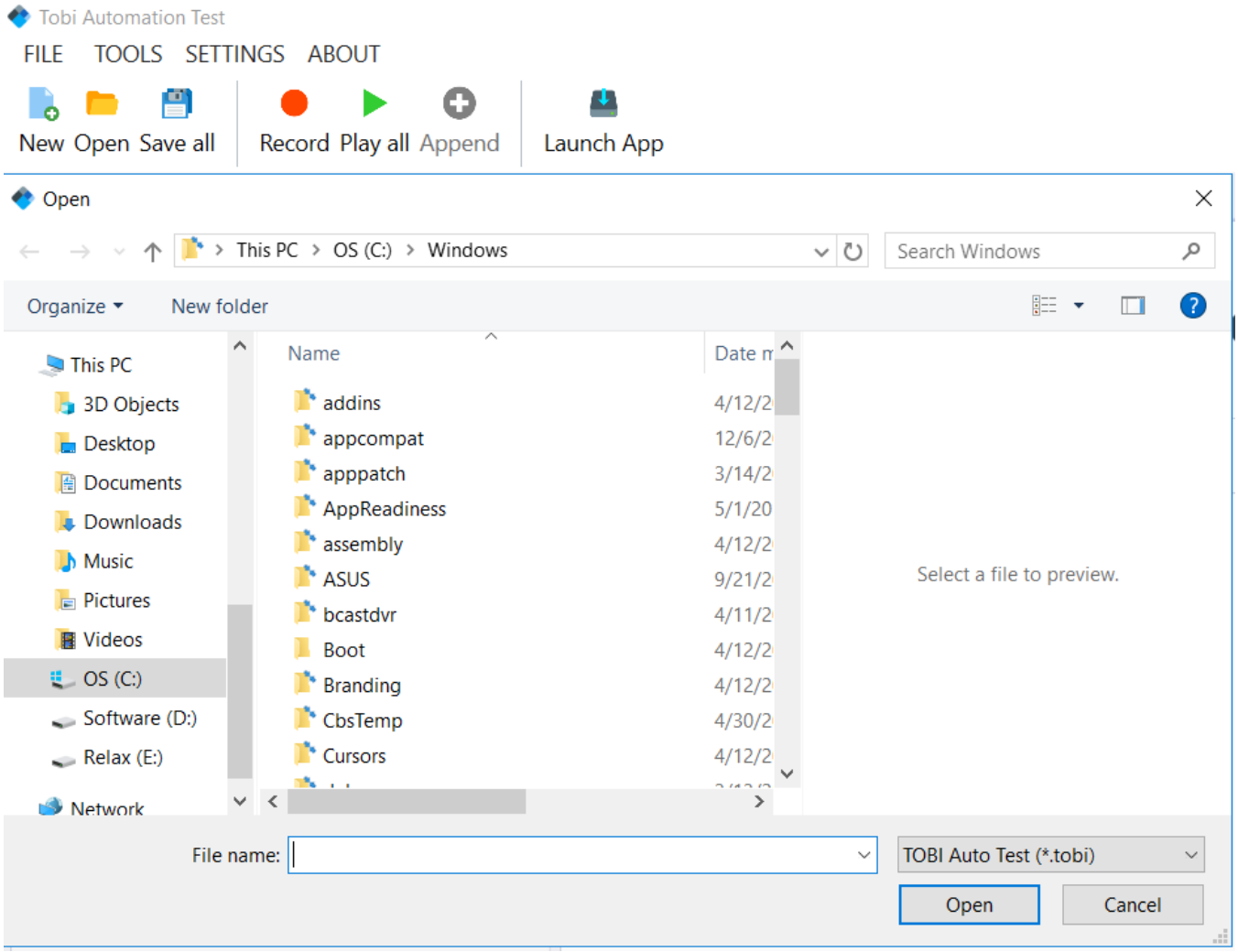


Figure 15. Open Project sequence diagram

Sequence Diagram

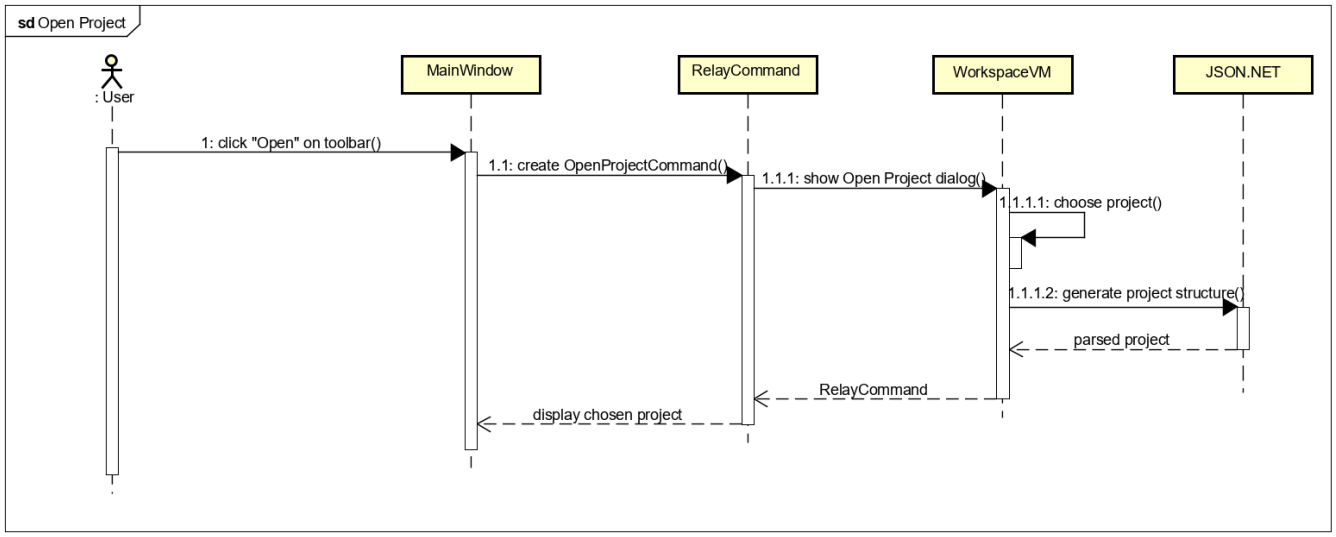


Figure 16. Open Project sequence diagram

4.5.3.1.3 Close Project

Screen Design

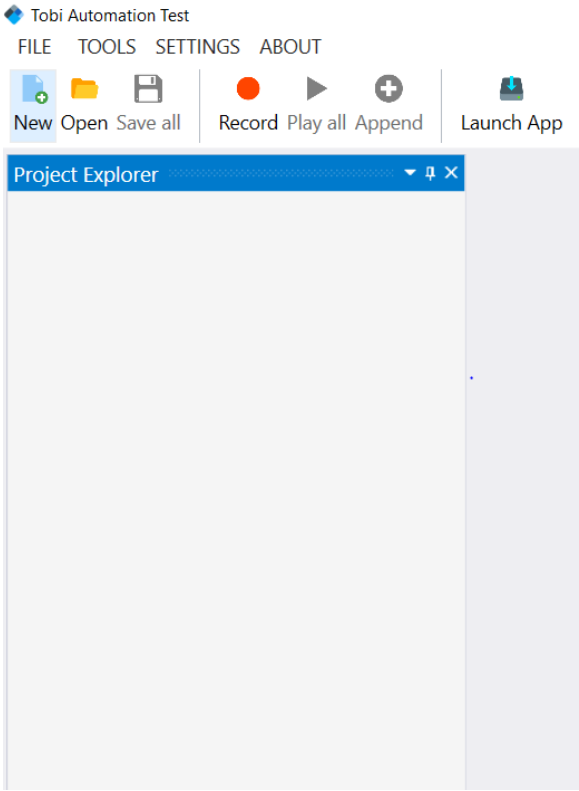


Figure 17.Close Project sequence diagram

Sequence Diagram

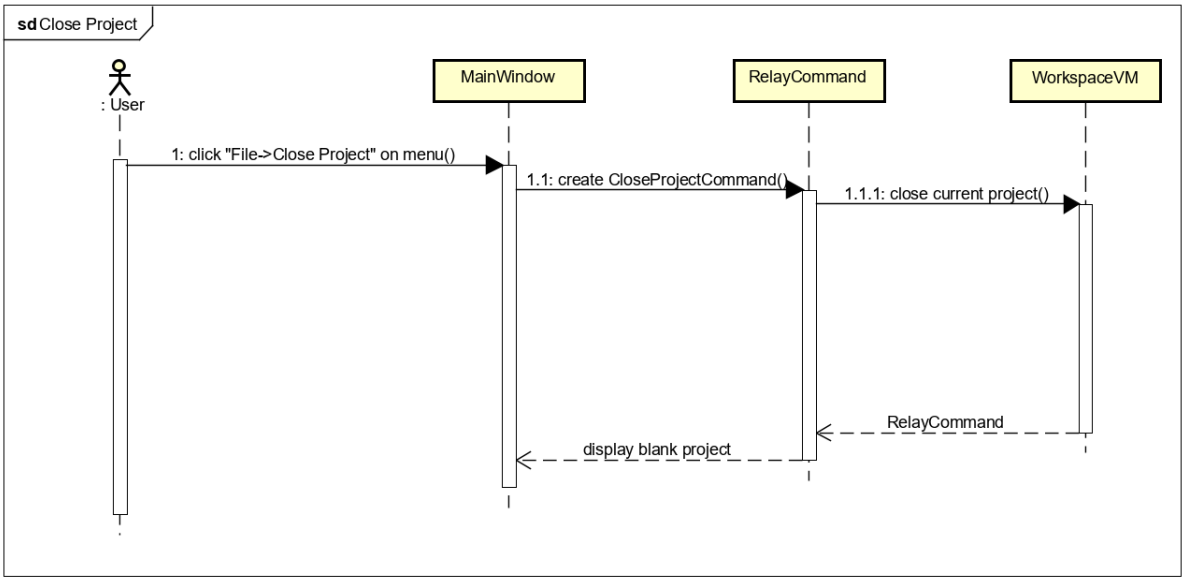


Figure 18. Close Project sequence diagram

4.5.3.1.4 Save File

Screen Design

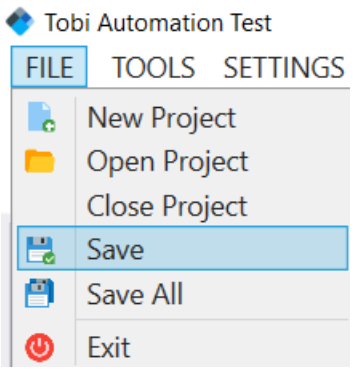


Figure 19. Save File screen design

Sequence Diagram

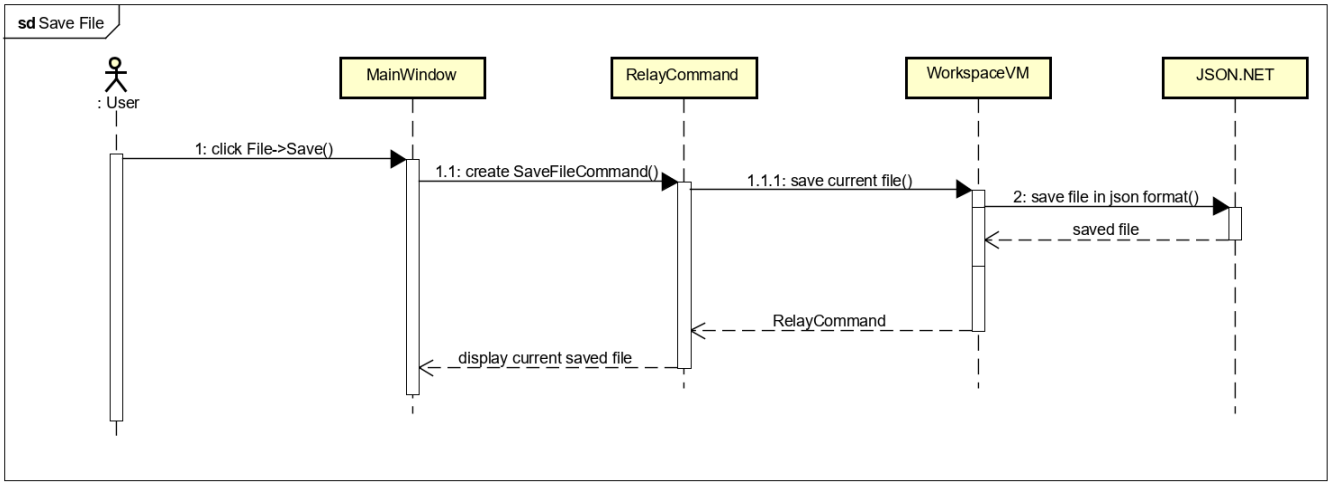


Figure 20. Save File sequence diagram

4.5.3.1.5 Save Current Project

Screen Design

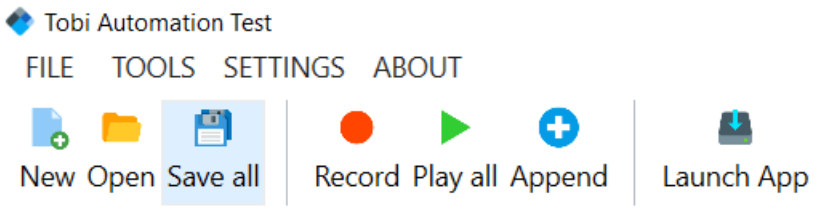


Figure 21. Save Project screen design

Sequence Diagram

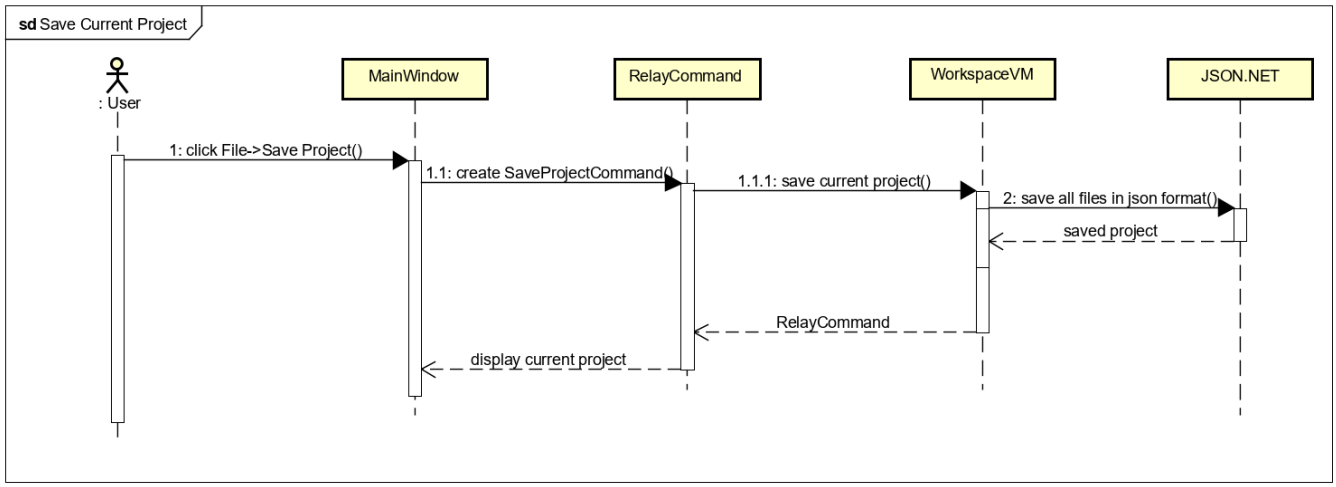


Figure 22. Save File sequence diagram

4.5.3.1.6 Exit Application

Screen Design

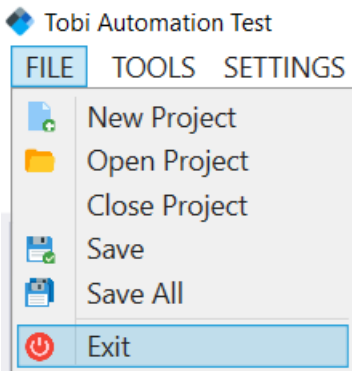


Figure 23. Exit Application screen design

Sequence Diagram

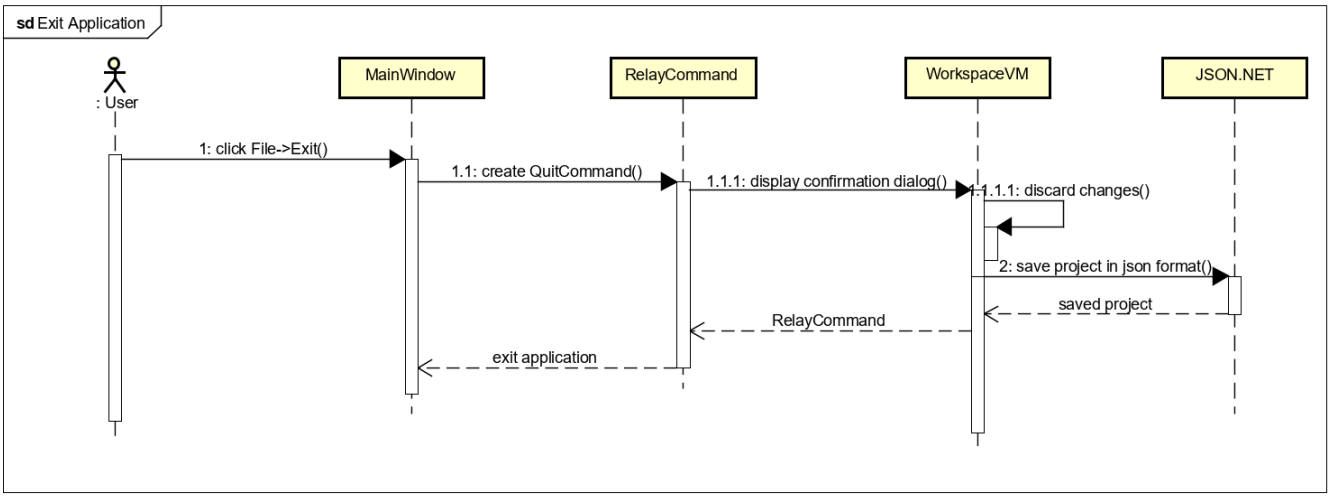


Figure 24. Exit Application sequence diagram

4.5.3.1.7 Open File

Screen Design

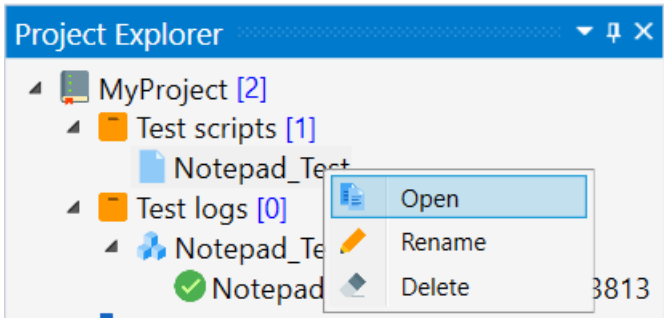


Figure 25. Open File screen design

Sequence Diagram

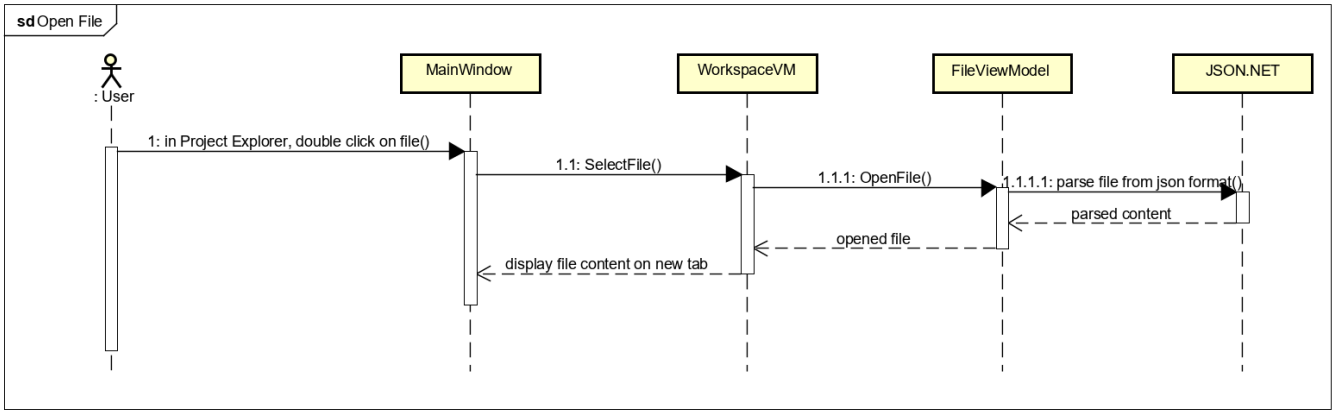


Figure 26. Open File sequence diagram

4.5.3.1.8 Rename File

Screen Design

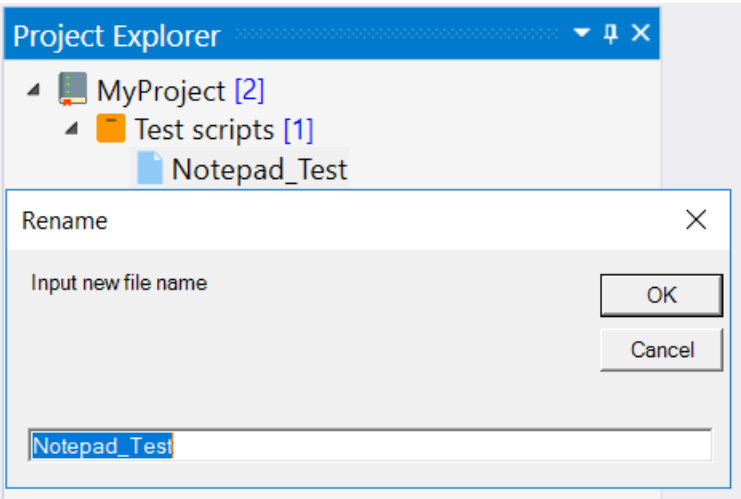


Figure 27. Rename File screen design

Sequence Diagram

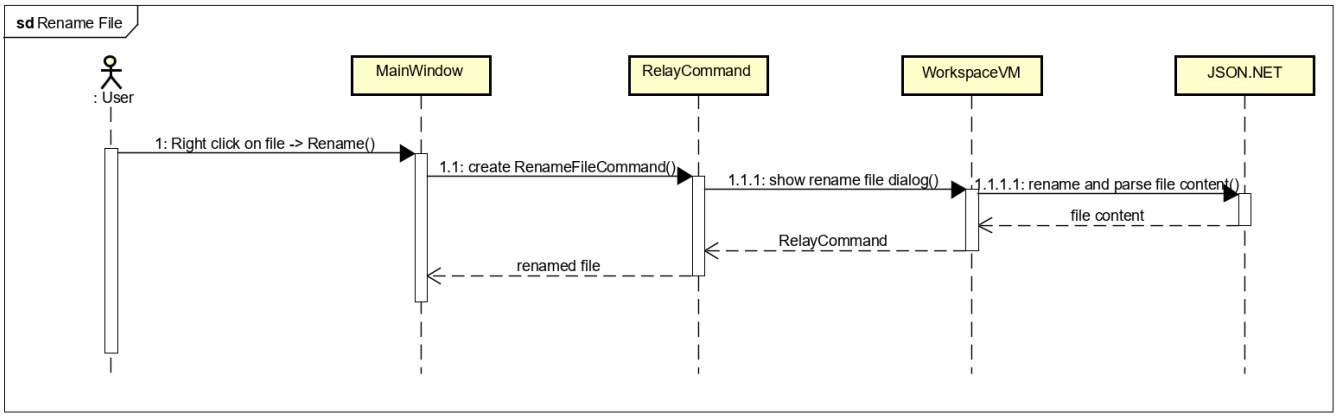


Figure 28. Rename File sequence diagram

4.5.3.1.9 Delete File

Screen Design

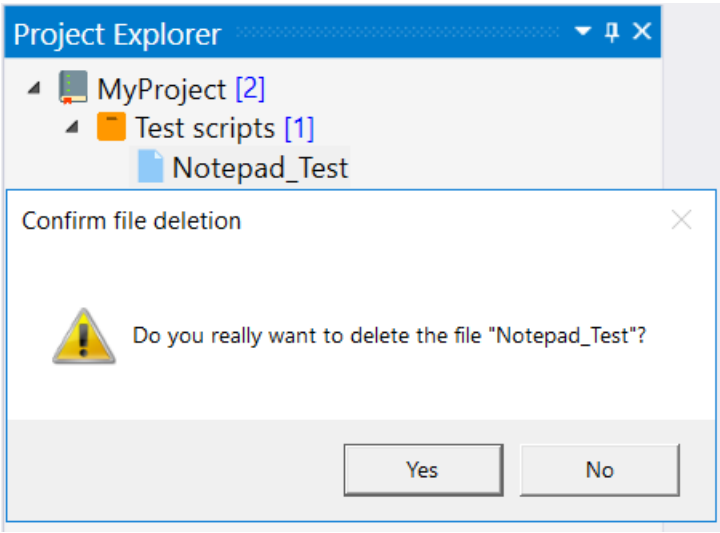


Figure 29. Delete File screen design

Sequence Diagram

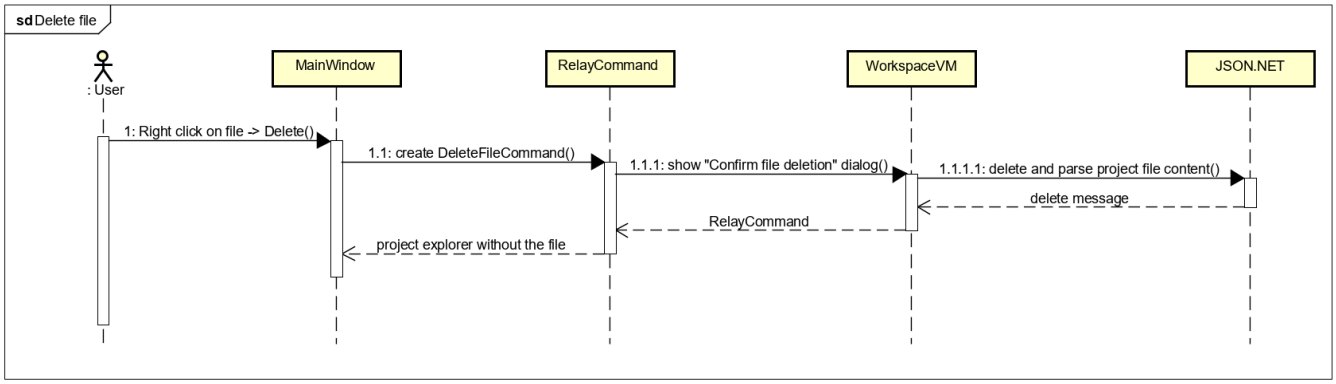


Figure 30. Delete File sequence diagram

4.5.3.2 Record

4.5.3.2.1 Record New Test Case

Screen Design

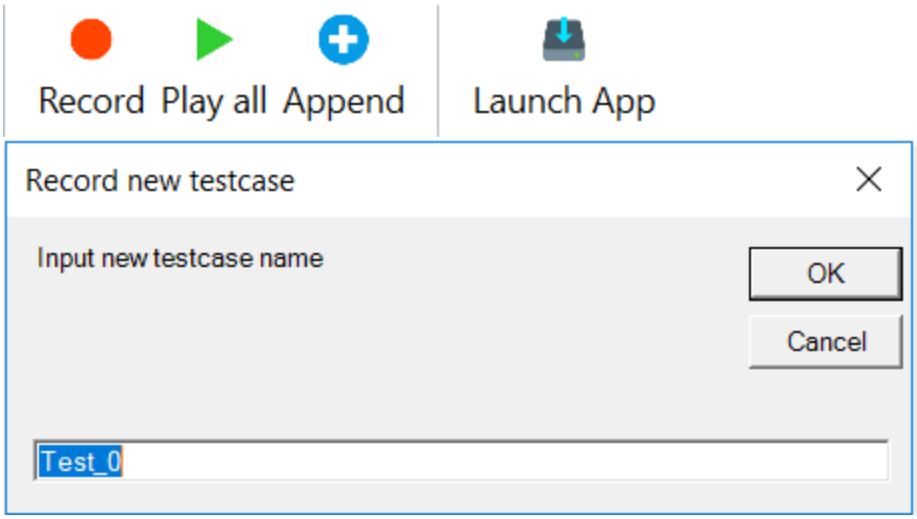


Figure 31. Record New Test Case screen design

Sequence Diagram

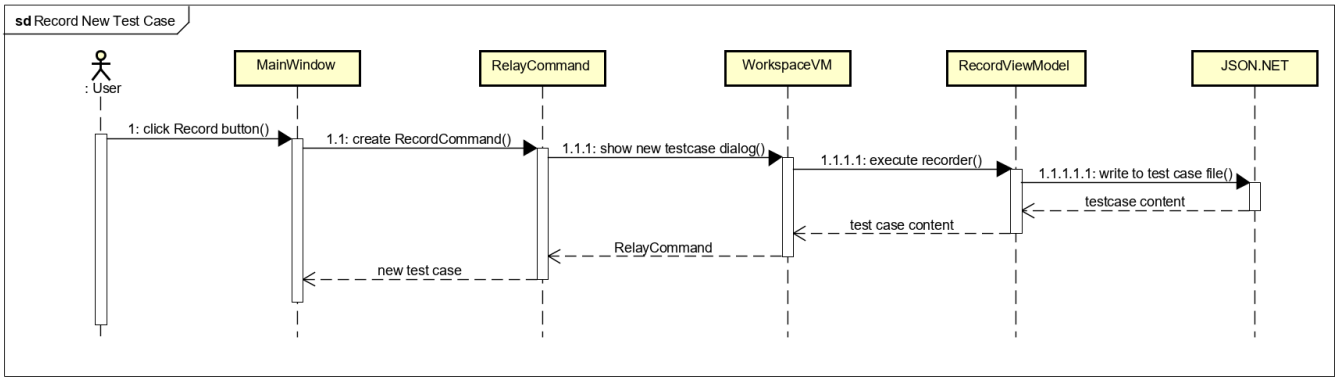


Figure 32. Record New Test Case sequence diagram

4.5.3.2.2 Record Current Test Case

Screen Design

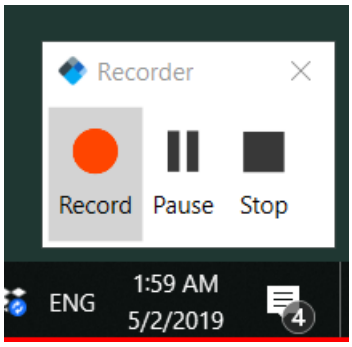


Figure 33. Record Current Test Case screen design

Sequence Diagram

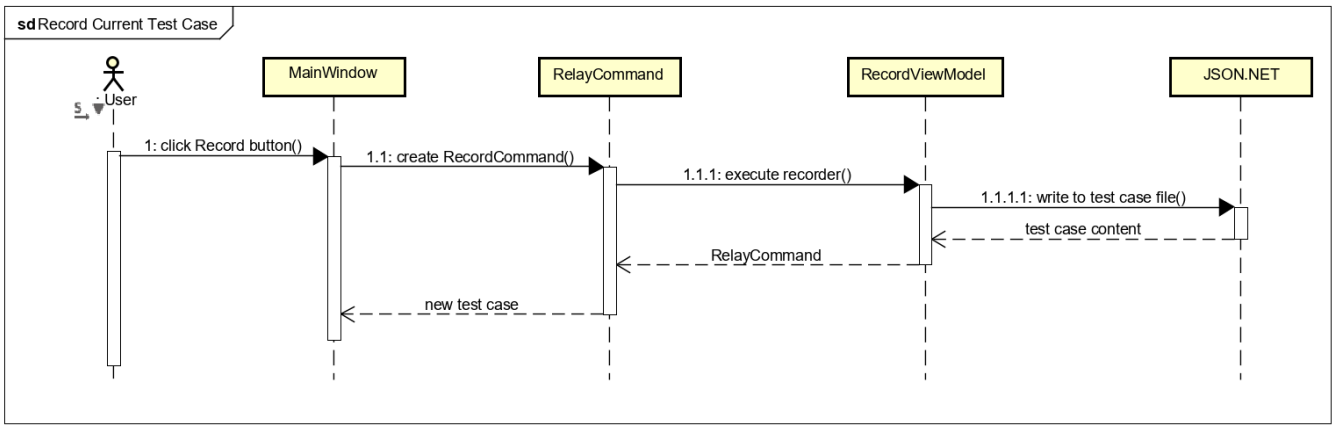


Figure 34. Record Current Test Case sequence diagram

4.5.3.2.3 Append Test Case

Screen Design

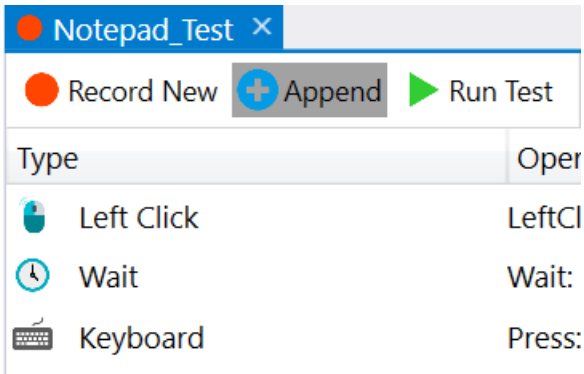


Figure 35. Append Test Case screen design

Sequence Diagram

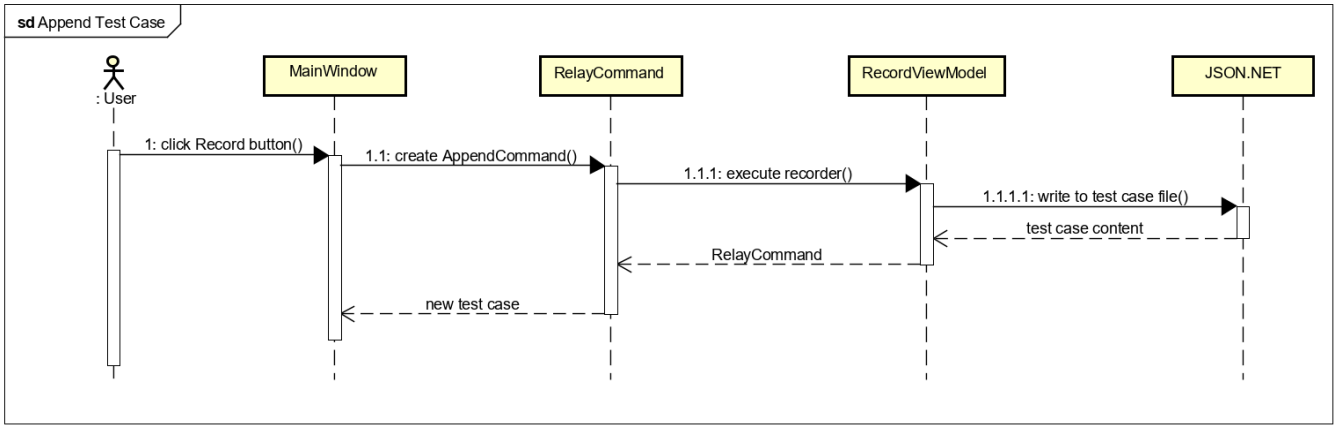


Figure 36. Append Test Case sequence diagram

4.5.3.2.4 Run Test Case

Screen Design

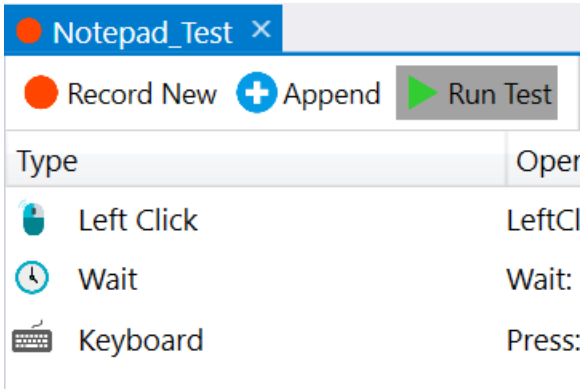


Figure 37. Run Test Case screen design

Sequence Diagram

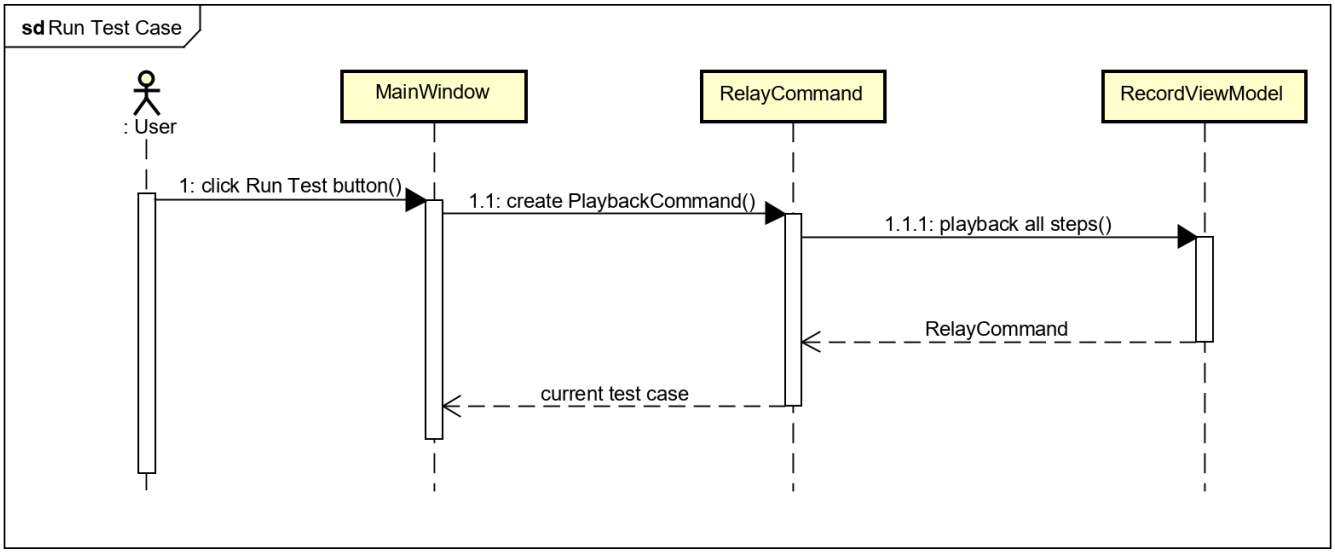


Figure 38. Run Test Case sequence diagram

4.5.3.2.5 Play All Test Cases

Screen Design

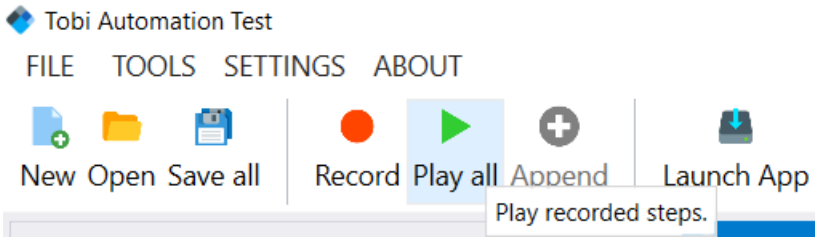


Figure 37. Play All Test Cases screen design

Sequence Diagram

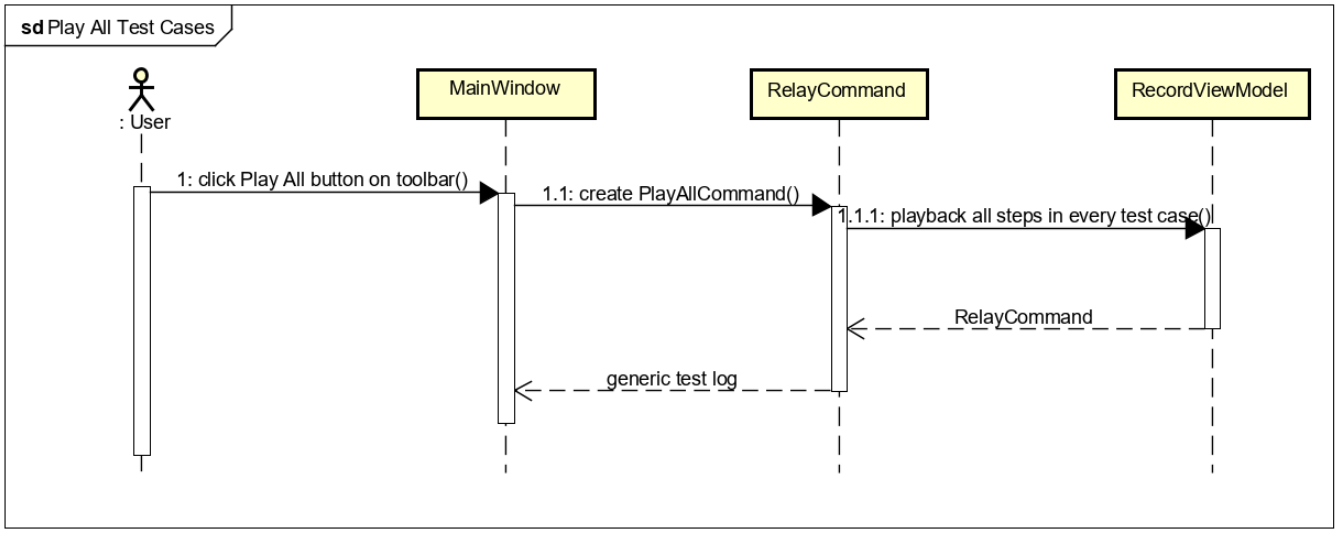


Figure 38. Play All Test Cases sequence diagram

4.5.3.2.6 Add Manual Mouse Action

Screen Design

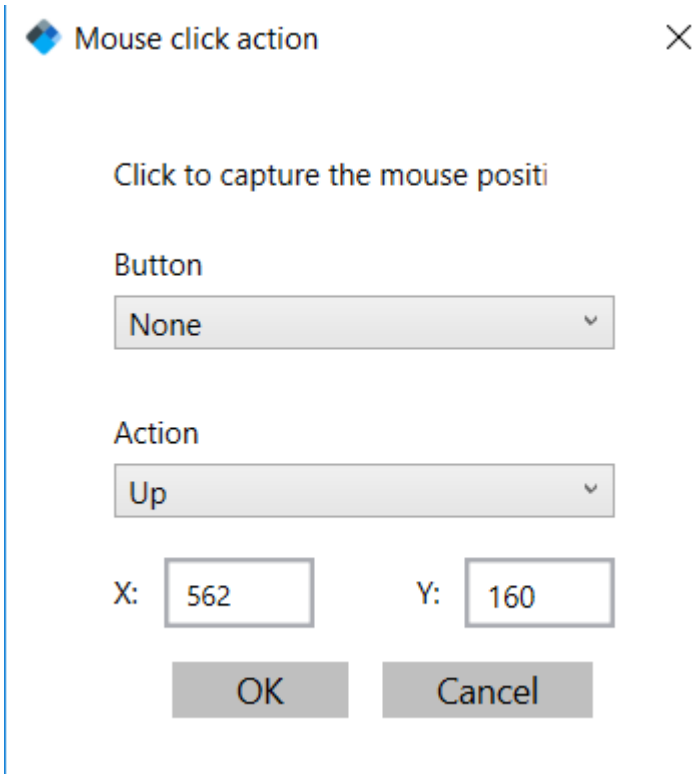


Figure 39.Add Manual Mouse Action screen design

Sequence Diagram

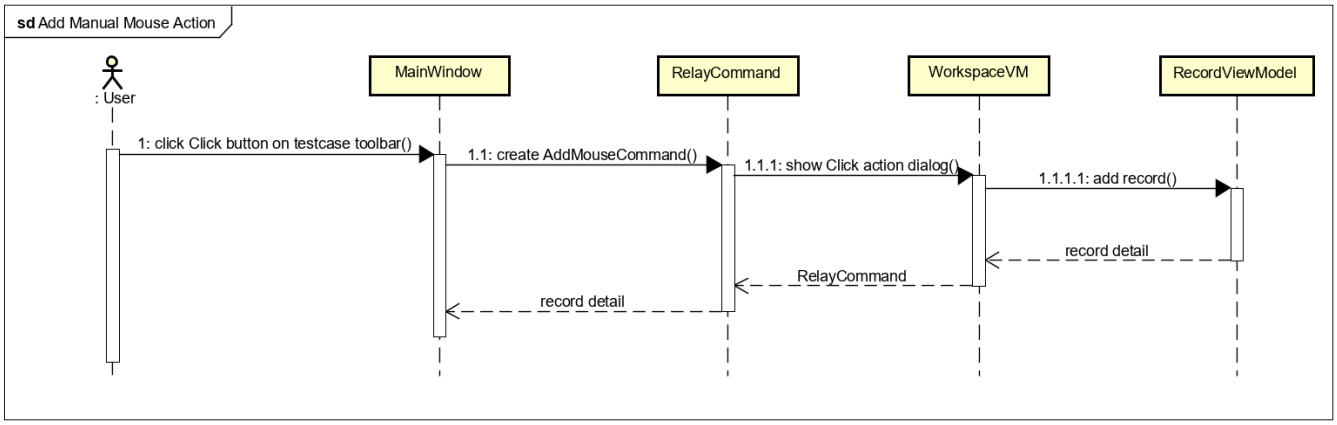


Figure 40.Add Manual Mouse Action sequence diagram

4.5.3.2.7 Add Manual Keyboard Action

Screen Design

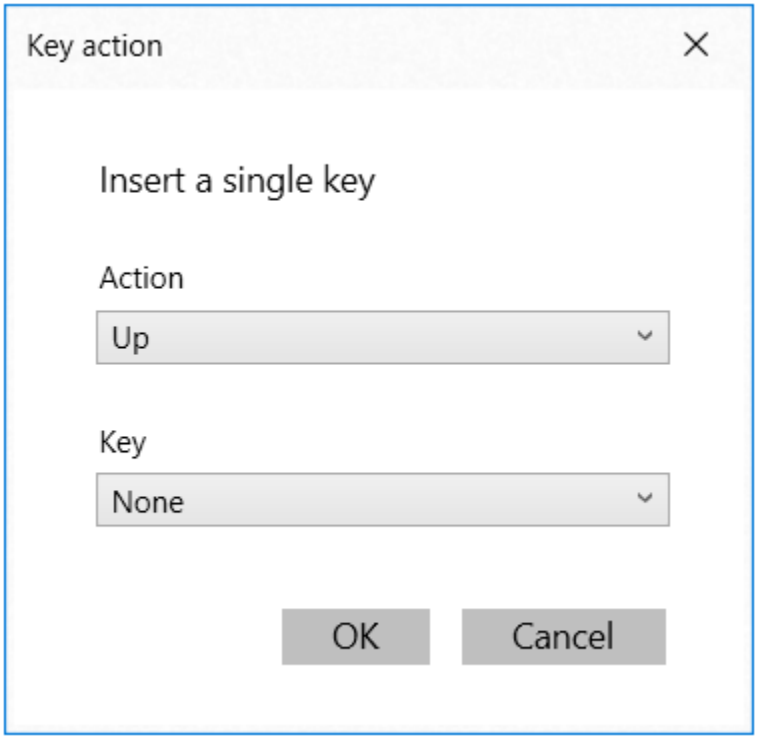


Figure 41.Add Manual Keyboard Action screen design

Sequence Diagram

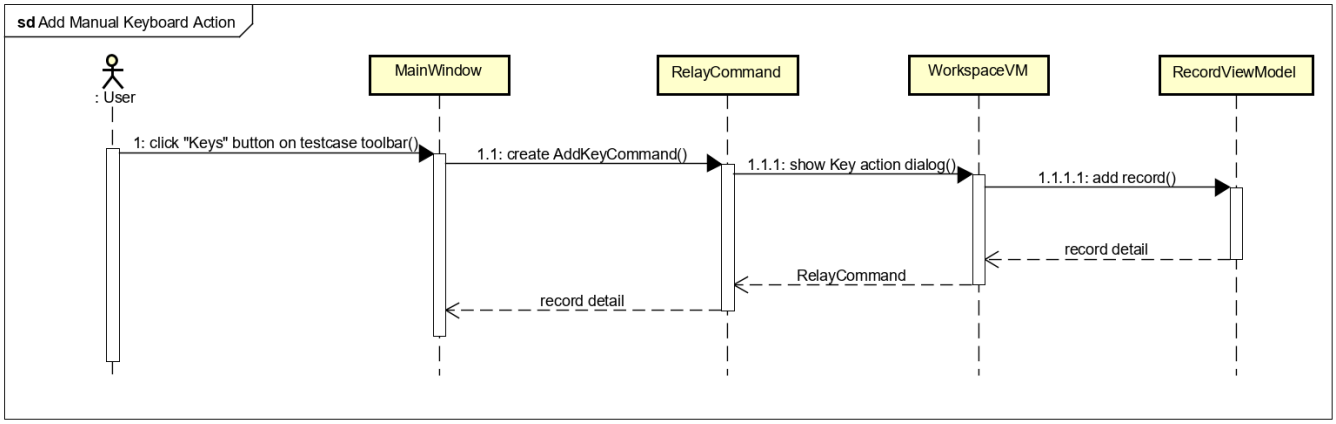


Figure 42.Add Manual Keyboard Action sequence diagram

4.5.3.2.8 Add Text Action

Screen Design

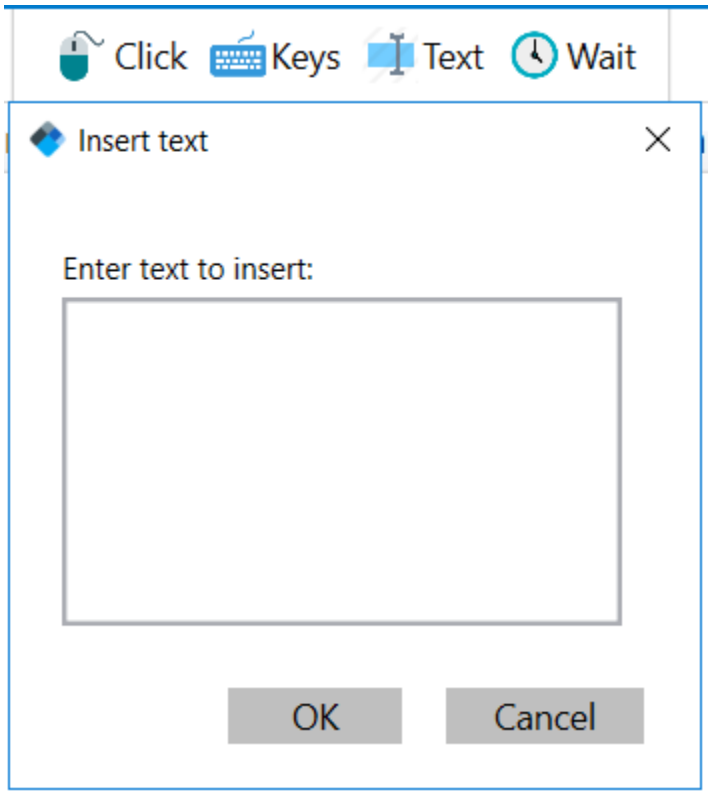


Figure 43.Add Text Action screen design

Sequence Diagram

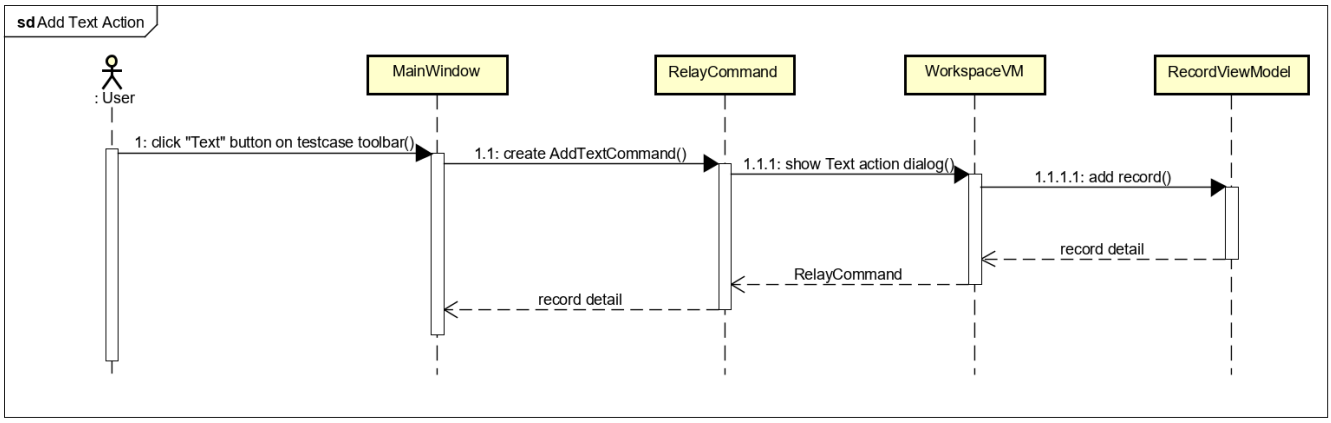


Figure 44.Add Text Action sequence diagram

4.5.3.2.9 Add Manual Wait Time

Screen Design

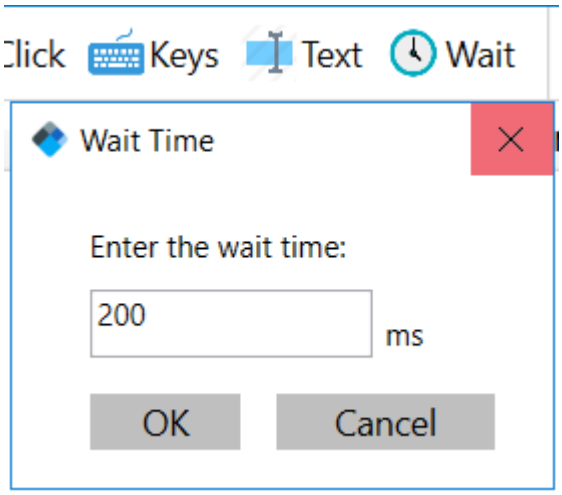


Figure 45.Add Manual Wait Action screen design

Sequence Diagram

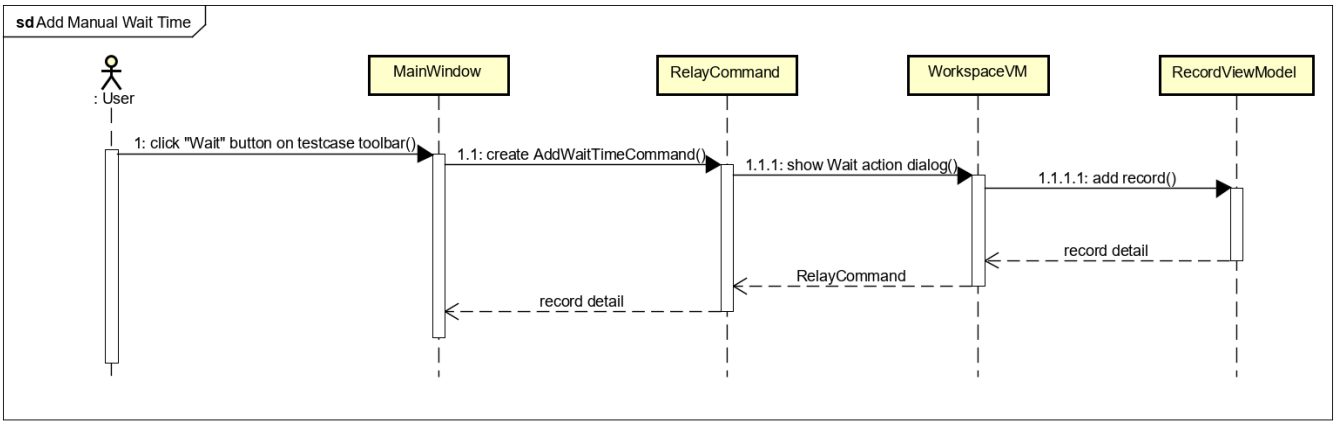


Figure 46.Add Manual Wait Action sequence diagram

4.5.3.2.10 Edit Step

Screen Design

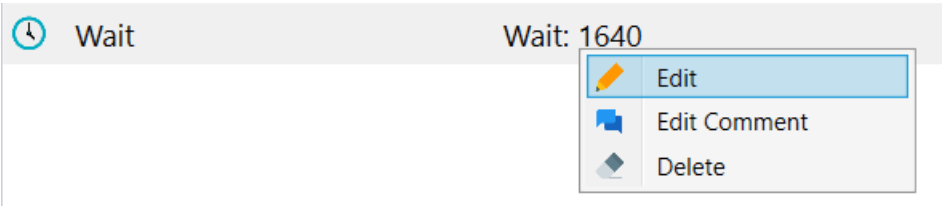


Figure 47. Edit Step screen design

Sequence Diagram

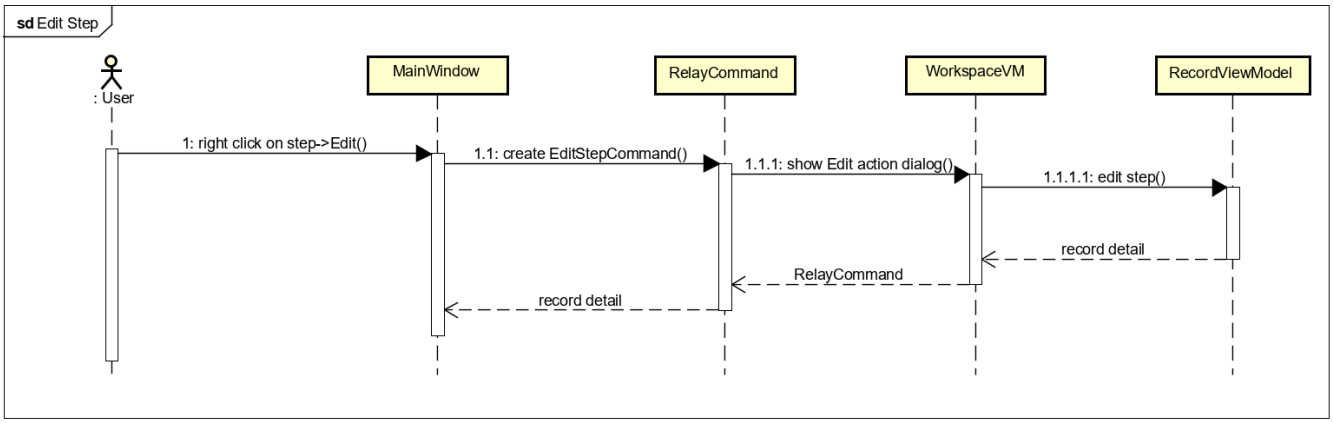


Figure 48. Edit Step sequence diagram

4.5.3.2.11 Delete Step

Screen Design

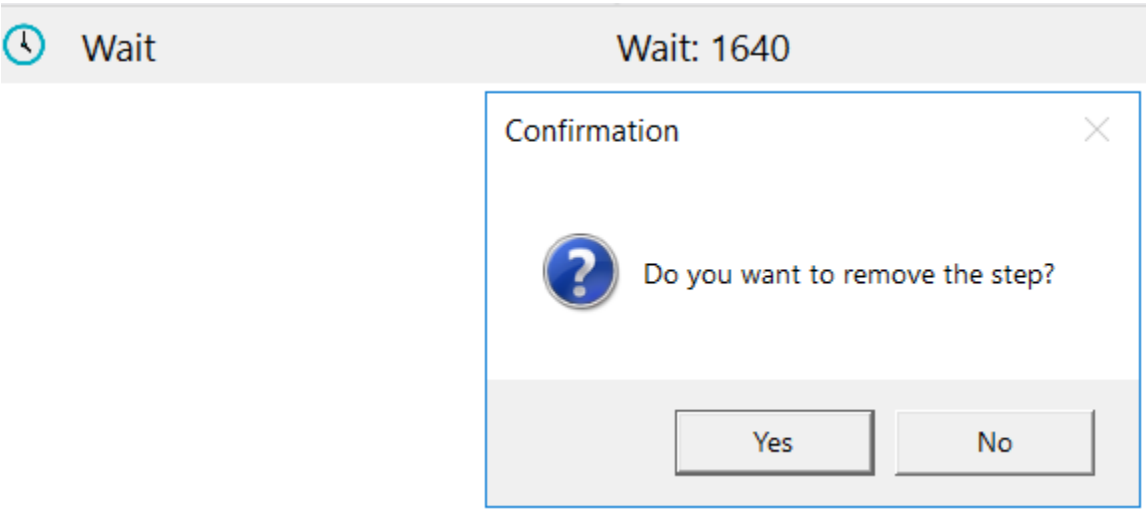


Figure 49. Delete Step screen design

Sequence Diagram

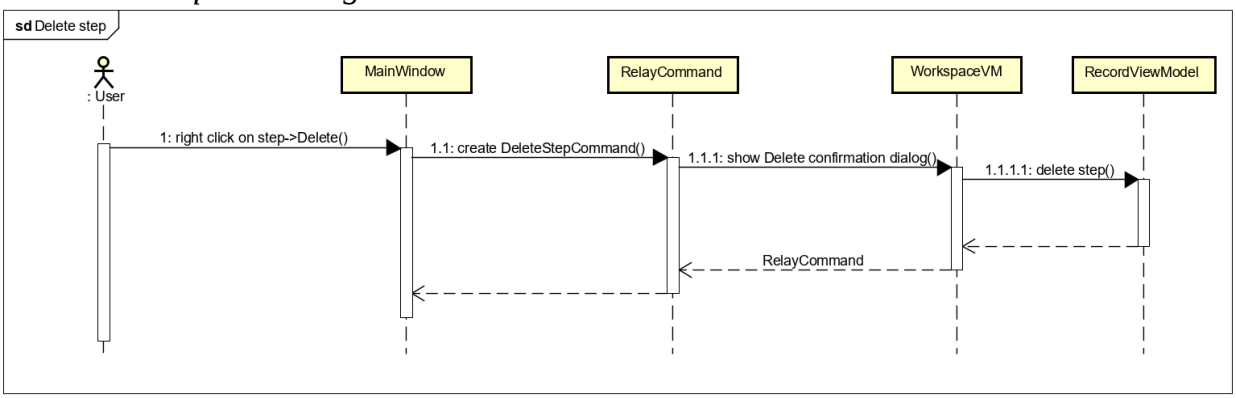


Figure 50. Delete Step sequence diagram

4.5.3.2.12 Open Test Case

Screen Design

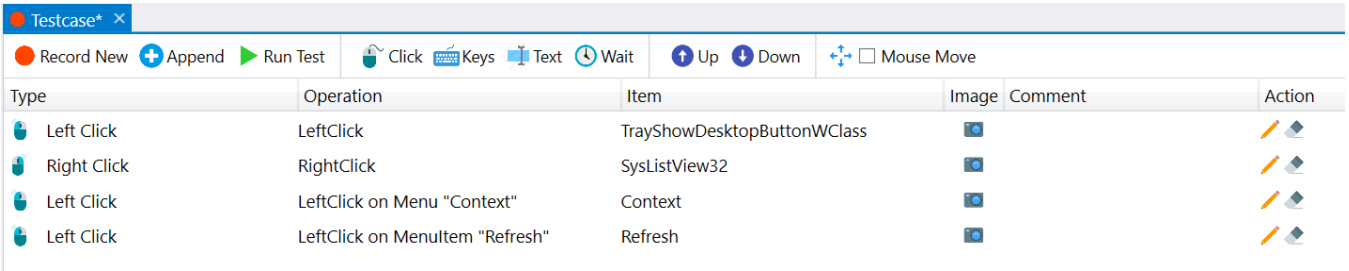


Figure 51. Open Test Case screen design

Sequence Diagram

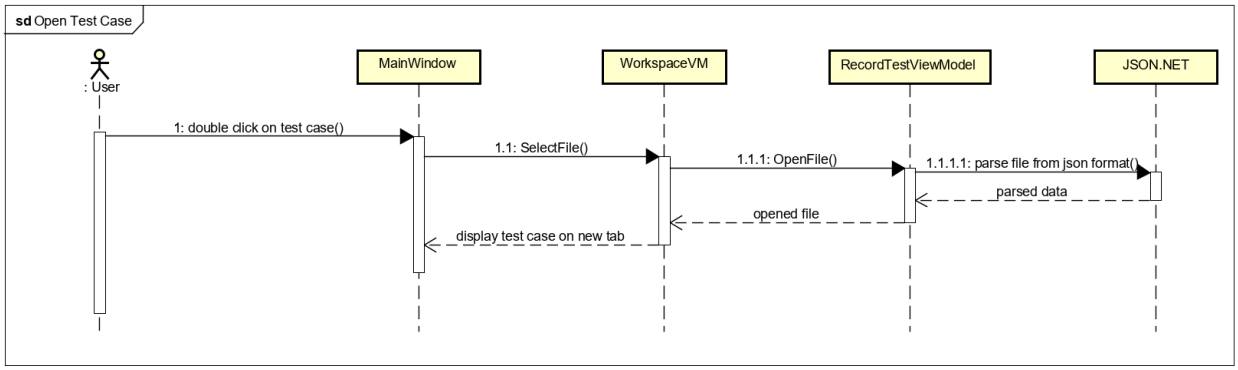


Figure 52.Open Test Case sequence diagram

4.5.3.2.13 Filter Step By Mouse Move

Screen Design

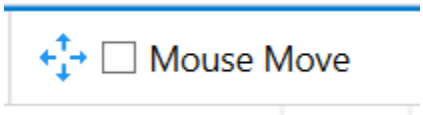


Figure 53.Filter Step By Mouse Move screen design

Sequence Diagram

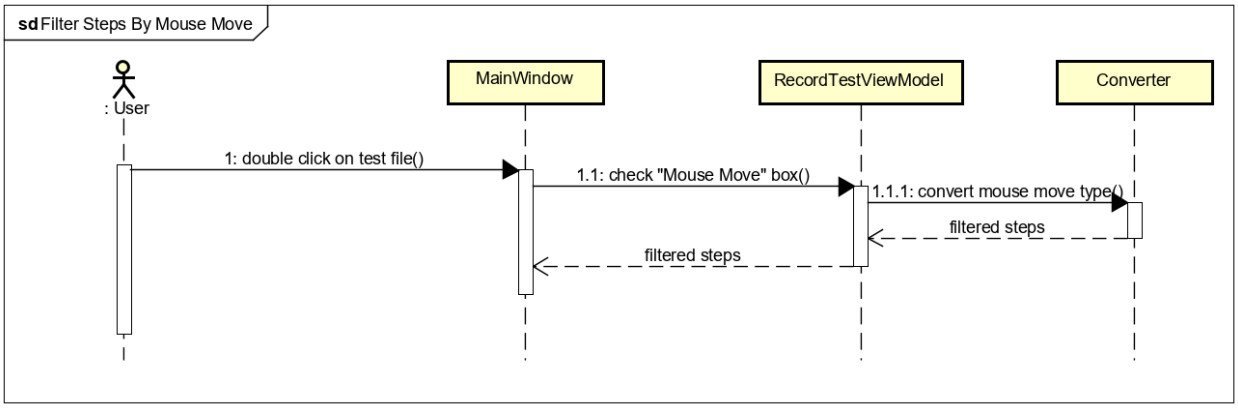


Figure 54.Filter Step By Mouse Move sequence diagram

4.5.3.3 Log

4.5.3.3.1 Open Log File

Screen Design

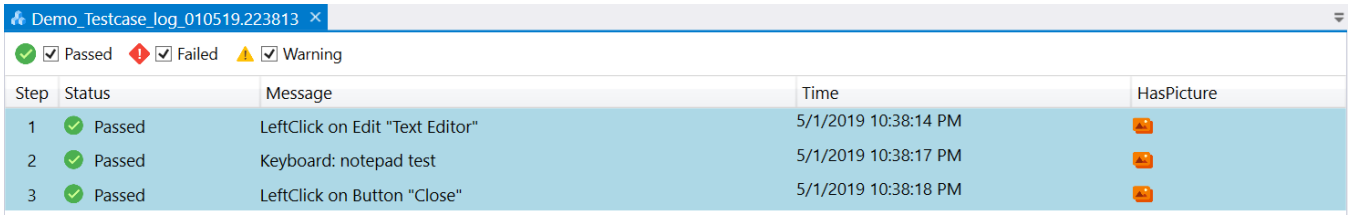


Figure 55.Open Log File screen design

Sequence Diagram

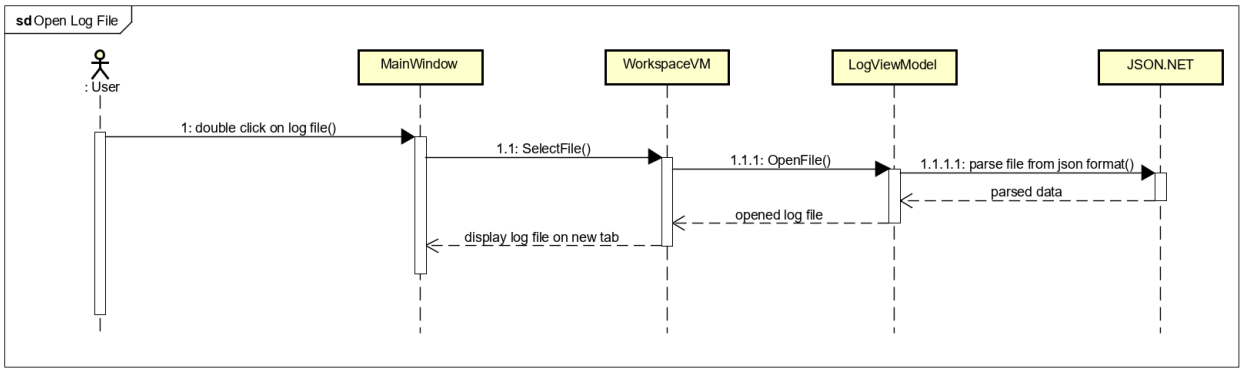


Figure 56.Open Log File sequence diagram

4.5.3.3.2 Count Displayed Steps

Screen Design

Step	Status	Message	Time	HasPicture
1	Passed	LeftClick on Edit "Text Editor"	5/1/2019 10:38:14 PM	
2	Passed	Keyboard: notepad test	5/1/2019 10:38:17 PM	
3	Passed	LeftClick on Button "Close"	5/1/2019 10:38:18 PM	

Logs: 3

Figure 57.Count Displayed Steps screen design

Sequence Diagram

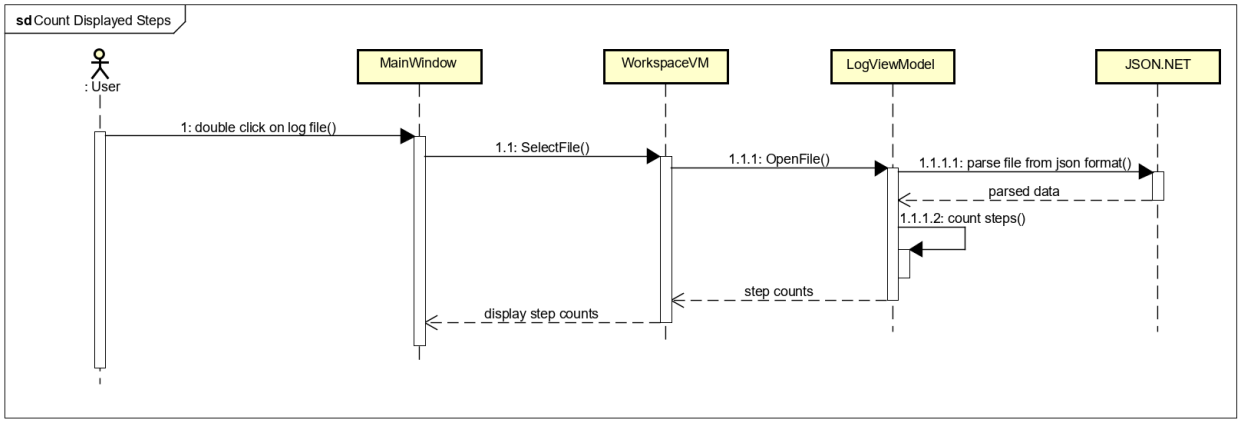


Figure 58.Count Displayed Steps sequence diagram

4.5.3.3.3 Filter Steps By Status

Screen Design

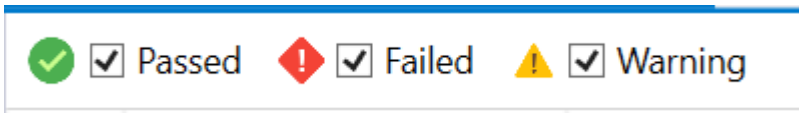


Figure 59.Filter Steps By Status screen design

Sequence Diagram

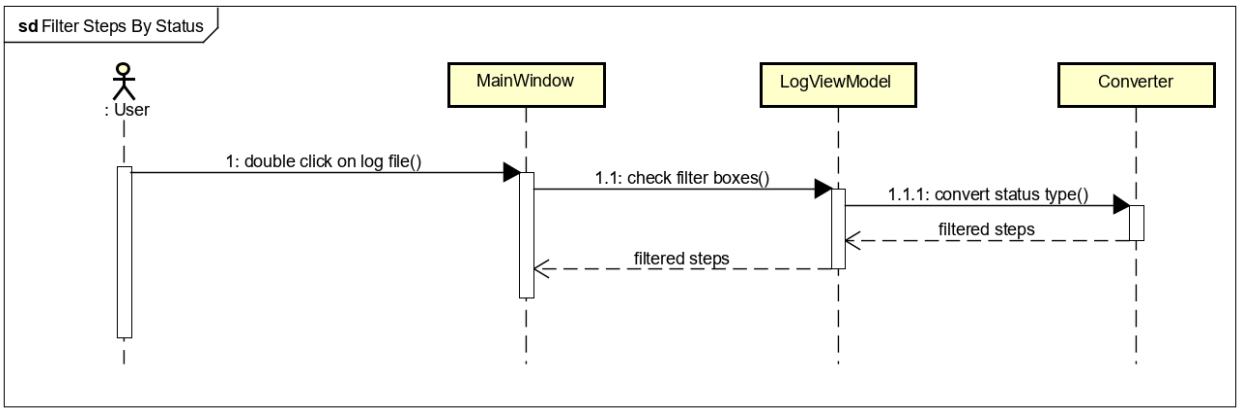


Figure 60.Filter Steps By Status sequence diagram

4.5.3.4 Test Visualizer

4.5.3.4.1 Open Test Visualizer

Screen Design

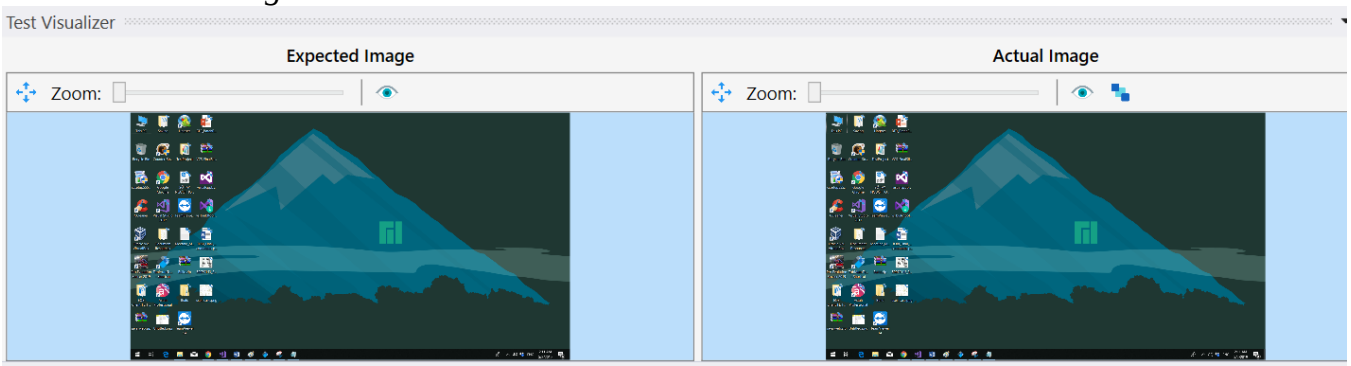


Figure 65. Open Test Visualizer screen design

Sequence Diagram

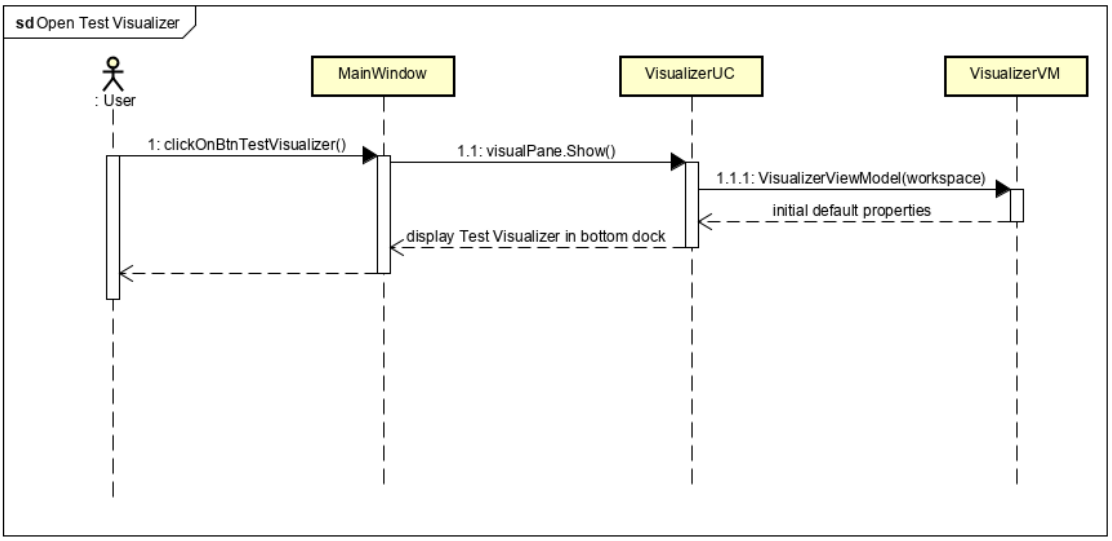


Figure 66. Open Test Visualizer sequence diagram

4.5.3.4.2 Display Image List (Record)

Screen Design

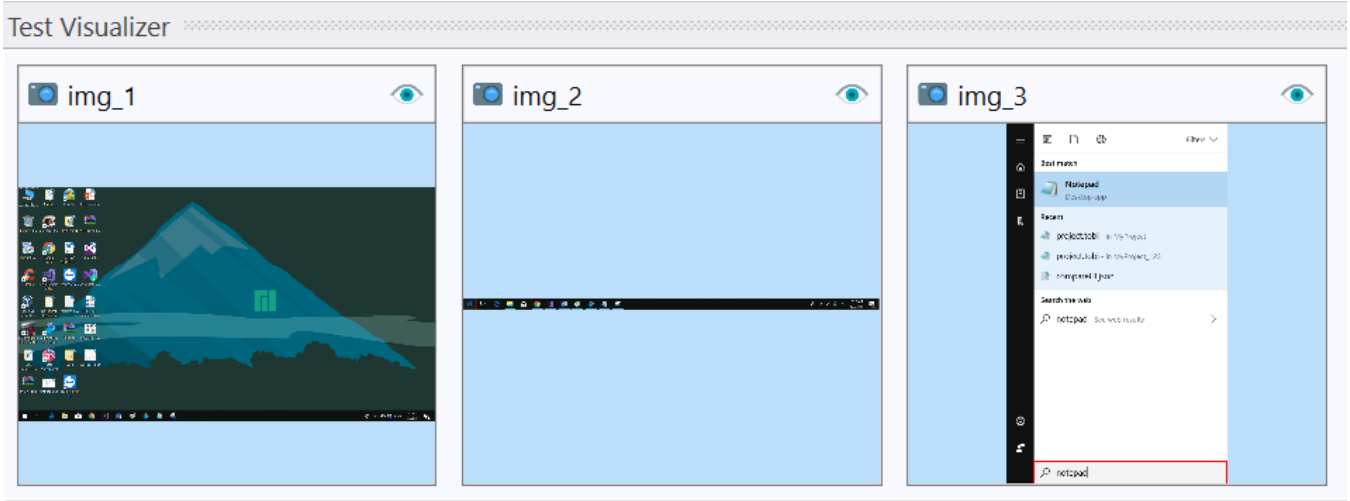


Figure 67. Display Image List (Record) screen design

Sequence diagram

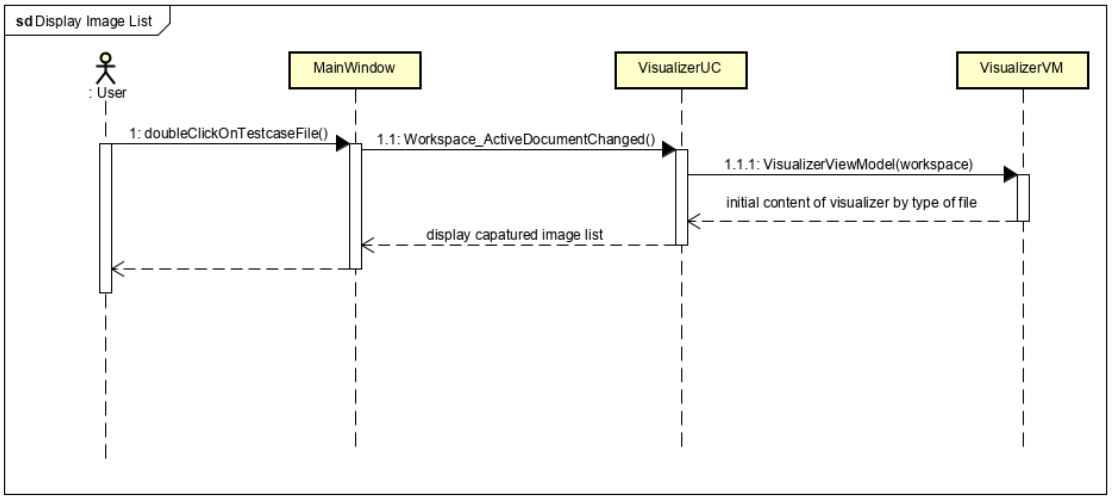


Figure 68. Display Image List (Record) sequence diagram

4.5.3.4.3 Display Image Pair

Screen design

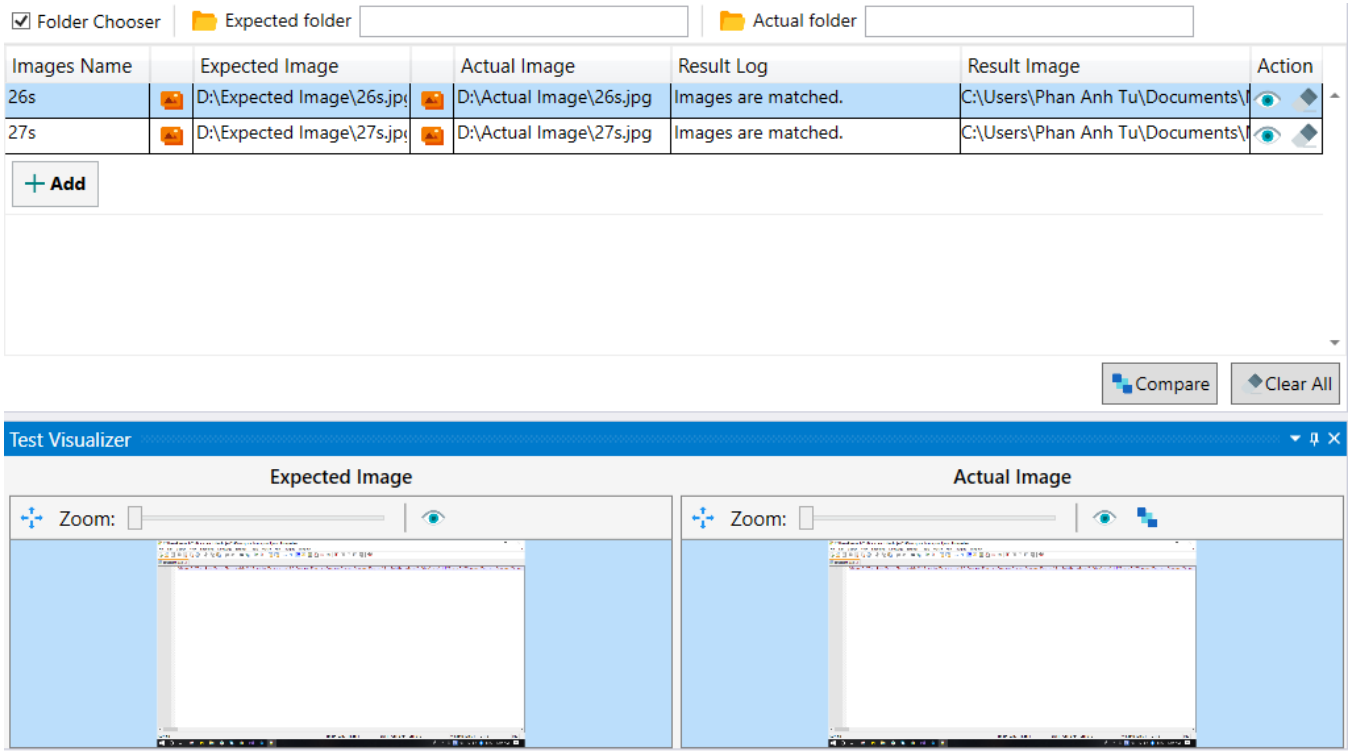


Figure 69. Display Image Pair (Log, UI Comparison) screen design

Sequence diagram

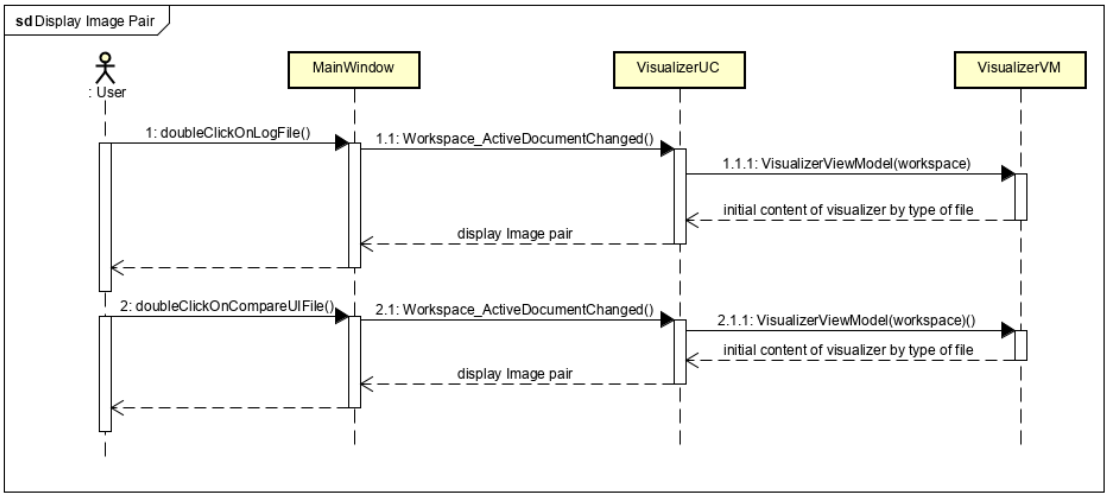


Figure 70. Display Image Pair (Log, UI Comparison) sequence diagram

4.5.3.4.4 Fit Image With Panel

Screen design

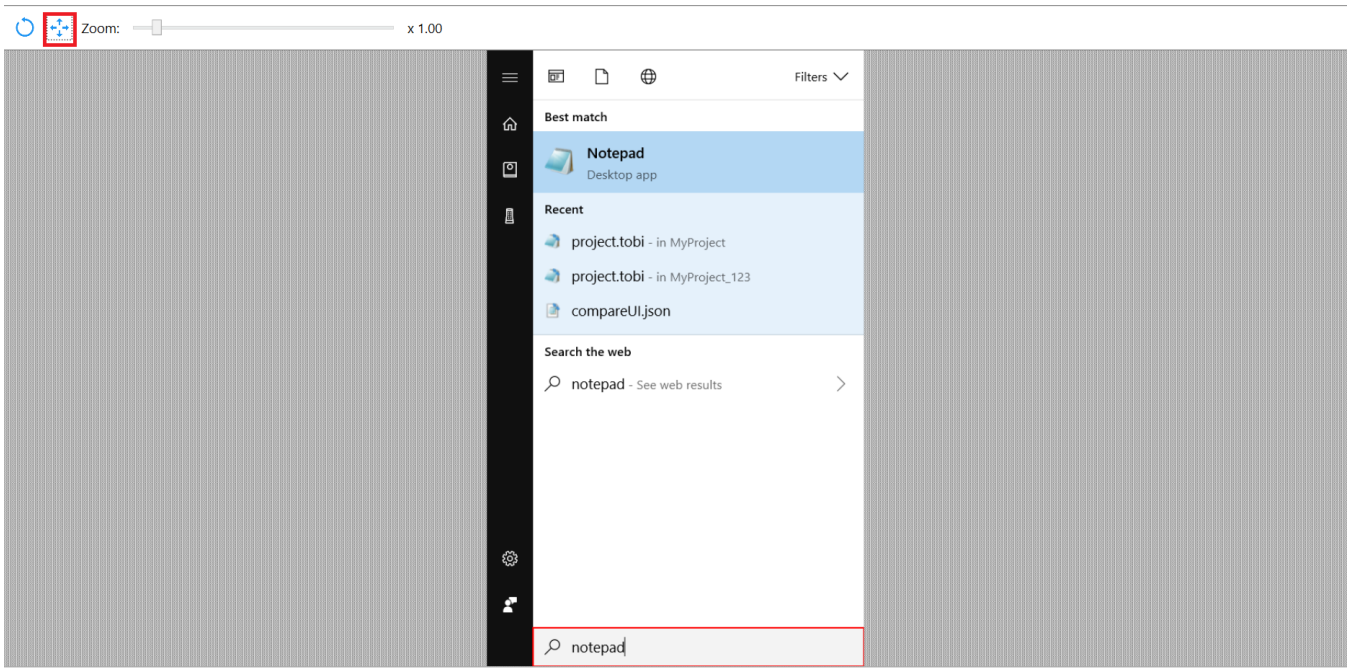


Figure 71. Fit Image With Panel screen design

Sequence diagram

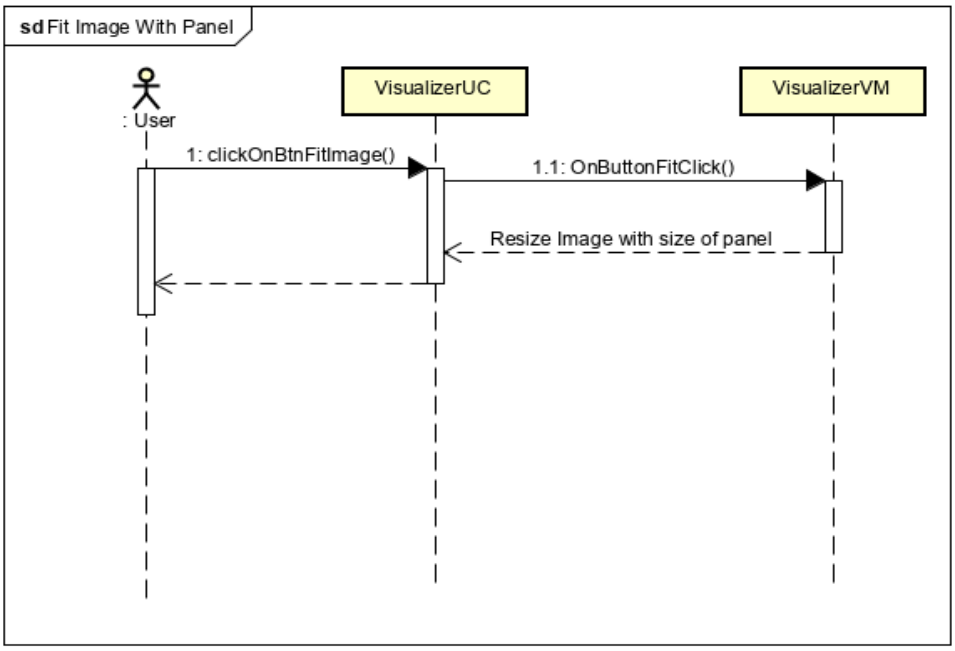


Figure 72. Fit Image With Panel sequence diagram

4.5.3.4.5 Zoom Image

Screen design

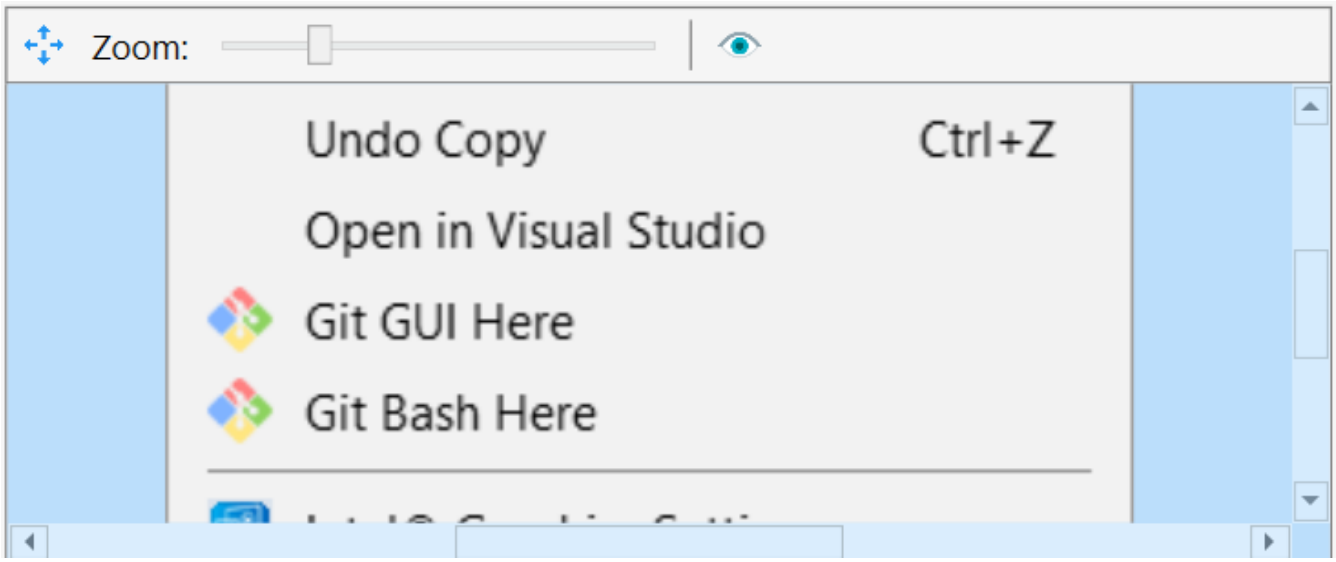


Figure 73. Zoom Image screen design

Sequence diagram

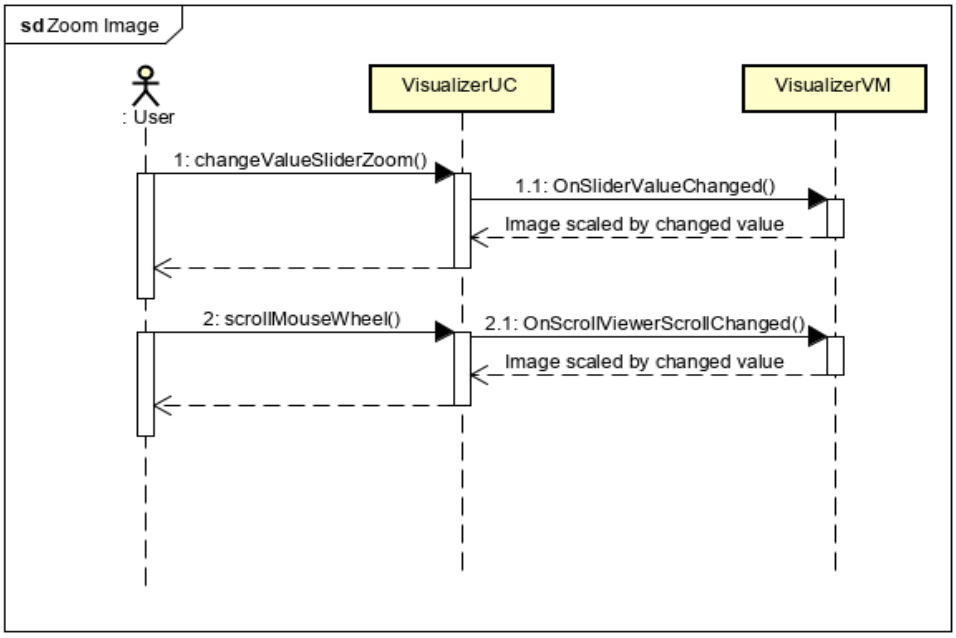


Figure 74. Zoom Image sequence diagram

4.5.3.4.6 Highlight Interacted UI Element

Screen design

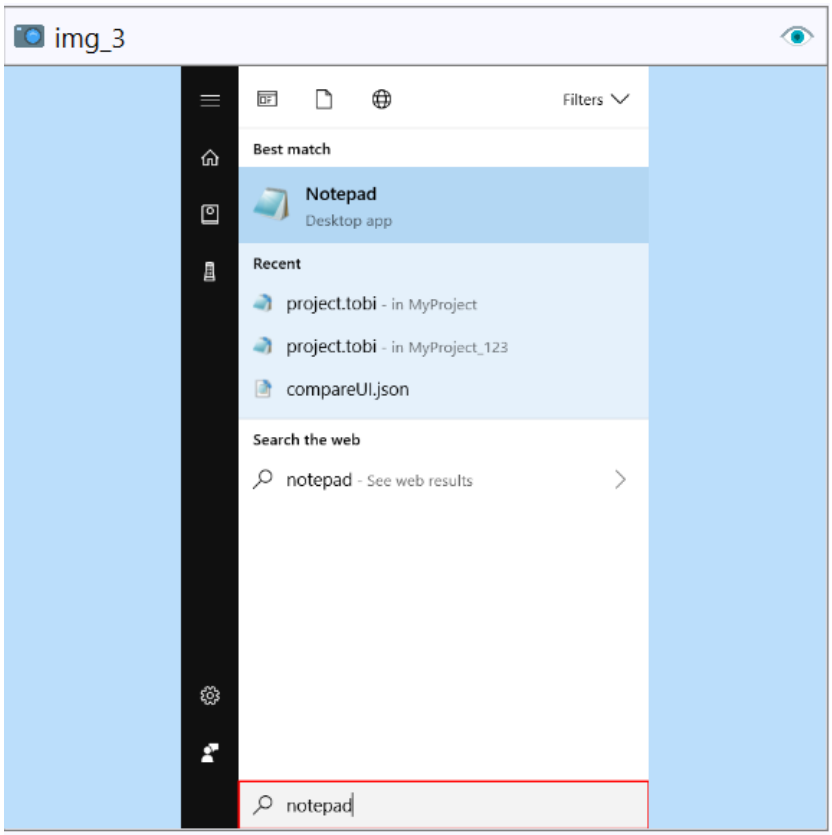


Figure 75. Highlight Interacted UI Element screen design

Sequence diagram

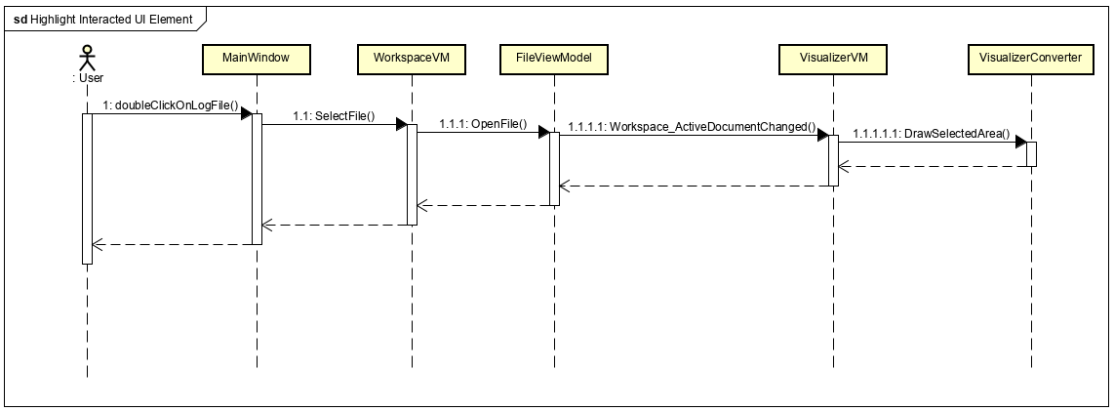


Figure 76. Highlight Interacted UI Element sequence diagram

4.5.3.4.7 Show Result Image

Screen design

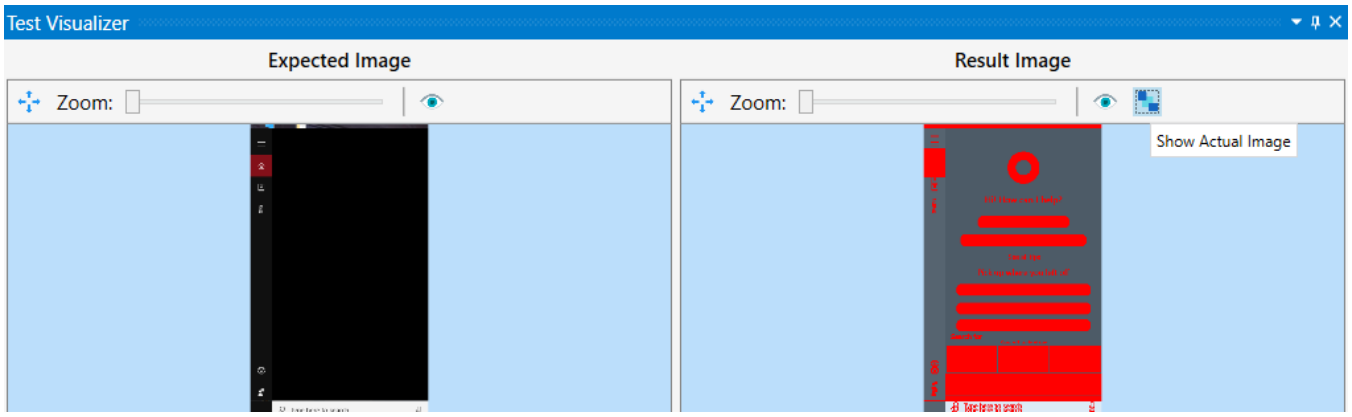


Figure 77. Show Result Image screen design

Sequence diagram

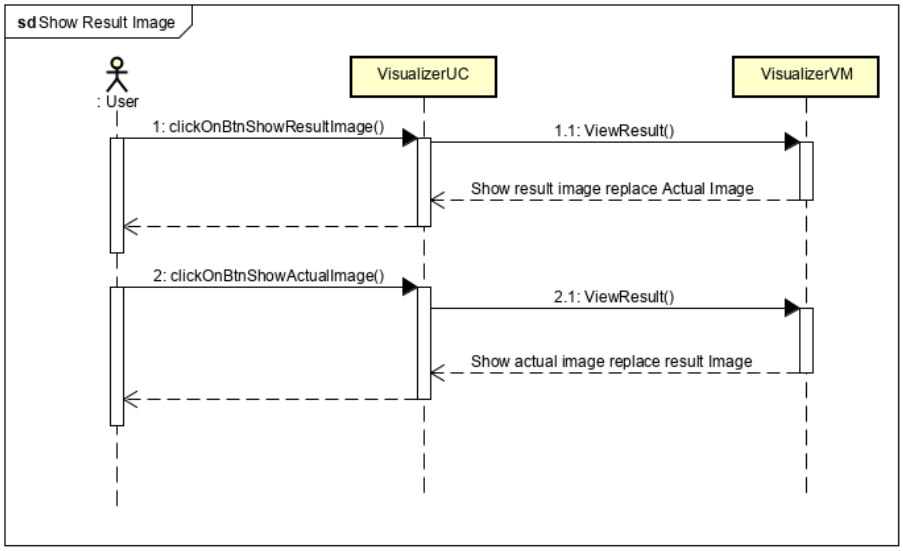


Figure 78. Show Result Image sequence diagram

4.5.3.5 Image Viewer

4.5.3.5.1 View Single Image

Screen design

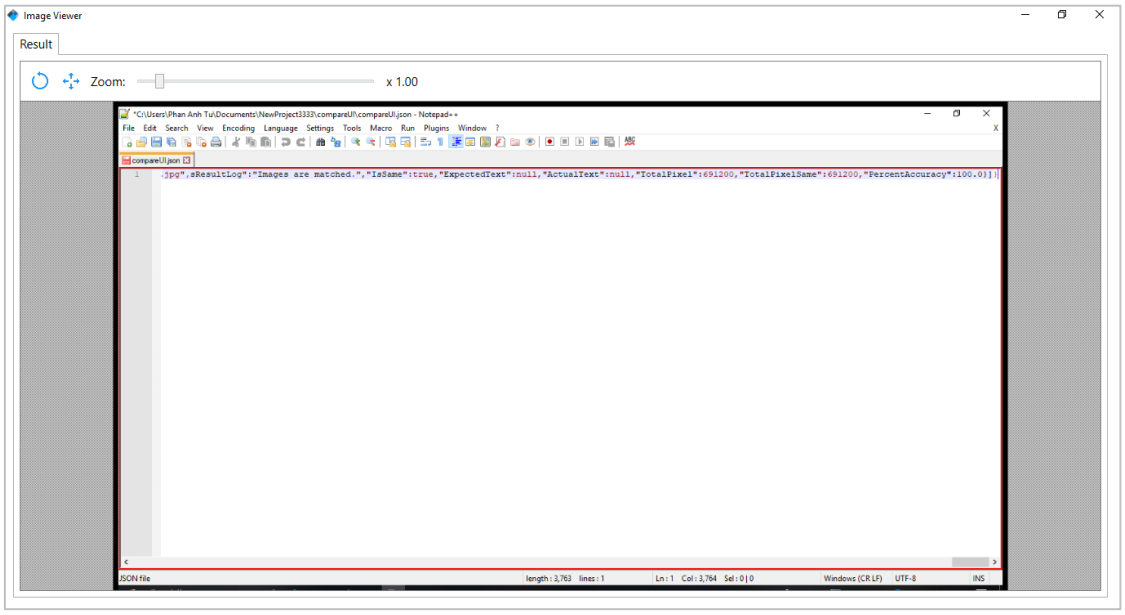


Figure 79. View Single Image screen design

Sequence diagram

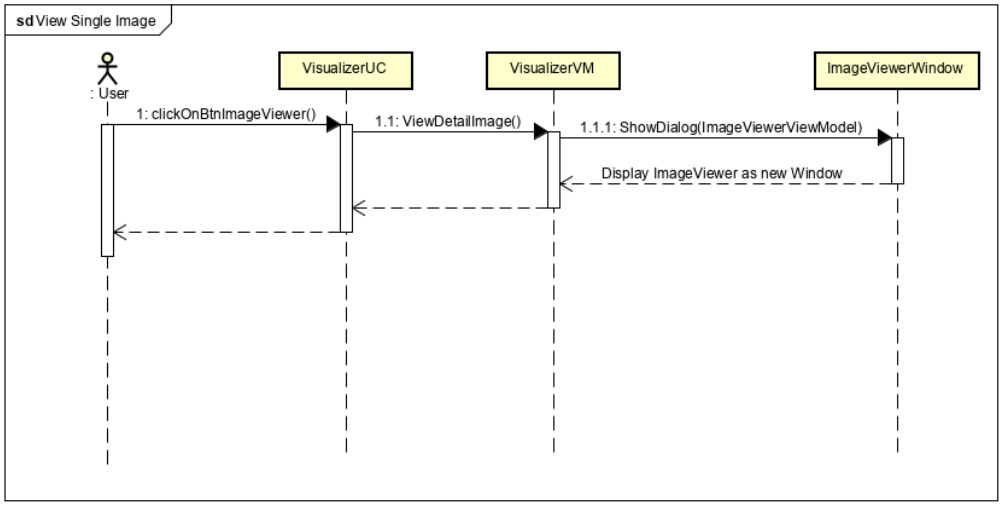


Figure 80. View Single Image sequence diagram

4.5.3.5.2 Show Original Image Size

Screen design

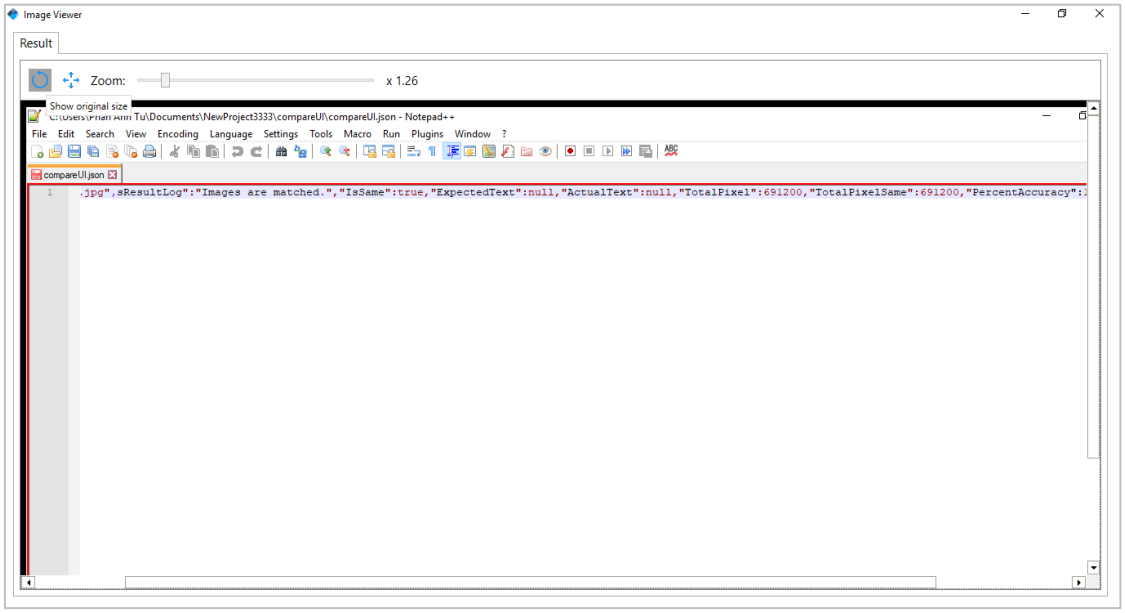


Figure 81. Show Original Image Size screen design

Sequence diagram

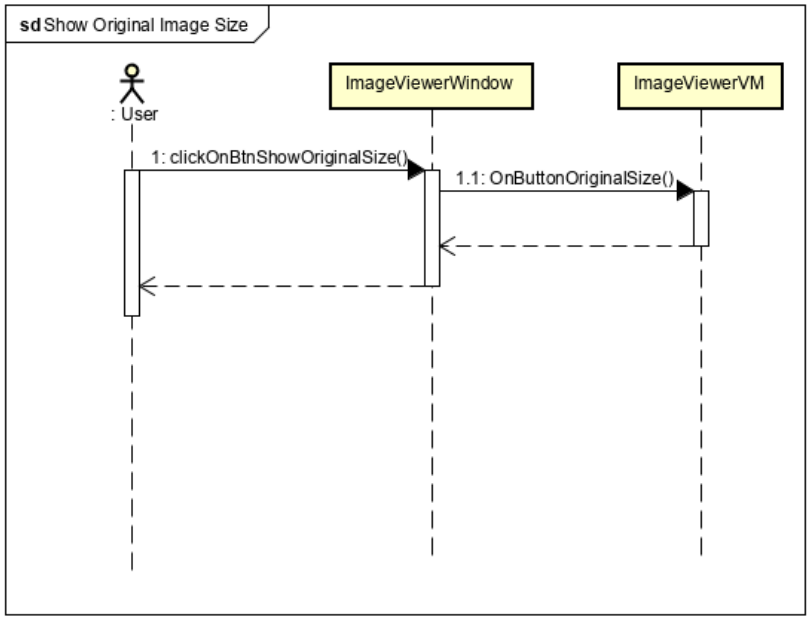


Figure 82. Show Original Image Size sequence diagram

4.5.3.5.3 Open UI Comparison Module

Screen design

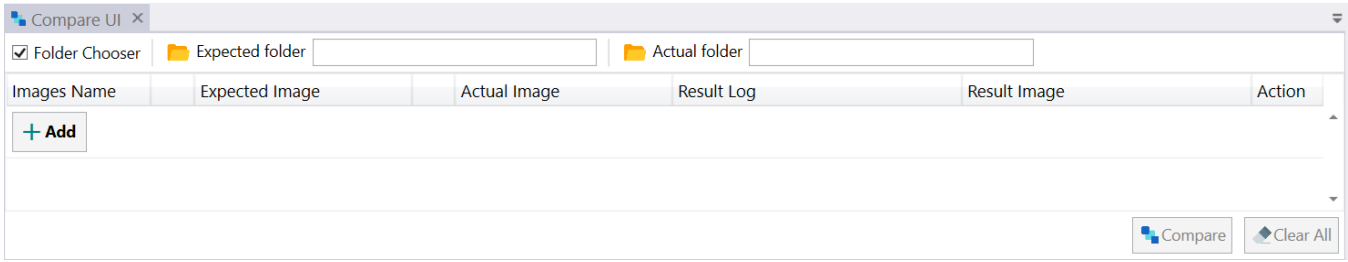


Figure 83. Open UI Comparison Module screen design

Sequence diagram

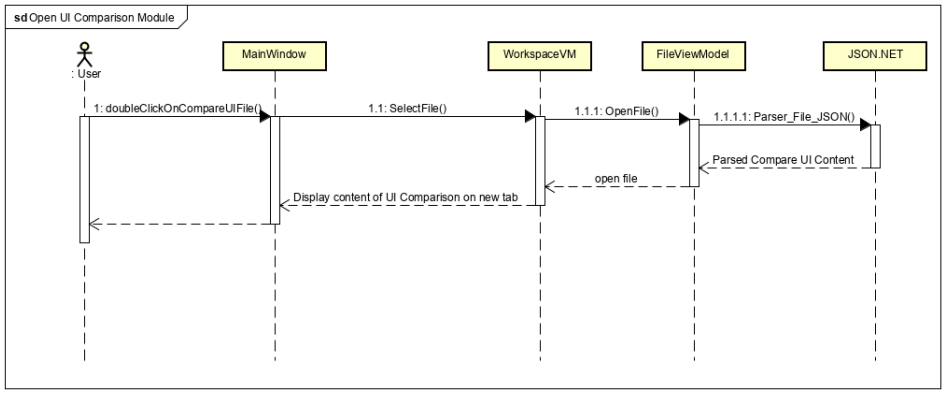


Figure 84. Open UI Comparison Module sequense diagram

4.5.3.6 Compare UI

4.5.3.6.1 Filter For Folder Chooser

Screen design

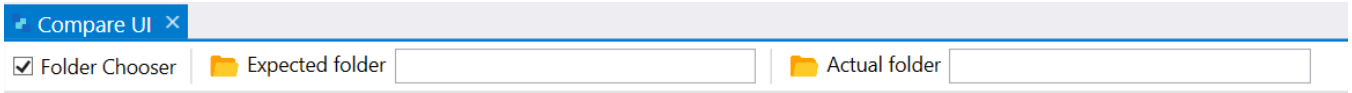


Figure 85. Filter For Folder Chooser screen design

Sequence diagram

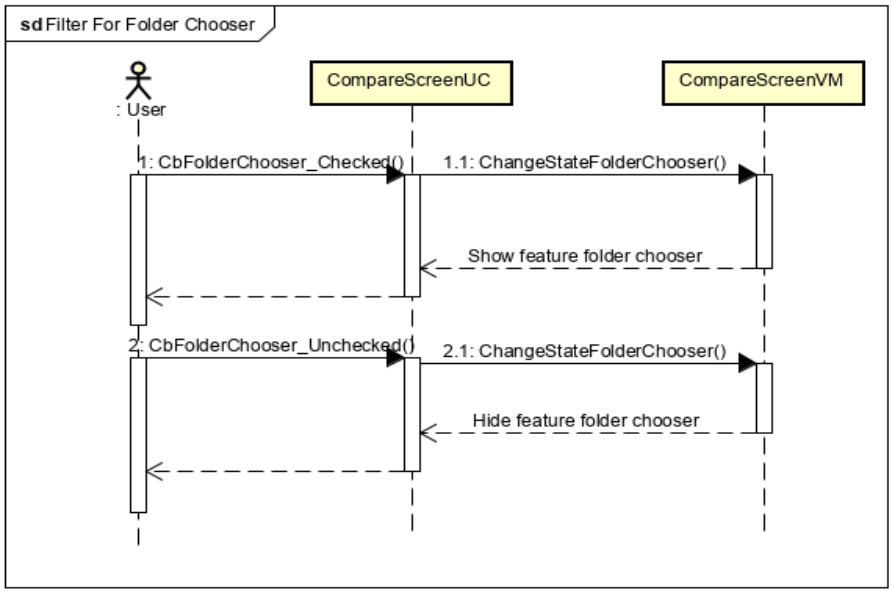


Figure 86. Filter For Folder Chooser sequence diagram

4.5.3.6.2 Select Expected Image Folder

Screen design

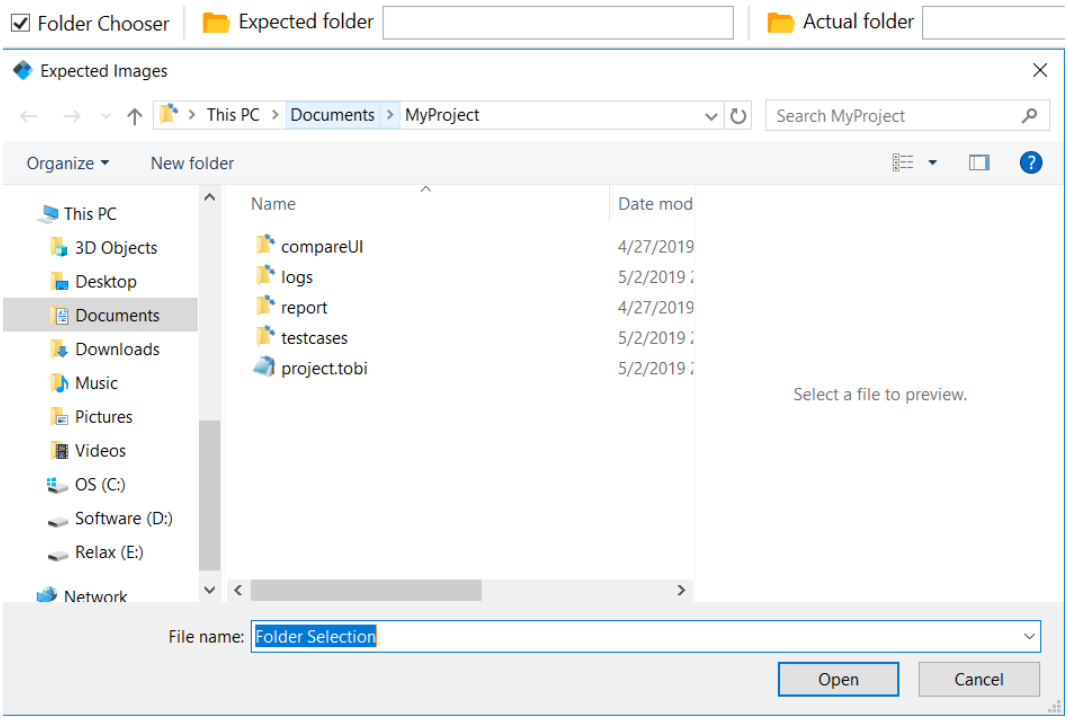


Figure 87. Select Expected Image Folder screen design

Sequence diagram

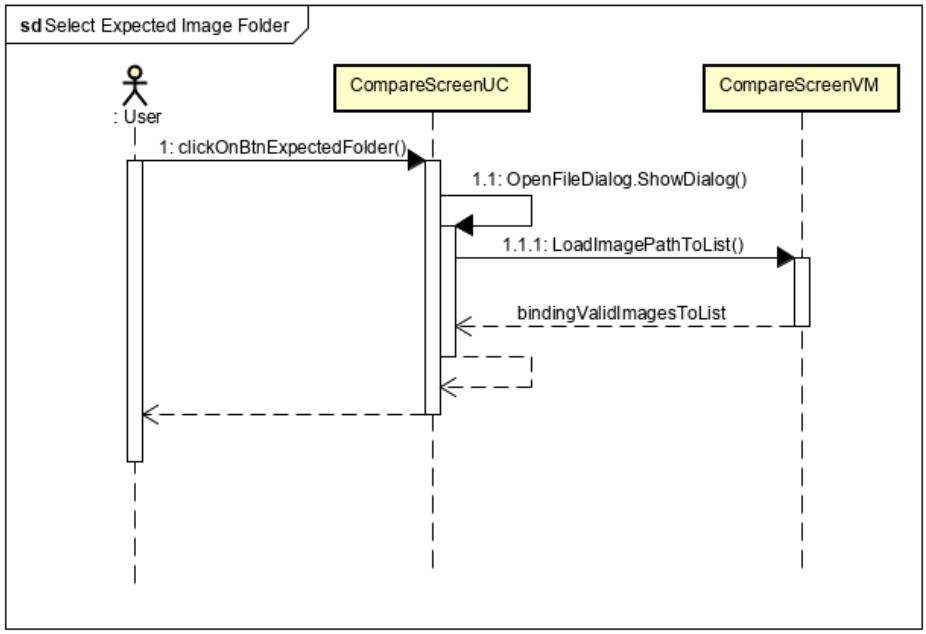


Figure 88. Select Expected Image Folder sequence diagram

4.5.3.6.3 Input Expected Image Folder Path

Screen design

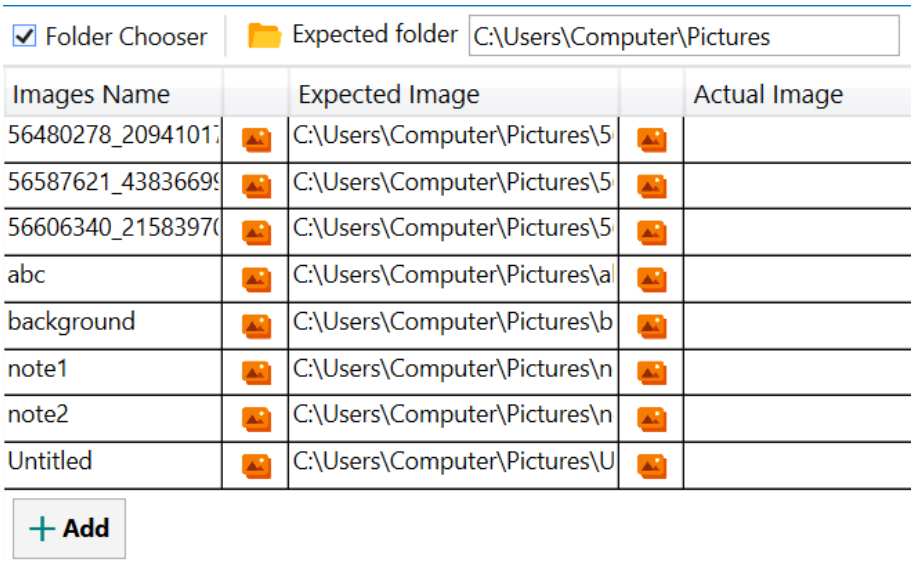


Figure 89. Input Expected Image Folder Path screen design

Sequence diagram

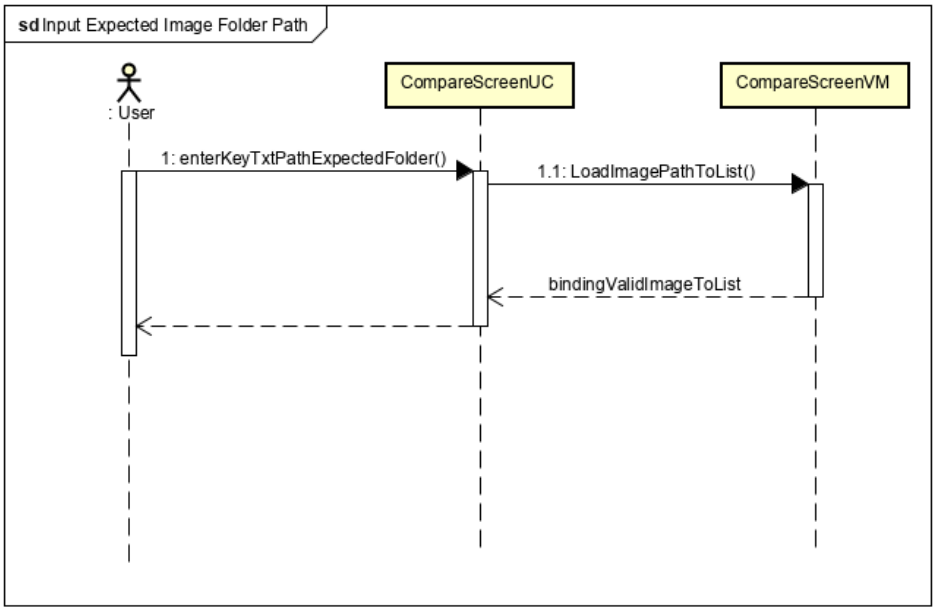


Figure 90. Input Expected Image Folder Path sequence diagram

4.5.3.6.4 Select Actual Image Folder

Screen design

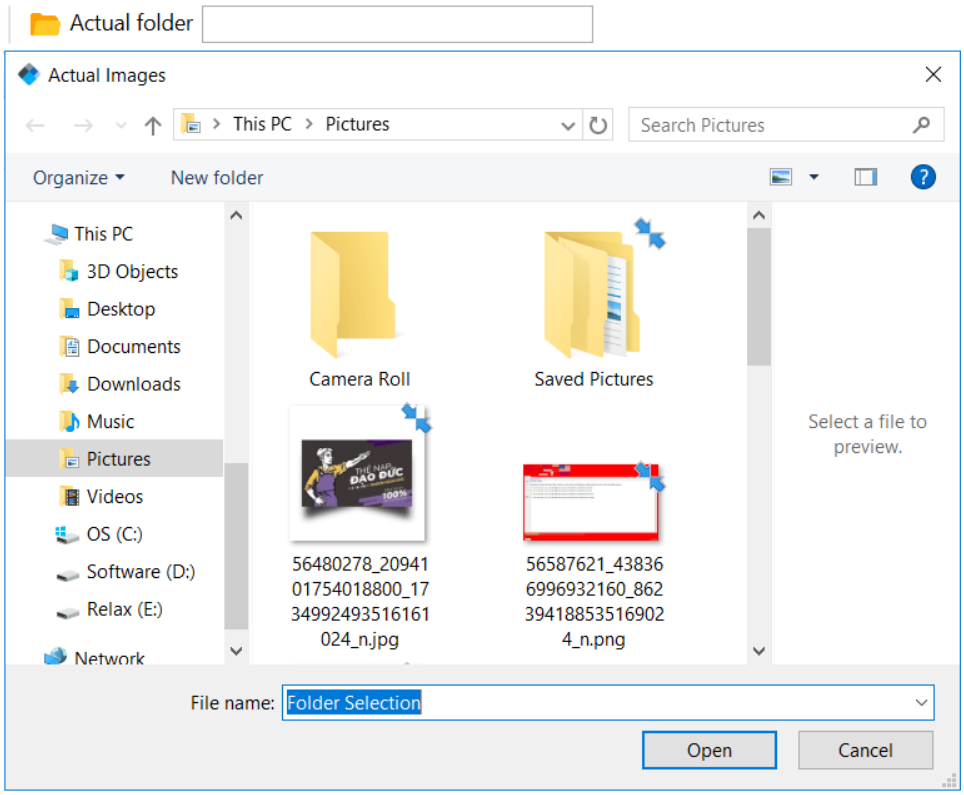


Figure 91. Select Actual Image Folder screen design

Sequence diagram

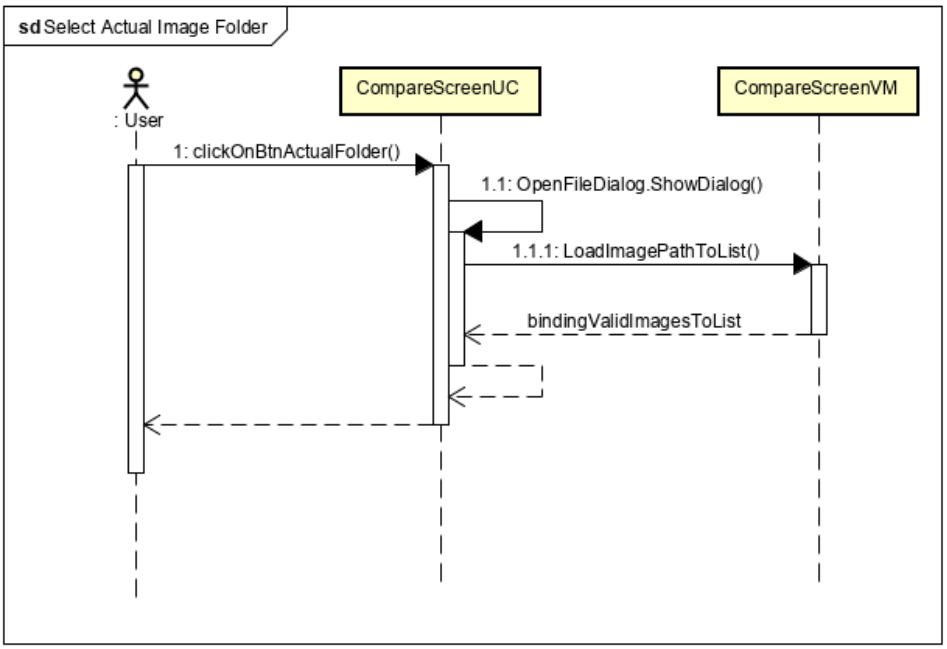


Figure 92. Select Actual Image Folder sequence diagram

4.5.3.6.5 Input Actual Image Folder Path

Screen design

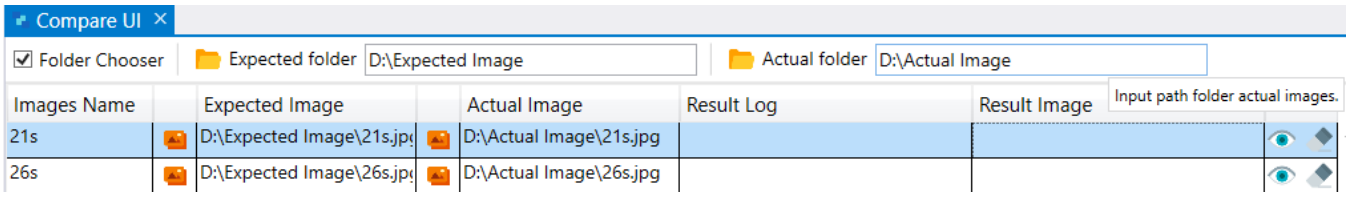


Figure 93. Input Actual Image Folder Path screen design

Sequence diagram

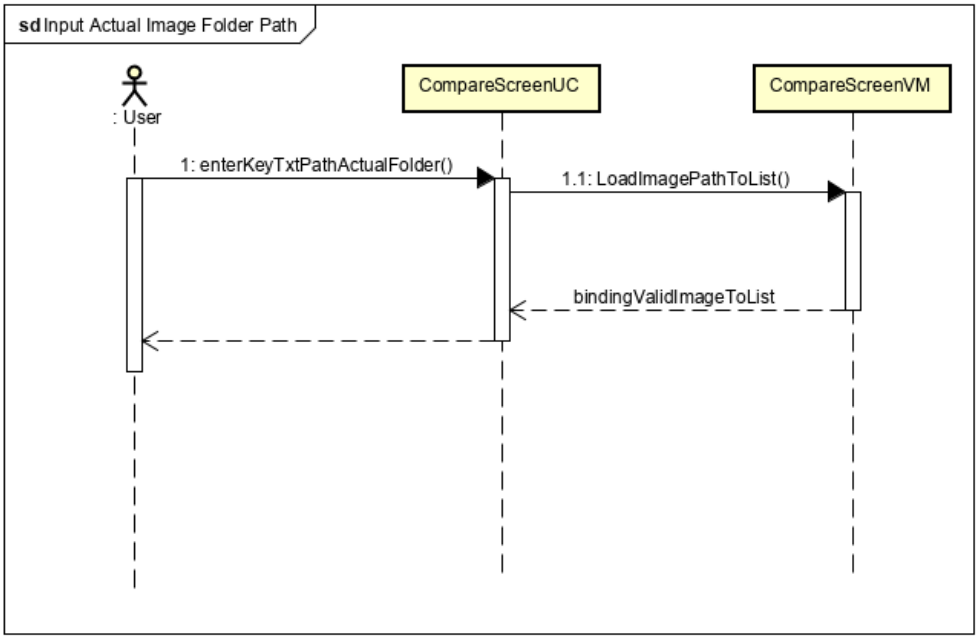


Figure 94. Input Actual Image Folder Path sequence diagram

4.5.3.6.7 Add Manual Compared Images

Screen design

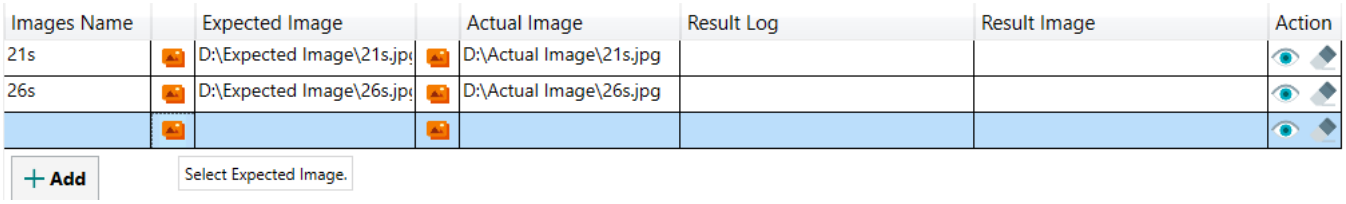


Figure 95. Add Manual Compared Images screen design

Sequence diagram

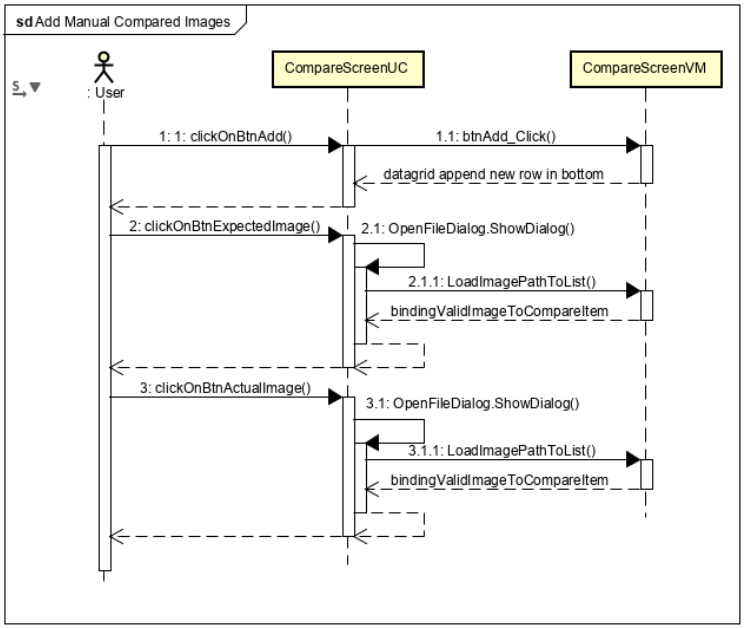


Figure 96. Add Manual Compared Images sequence diagram

4.5.3.6.8 Clear All Images

Screen design

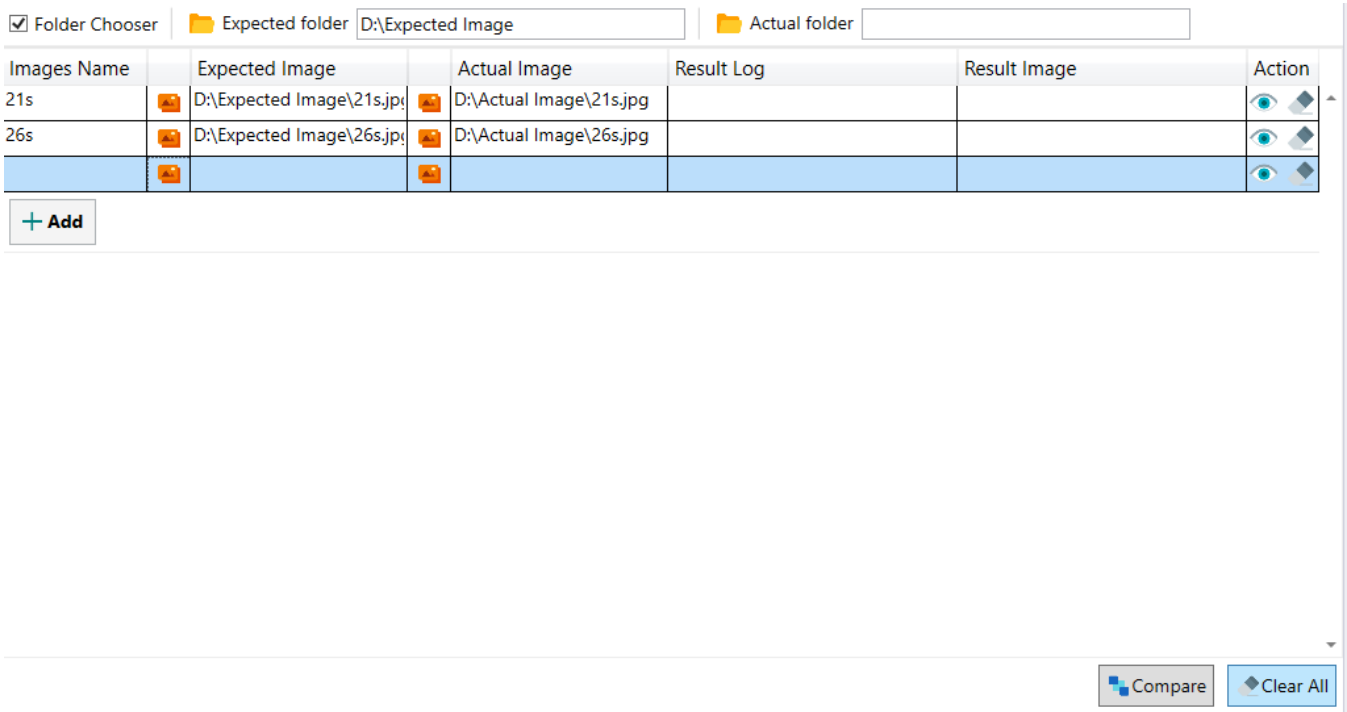


Figure 97. Clear All Images screen design

Sequence diagram

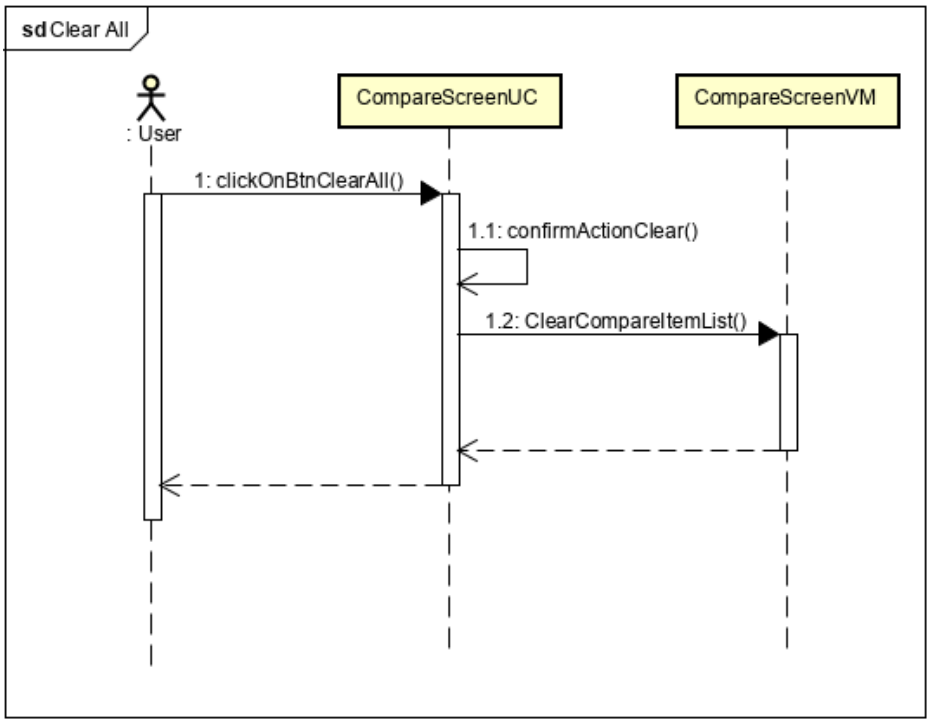


Figure 98. Clear All Images screen design

4.5.3.6.9 Compare All Images

Screen design

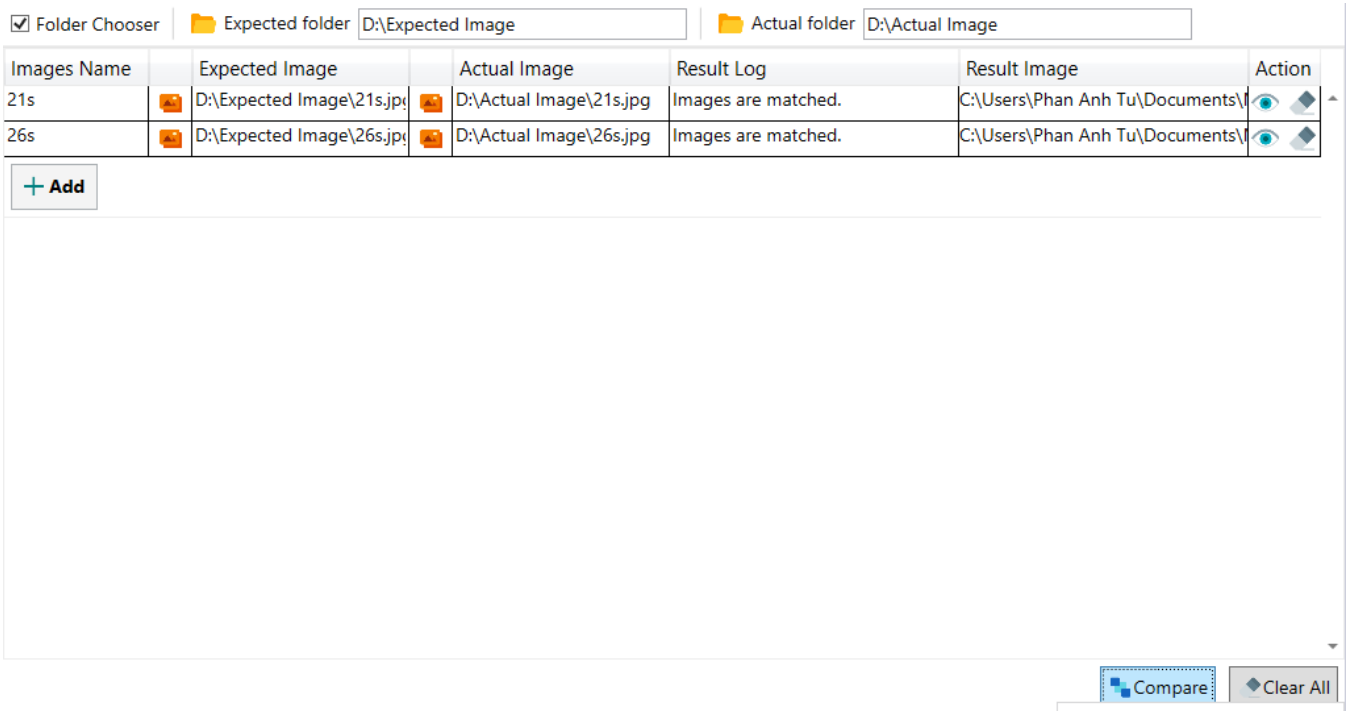


Figure 99. Compare All Images screen design

Sequence diagram

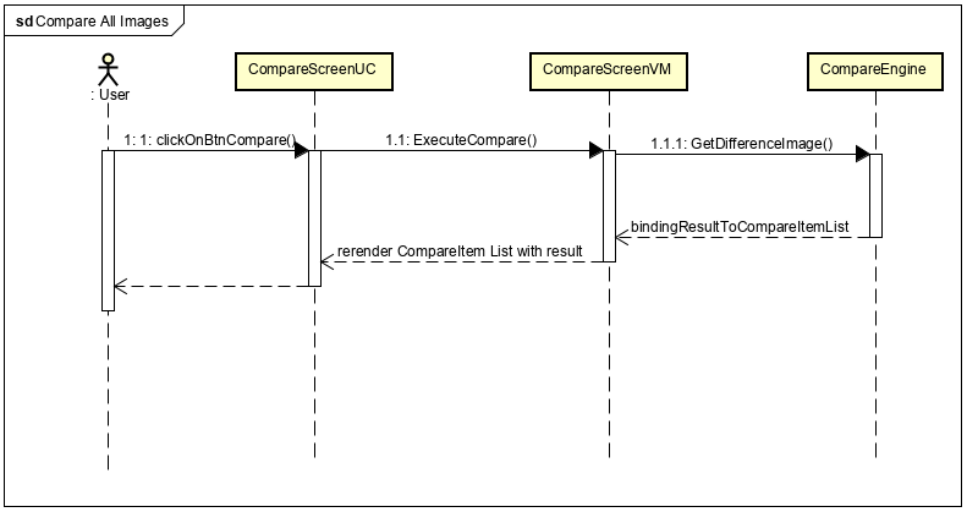


Figure 100. Compare All Images sequence diagram

4.5.3.6.10 View Image Pair In Image Viewer

Screen design

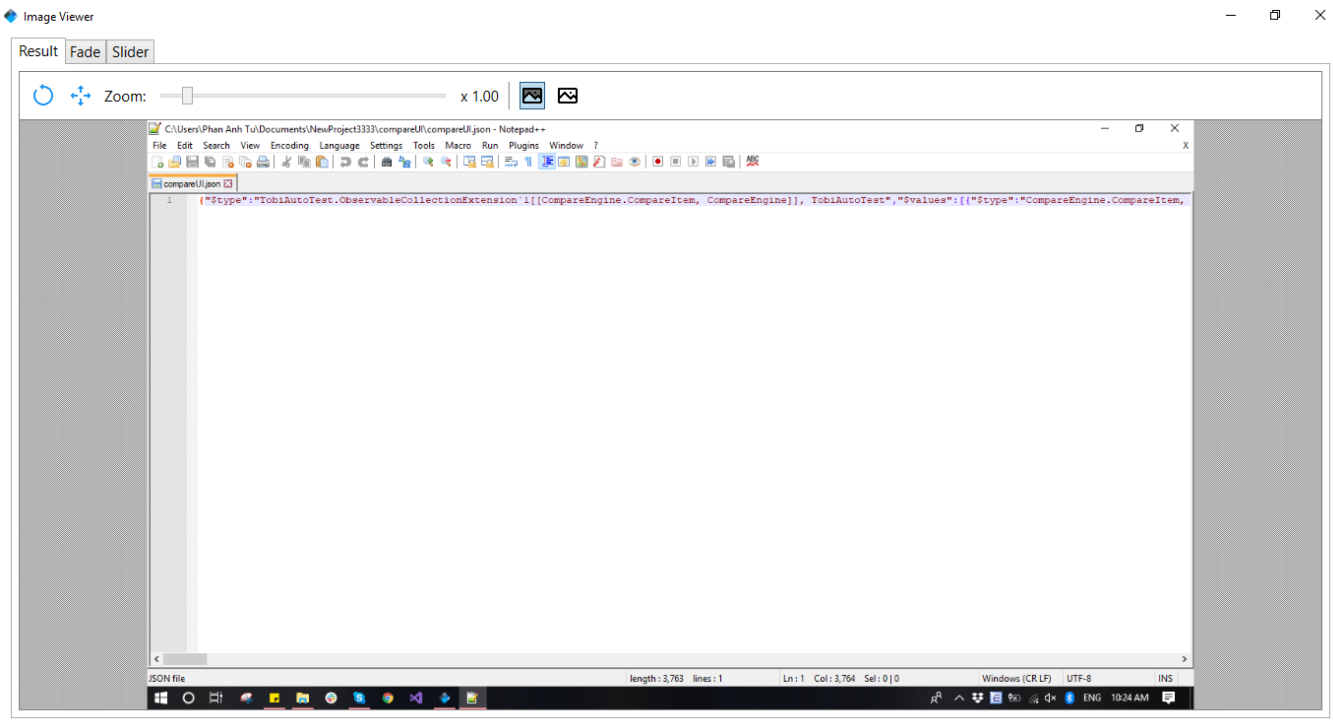


Figure 101. View Image Pair in Image Viewer screen design

Sequence diagram

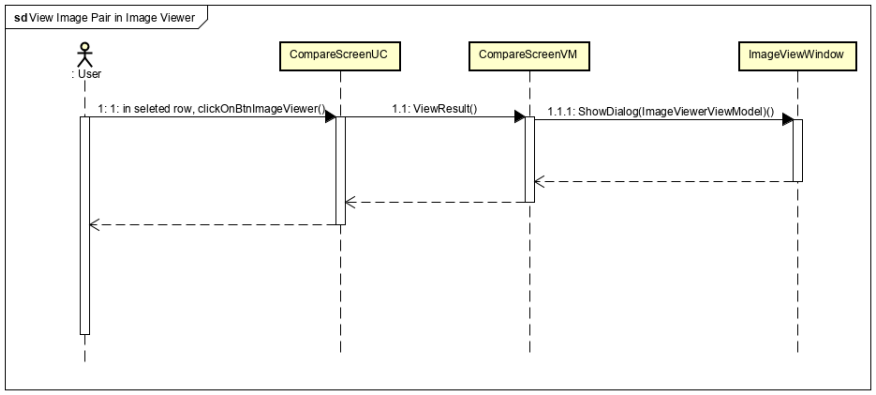


Figure 102. View Image Pair in Image Viewer sequence diagram

4.5.3.6.11 Delete Image Pair

Screen design

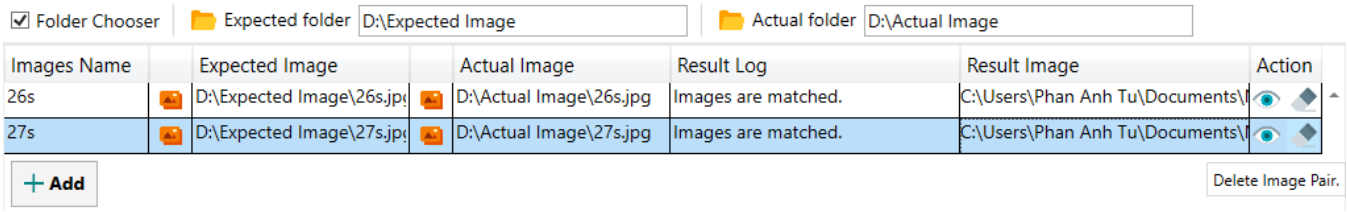


Figure 103. Delete Image Pair screen design

Sequence diagram

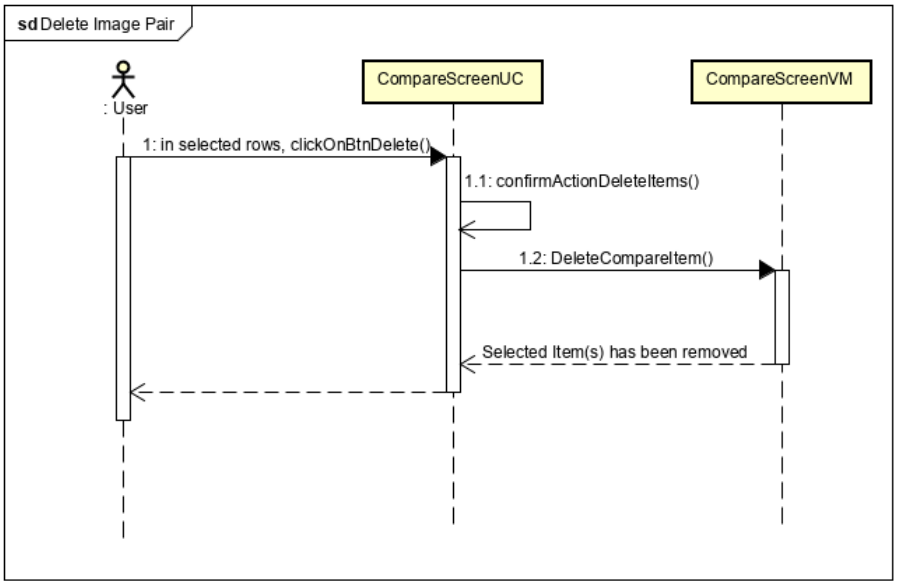


Figure 104. Delete Image Pair sequence diagram

4.5.3.7 App Settings

4.5.3.7.1 Setting General Preferences

Screen design

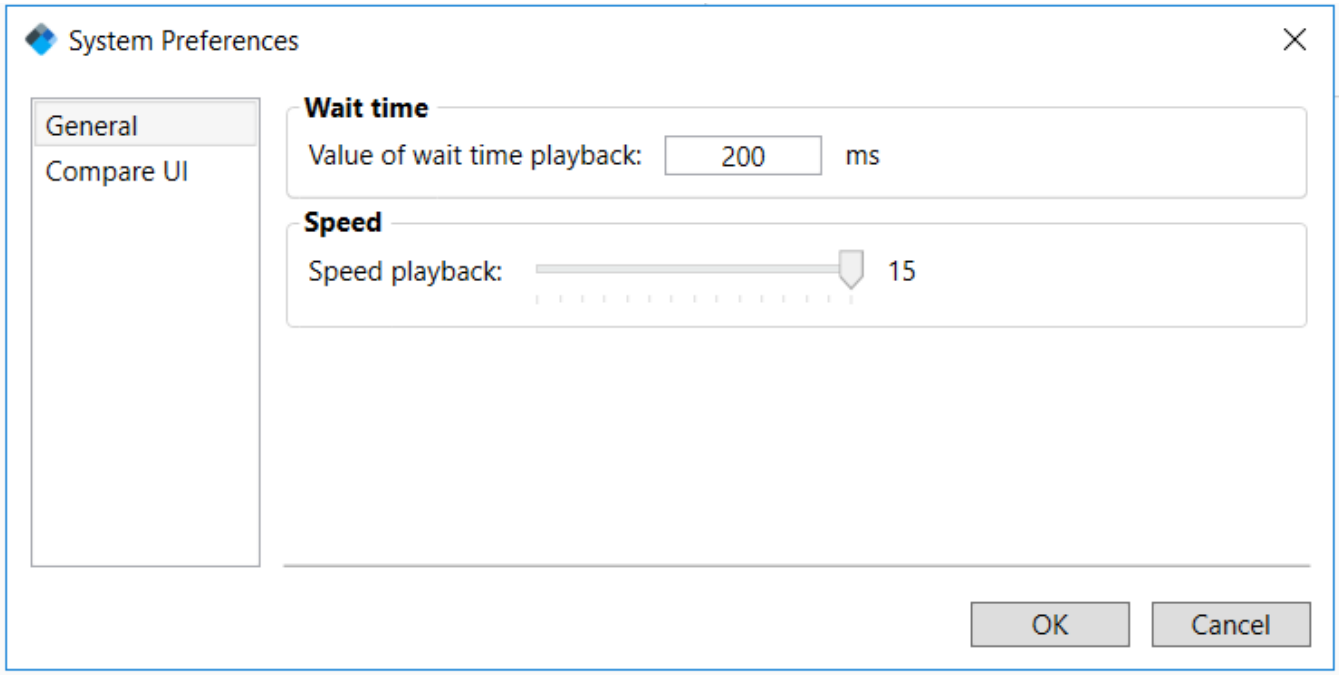


Figure 105. Setting preferences general screen design

Sequence diagram

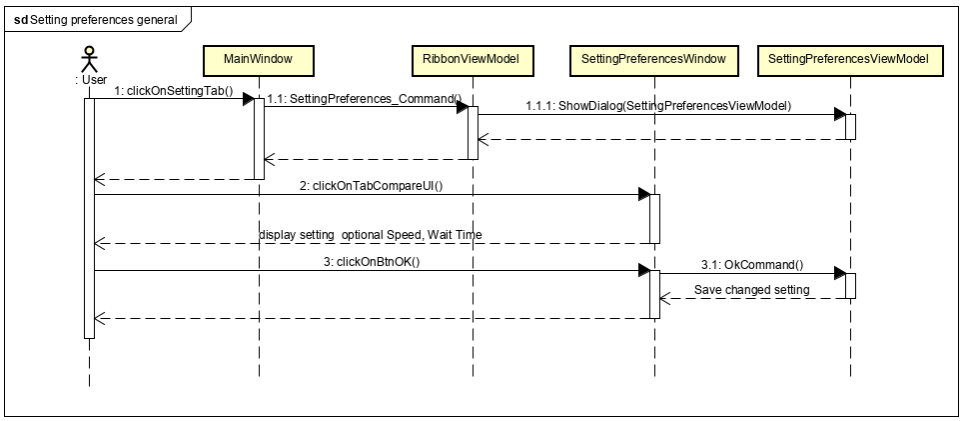


Figure 106. Setting preferences general sequence diagram

4.5.3.7.2 Setting Compare UI Preferences

Screen design

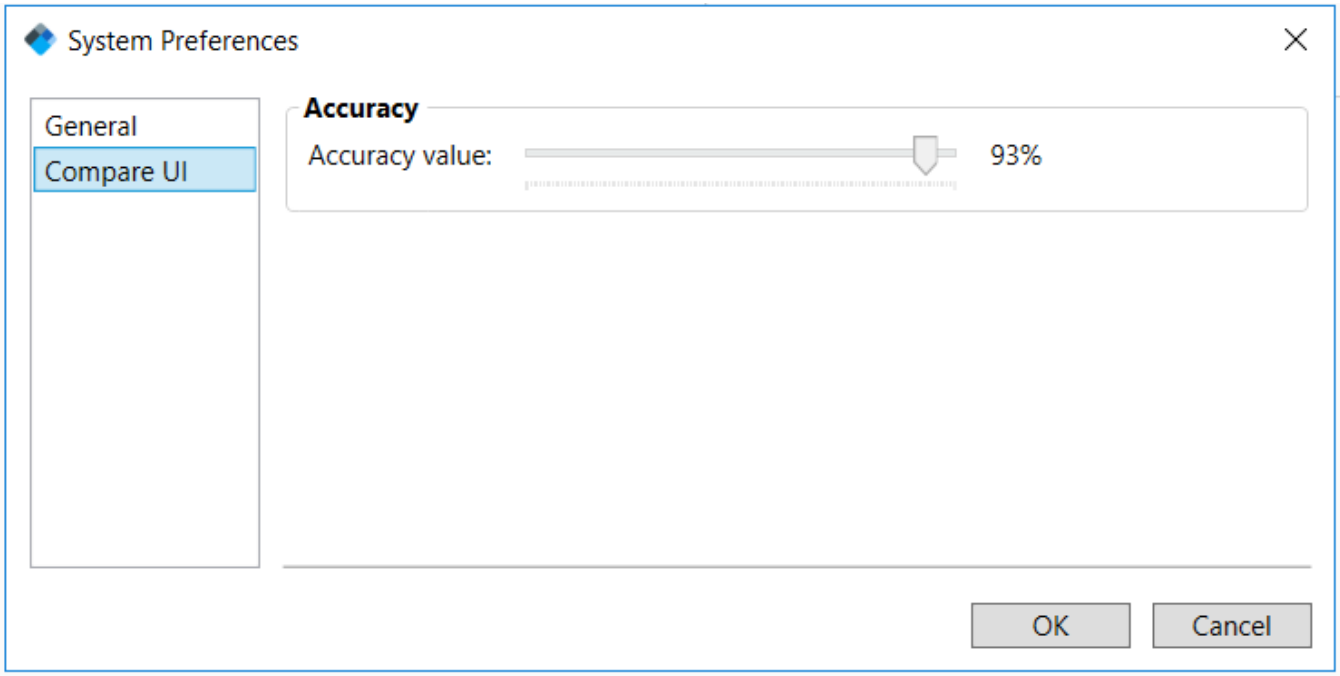


Figure 107. Setting preferences Compare UI screen design

Sequence diagram

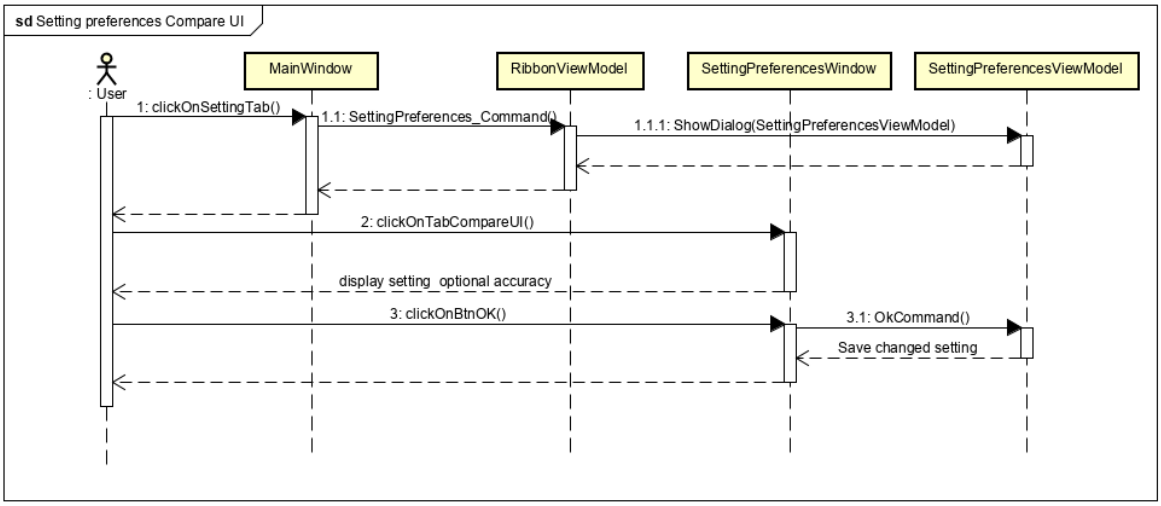


Figure 108. Setting preferences Compare UI sequence diagram

4.5.3.8 Report

4.5.3.8.1 View Report

Screen design

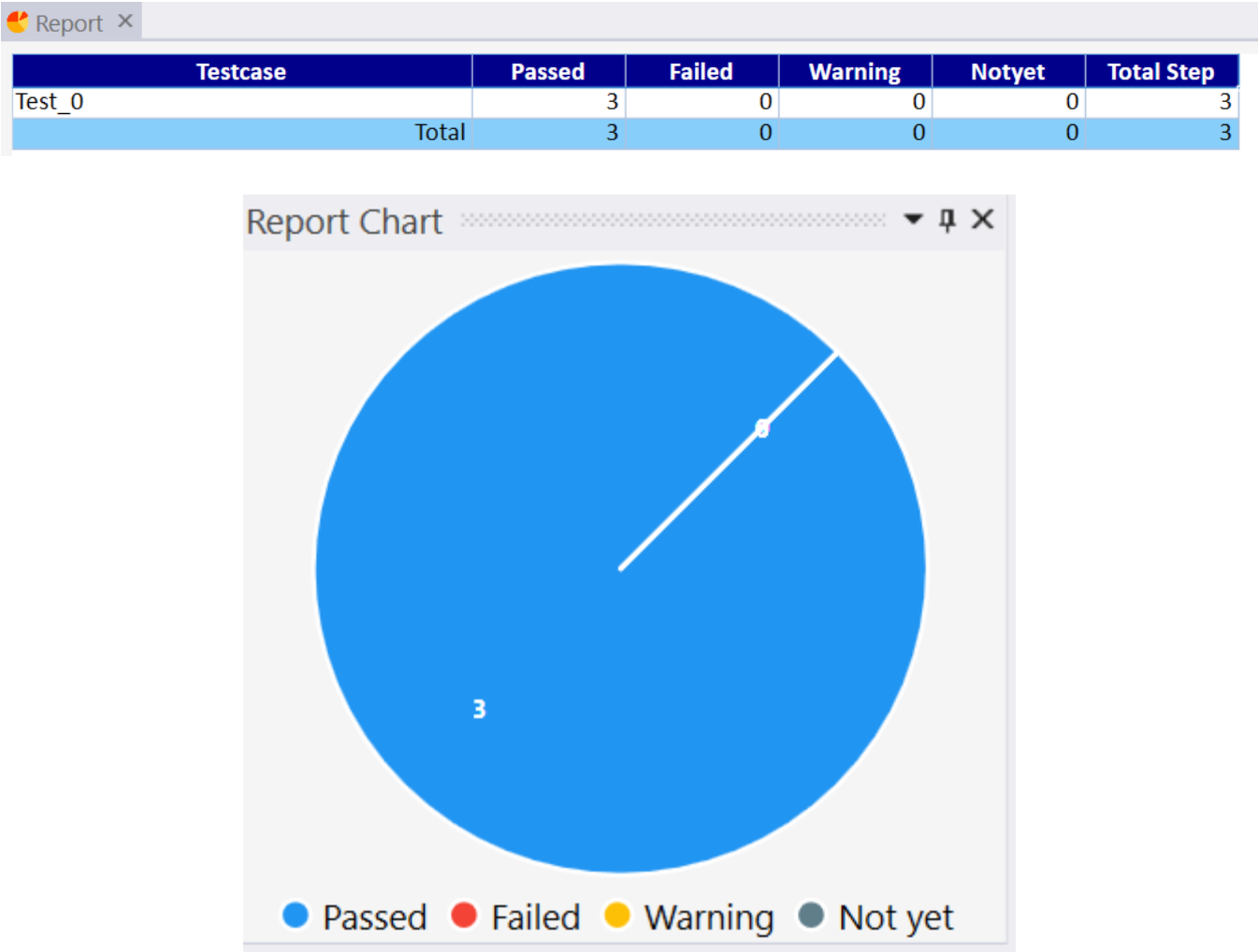


Figure 109. View "Report" screen design

Sequence diagram

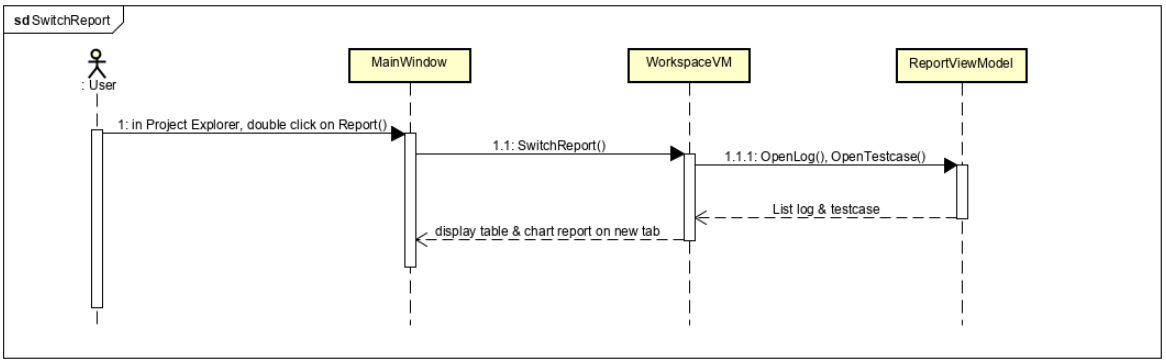


Figure 110. View " Report" sequence diagram

4.5.3.8.2 Export To Excel

Screen design

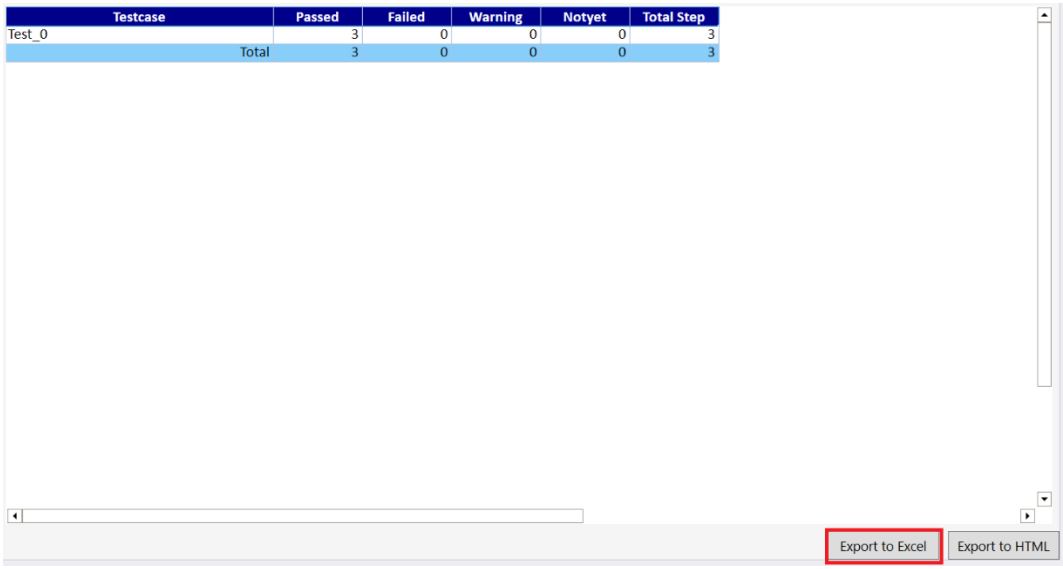


Figure 111. Export to Excel design

Sequence diagram

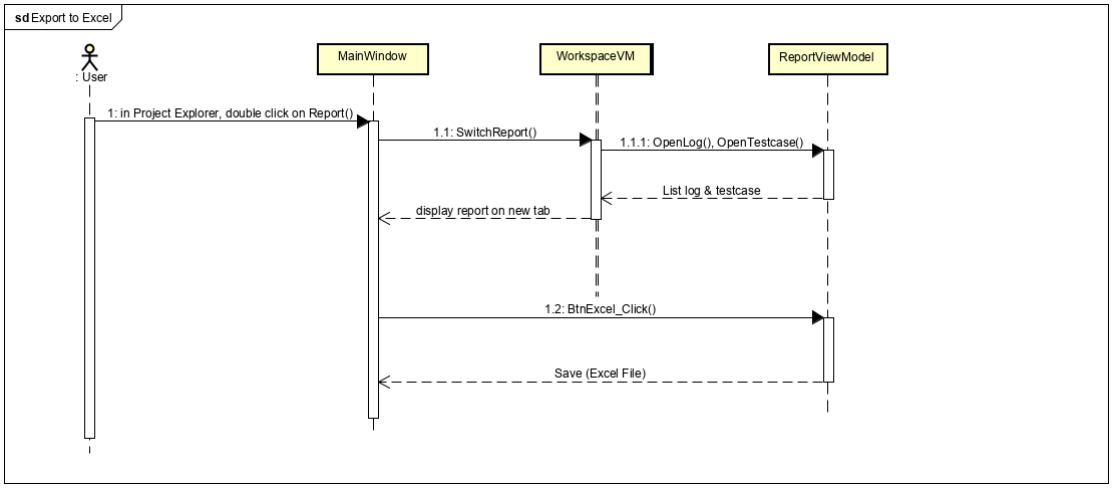


Figure 112. Export to Excel sequence diagram

4.5.3.8.3 Export To HTML

Screen design

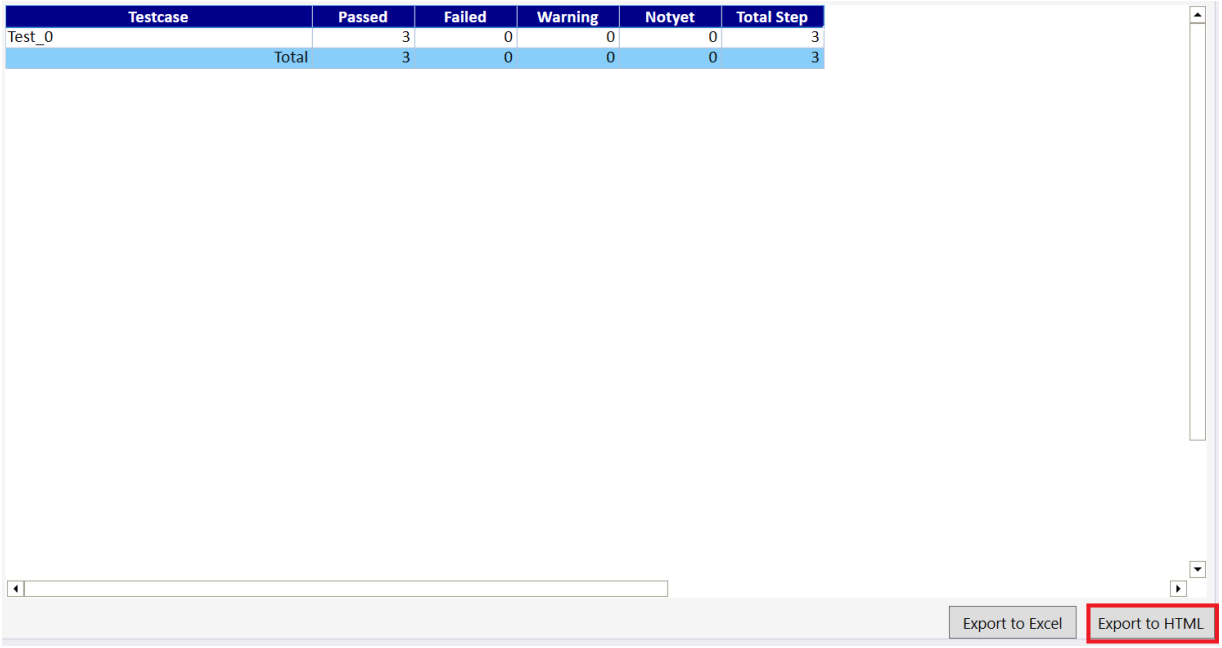


Figure 113. Export to HTML screen design

Sequence diagram

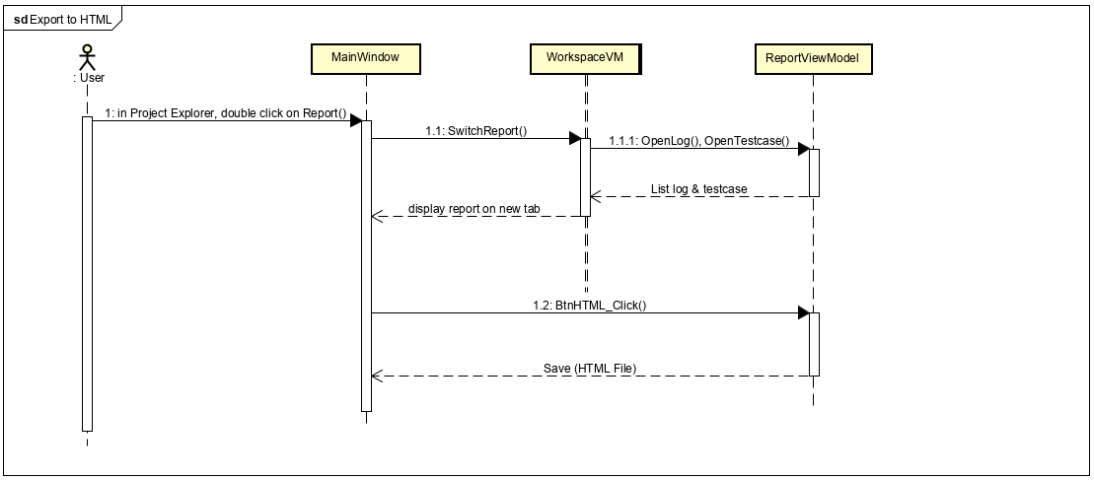


Figure 114. Export to HTML sequence diagram

4.5.3.9 Other Features

4.5.3.9.1 Reset Layout

Screen Design

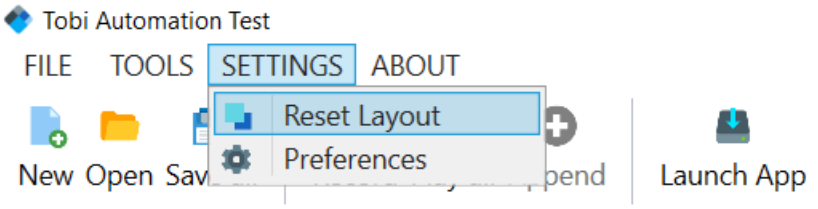


Figure 61.Reset Layout screen design

Sequence Diagram

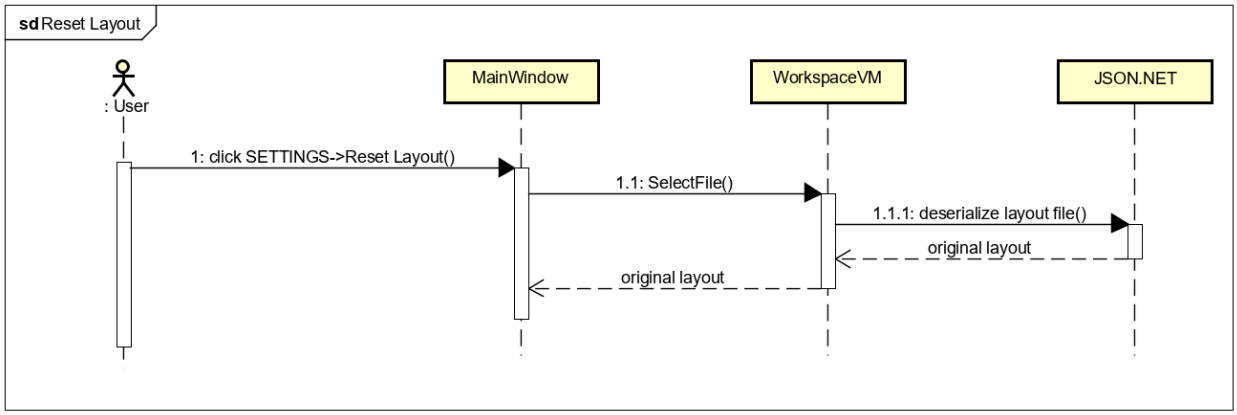


Figure 62.Reset Layout sequence diagram

4.5.3.9.2 View About System

Screen Design

Figure 63.View About System screen design

Sequence Diagram

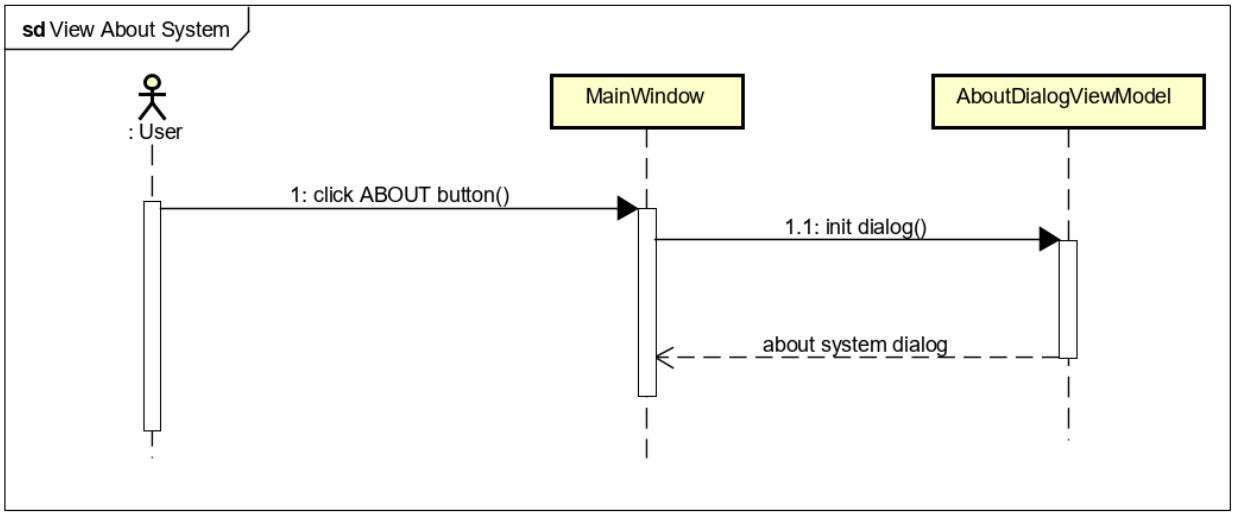


Figure 64.View About System sequence diagram

4.5.3.9.3 Launch Apps

Screen Design

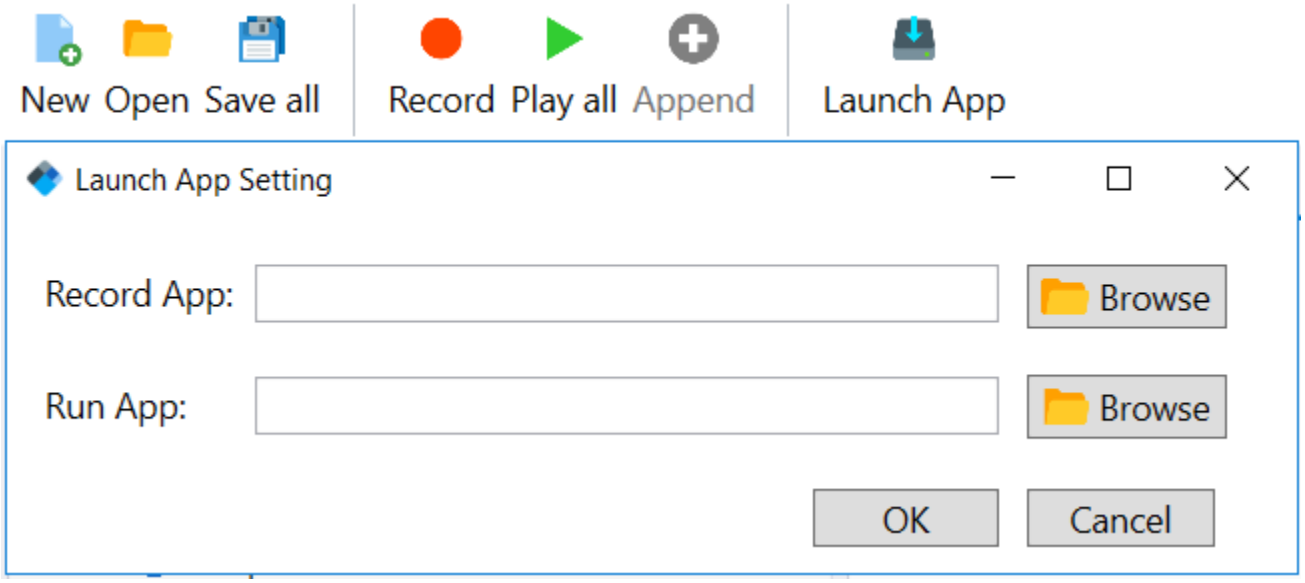


Figure 63.Launch Apps screen design

Sequence Diagram

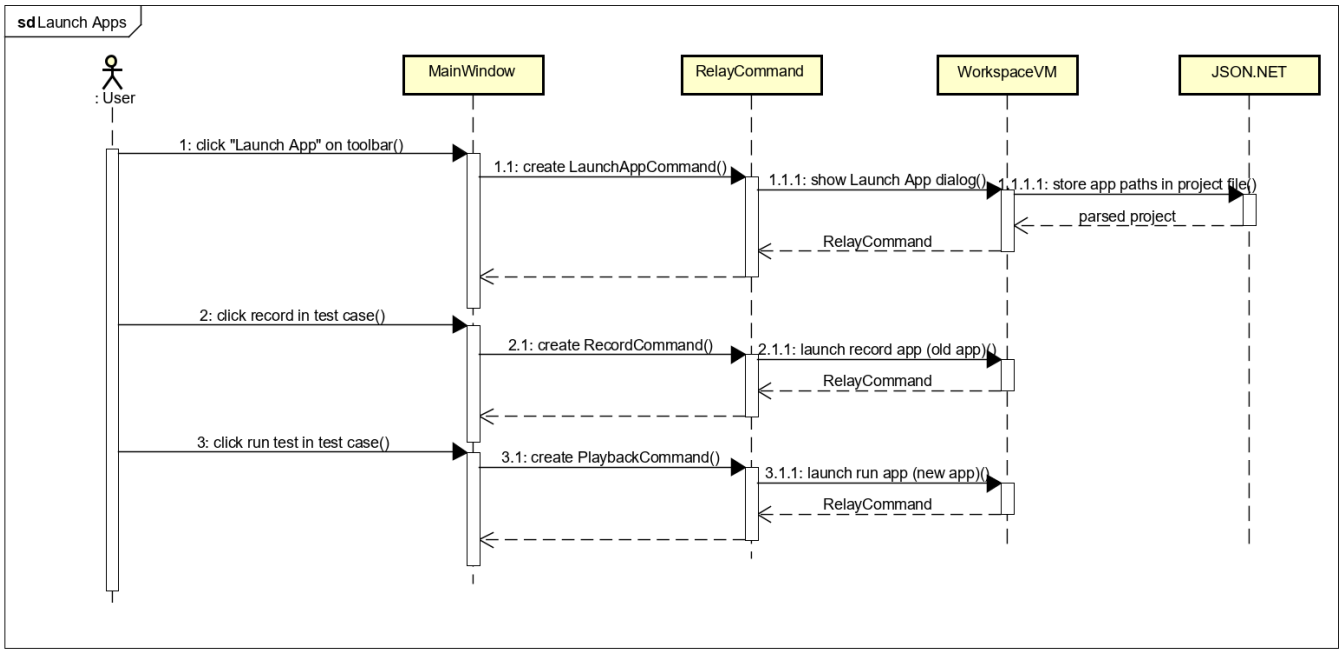


Figure 64.Launch Apps sequence diagram

CHAPTER 5 – SOFTWARE TESTING DOCUMENTATION

5.1. Purpose

This chapter purpose is to detect software failures so that defects can be discovered and corrected to ensure that our project is thoroughly tested and resulting in a successful implementation of application that we are developing. It contains the following sections:

- Scope of testing.
- Testing tools and environment.
- Resources and responsibilities.
- Test strategy: Test approach, test stages.
- Features to be tested.
- Features not to be tested.
- Defects log
- Test report

5.2. Scope of testing

5.2.1. Stages of testing

There are 4 phases in testing process: Unit testing, Integration testing, System testing and Acceptance testing.

No.	Test stages	Description	Implementer
1	**Unit testing**	Unit testing will be done by developers and approved by team leader.	Developers
2	**Integration testing**	Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group. This testing	Tester

		level is designed to find interface defects between modules/functions. It is particularly beneficial because it determines how efficiently the units are running together.	
3	**System testing**	System testing is the first level in which the complete application is tested as a whole. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards or not.	Testers, Technical leader
4	**Acceptance test**	Acceptance testing will be executed by all team members in view of end-user. It determines whether system satisfies requirements specified in the requirement analysis phase. Finding defect is not the main task in this stage. Acceptance test will access the system's readiness for deployment and usage.	End-users

Table 78. Stages of testing

5.2.2. Types of testing

No.	Test types	Description
1	**Function test**	• Functional testing is a type of software testing whereby the system is tested against the functional requirements or specifications. • Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. • During functional testing, Black box testing technique is used in which the internal logic of the system being tested is not known to the tester.
2	**GUI test**	• GUI testing verifies a user's interaction with the software. The goal of GUI testing is to ensure that the GUI provides the user with an appropriate access and navigation through the functions of the target-of-test. In addition, GUI testing ensures that the objects within the GUI function as expected and conform to requirement • GUI test will be performed fully on all screens. • This test targets to cover the verification of the overall look and feel of the TOBI Tool including initial position, font, text size, color, focus, initial button, tab order, label, screen sizes and sentences width.

		• Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, it must be able to provide inputs to the input fields. • Check if error messages are displayed correctly. • Check if fonts used in application is readable. • Check if the alignment of the text is proper. • Check if the color of the font and warning messages is aesthetically pleasing. • Check if images have good clarity and are properly aligned. • Check the positioning of GUI elements for different screen resolution..
3	**Regression test**	• The testing is to confirm that the bug was removed including the extent of the impact, when developers fix bug, developers and testers will confirm with each other what is the impacts of fix bug modification, after that all impact unit or function will be retested by developers then testers after that

Table 79. Types of testing

5.2.3. Range of testing

Team performs all functions defined in the SRS based on the approved version.

No	Testing level	Description	Implementation
1	Unit test	• **Unit test is performed by developers**. • **<u>All the test cases are NOT documented.</u>**	Development team: 1. Ngo Ngoc Lam 2. Phan Anh Tu 3. Nguyen Dinh Dong 4. Len Van Hau
2	Integration test	• **Integration testing is performed by testers.** Material are "TOBI_TestCase_Final.xlsx".	Tester team: 1. Nguyen Thi Giang 2. Phan Anh Tu 3. Len Van Hau
3	System test	• **System test is performed by testers**. • Material are "TOBI_TestCase_Final.xlsx".	Tester team: 1. Nguyen Thi Giang 2. Phan Anh Tu 3. Len Van Hau
4	Regression test	• **Regression test is performed by testers** • Material are "TOBI_TestCase_Final.xlsx".	Tester team: 1. Nguyen Thi Giang 2. Phan Anh Tu 3. Len Van Hau
5	Acceptance test	• **Acceptance test is performed by End user.** • Material are "Funtional Requirement", "Acceptance test" • **<u>All the acceptance test cases are not documented.</u>**	End user: 1. Nguyen Manh Hieu

5.3. Test plan

5.3.1. Testing tools and environment

5.3.3.1. Testing tools

- Microsoft Excel 2013: Excel is used to manage test cases and manage bugs.
- Visual studio: Unit test.

5.3.3.2. Testing environment

a. Software

Category	Test machine
OS	Windows 10,64 bit
Screen size	15.6 inch
Screen resolution	1366x768
OS Language	English
Database	N/A
Framework	.Net Framework 4.5.2
Dependency	Microsoft Visual C++ Redistributable for Visual Studio 2017

5.3.2. Resources and responsibilities

No.	Resources	Responsibilities
1	Project manager	• Reviewing test case and report.
2	Tester	• Performing the actual system testing. • Managing test resource and assign test tasks. • Creating test plan. • Creating test cases. • Creating test report. • Executing test and testing log report.
3	Developer	• Creating unit test. • Fixing bugs

Table 80. Resources and responsibilities

5.3.3. Test strategy

5.3.3.3. Test model

TOBI follows Iterative Model process to implement testing. The Iterative model is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. In short, iterative development is a way of breaking down the software development of a large application into smaller pieces.

It starts with a simple implementation of a small set of the software requiresments, which iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed. Each release of the model is developed in a fix time period and is called iteration.

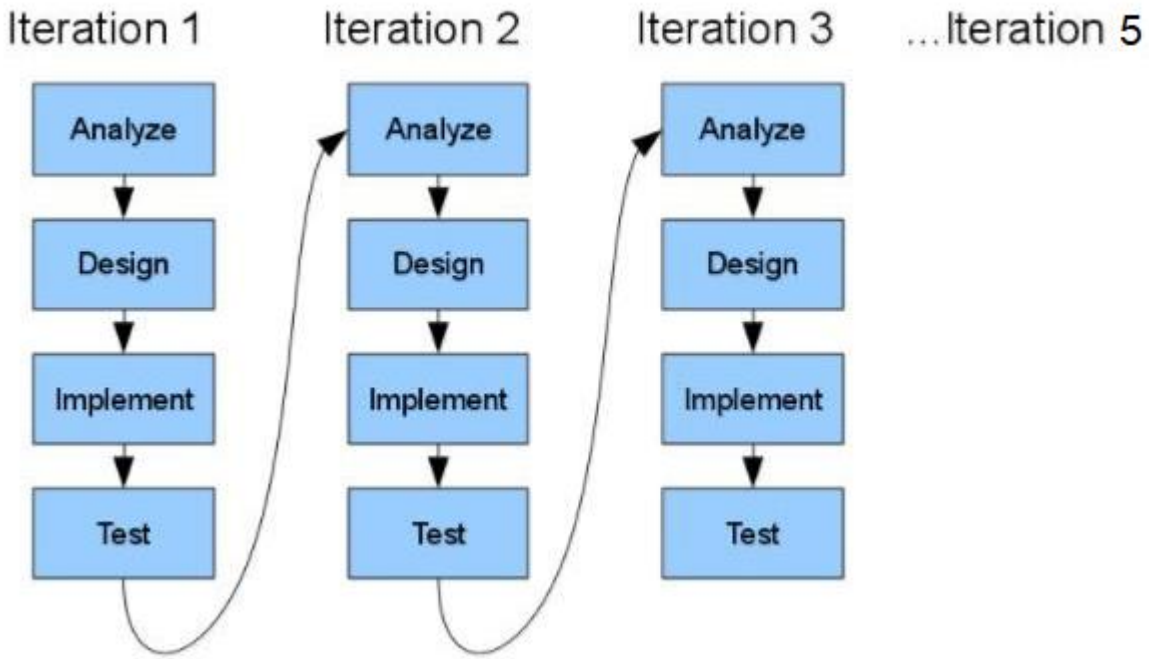


Figure 1. Iterative model of testing

TOBI uses Iterative model because:

- Some working functionality can be developed and early in the software development life cycle (SDLC).
- It is easily adaptable to the ever changing needs of the project as well as the client.
- It is best suited for agile organisations.
- It is more cost effective to change the scope or requirements in Iterative model.
- Parallel development can be planned.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed.
- In iterative model less time is spent on documenting and more time is given for designing.

- One can get reliable user feedback, when presenting sketches and blueprints of the product to users for their feedback.

5.3.3.4. Test stage

Types of test	Stages of test			
	Unit test	Integration test	System test	Acceptance test
Function test	✓	✓	✓	✓
Regression test		✓	✓	✓
GUI test			✓	✓

Table 31. Test stage

5.3.3.5. Test process

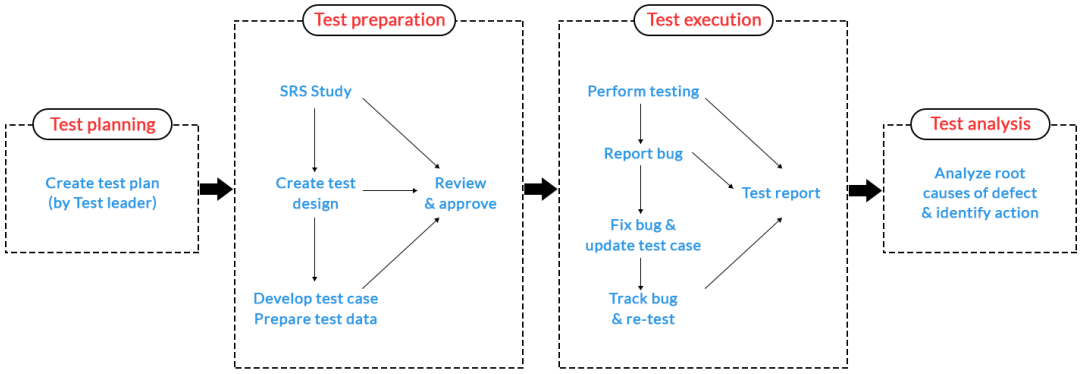


Figure 2. Test process

Description:

1. **Test plan**: test leader takes responsible to create a test plan.
2. **Test preparation**: Testing team study the SRS document, create test design for each stage.
3. **Test execution**: tester and developer perform the test following the plan. The bug is logged by tester and developer is assigned to fix the code in order to fix the bugs.
4. **Test analysis**: test leader analyze root cause of defect & identity action.

5.3.3.6. Test schedule

Test Schedule	Start Date	End Date
Create test plan	15/01/2019	16/01/2019
Learning for Recorder	17/01/2019	22/01/2019
Testing for Recorder	23/01/2019	27/01/2019
Learning for Playback	28/01/2019	30/01/2019
Testing for Playback	30/01/2019	01/02/2019
Learning for Log	11/02/2019	13/02/2019
Testing for Log	13/02/2019	15/02/2019
Learning for Compare UI	18/02/2019	22/02/2019
Testing for Compare UI	25/02/2019	28/02/2019
Learning for Report	04/03/2019	08/03/2019
Testing for Report	11/03/2019	15/03/2019
Testing final	18/03/2019	15/04/2019
Acceptance test	16/04/2019	26/04/2019

Table 82. Test schedule

5.3.4. Features to be tested

Group of function	Function Name	Actor
Project	Create New Project	User
	Open Existing Project	User
	Close Current Project	User
	Save File	User
	Save Current Project	User
	Exit Application	User
	Open File	User
	Rename File	User
	Delete File	User
Record	Record New Test Case	User
	Record Current Test Case	User
	Append Test Case	User
	Run Test Case	User
	Play All Test Cases	User
	Add Manual Mouse Action	User
	Add Manual Keyboard Action	User
	Add Text Action	User
	Add Manual Wait Time	User
	Move Step Up/Down	User
	Edit Step	User
	Delete Step	User
	Open Test Case	User
	Filter Step By Mouse Move	User
Log	Open Log File	User
	Count Displayed Steps	User
	Filter Step By Status	User
Visualizer	Open Test Visualizer	User
	Display Image List (Record)	User
	Display Image Pair (Log, UI Comparison)	User
	Fit Image With Panel	User
	Zoom Image	User
	Highlight Interacted UI Element	User

	Show Result Image	User
Image Viewer	View Single Image	User
	Show Original Image Size	User
	View Image Pair	User
Compare UI	Open UI Comparison Module	User
	Filter For Folder Chooser	User
	Select Expected Image Folder	User
	Input Expected Image Folder Path	User
	Select Actual Image Folder	User
	Input Actual Image Folder Path	User
	Add Manual Compared Images	User
	Clear All Images	User
	Compare All Images	User
	View Image Pair in Image Viewer	User
	Delete Image Pair	User
Report	View “Report”	User
	Export Report To Excel	User
	Export Report To HTML	User
Settings	Setting preferences general	User
	Setting preferences Compare UI	User
Other	Reset Layout	User
	View About System	User
	Launch Apps	User

5.3.5. Features not to be tested

- Functions that are not listed in Use case.
- Functions that are out of scope.

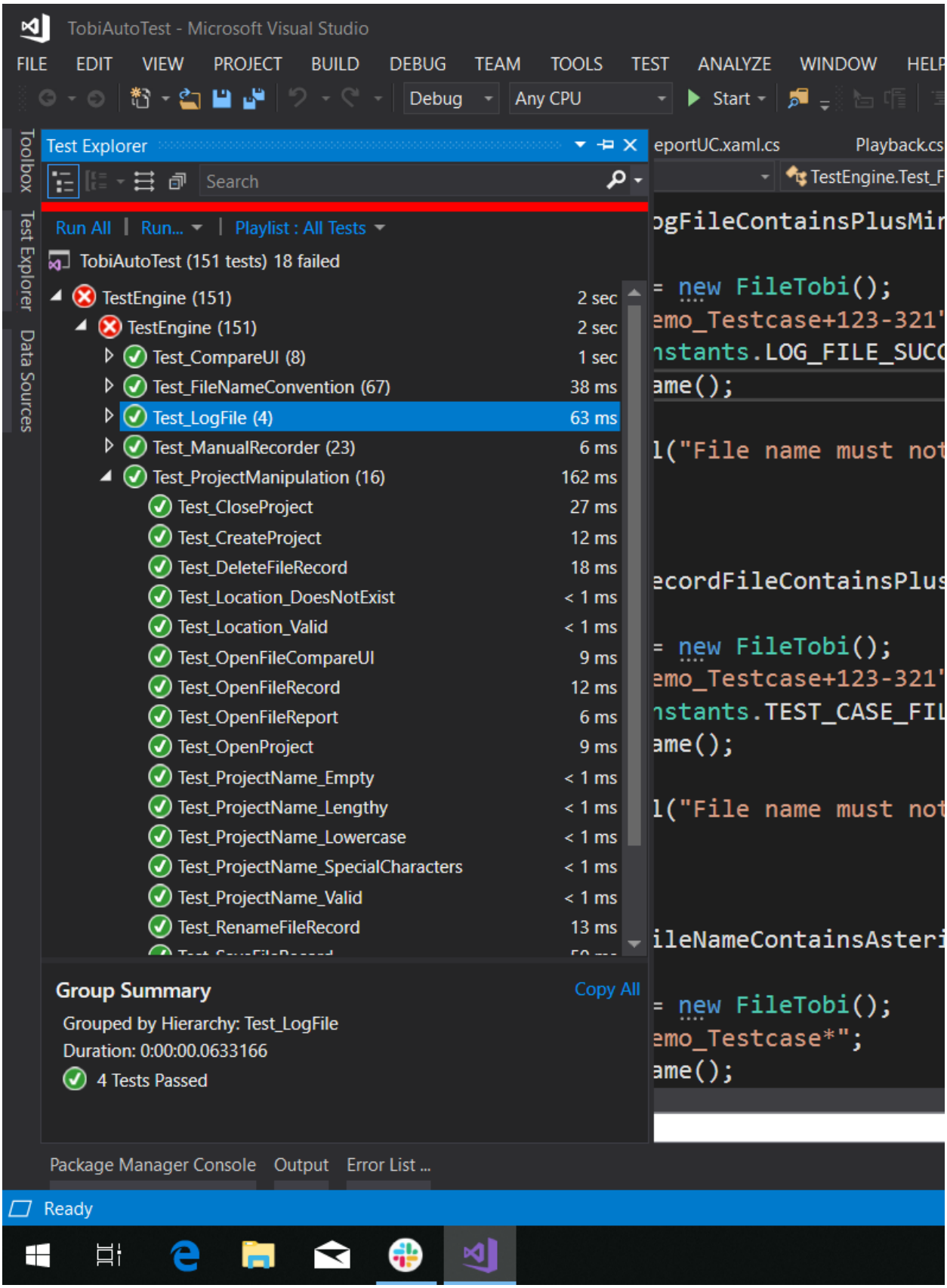
5.4. Test approach

5.4.1. Unit test

Unit testing will be done by developers and approved by team leader.

TOBI develop team embraces this feature to gain the following advantages:

- Reducing the level of bugs in production code.
- Saving development time.
- Automation tests can be run as frequently as required.
- Making it easier to change and refactor code by improving the design of code, especially with Test-driven development.
- It can easily form a document from the tests.
- It is easier to maintain than Front-end tests which are difficult to maintain with the short release cycles and frequent changes and with a complex system.
- Reducing cost of resource to corresponding Front-end testing.
- This is sample tests of Unit Test in TOBI tool:


TobiAutoTest - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Debug
Any CPU
Start
Toolbox
Test Explorer
Data Sources
Search
Run All
Run...
Playlist : All Tests
TobiAutoTest (151 tests) 18 failed
TestEngine (151) 2 sec
TestEngine (151) 2 sec
Test_CompareUI (8) 1 sec
Test_FileNameConvention (67) 38 ms
Test_LogFile (4) 63 ms
Test_ManualRecorder (23) 6 ms
Test_ProjectManipulation (16) 162 ms
Test_CloseProject 27 ms
Test_CreateProject 12 ms
Test_DeleteFileRecord 18 ms
Test_Location_DoesNotExist < 1 ms
Test_Location_Valid < 1 ms
Test_OpenFileCompareUI 9 ms
Test_OpenFileRecord 12 ms
Test_OpenFileReport 6 ms
Test_OpenProject 9 ms
Test_ProjectName_Empty < 1 ms
Test_ProjectName_Lengthy < 1 ms
Test_ProjectName_Lowercase < 1 ms
Test_ProjectName_SpecialCharacters < 1 ms
Test_ProjectName_Valid < 1 ms
Test_RenameFileRecord 13 ms
Group Summary
Copy All
Grouped by Hierarchy: Test_LogFile
Duration: 0:00:00.0633166
4 Tests Passed
eportUC.xaml.cs
Playback.cs
TestEngine.Test_F
ogFileContainsPlusMir
= new FileTobi();
emo_Testcase+123-321'
nstants.LOG_FILE_SUCC
ame();
l("File name must not
ecordFileContainsPlus
= new FileTobi();
emo_Testcase+123-321'
nstants.TEST_CASE_FIL
ame();
l("File name must not
ileNameContainsAsteri
= new FileTobi();
emo_Testcase*";
ame();
Package Manager Console
Output
Error List ...
Ready

5.4.2. Integration and System test

Detail test cases will be described in TOBI_TestCase_Final.xlsx file.

As a standard definition, TOBI defines that a test case is:

- A set of test data and test programs (test scripts) and their expected results. A test case validates one or more system requirements and generates a pass or fail.
- A good test case should follow two basic aspects, the Contents and the Style. Test cases for functional testing are derived from the target of test's use cases Test cases should be developed for each use case scenario. The use case scenarios are identified by describing the paths through the use case that traverse the basic flow and alternate flows start to finish through the use case.

5.4.3. Regression test

We use Regression Testing in following occasions:

- Bugs Fixes.
- New Functionality addition.

On all above situations we need to perform integration testing.

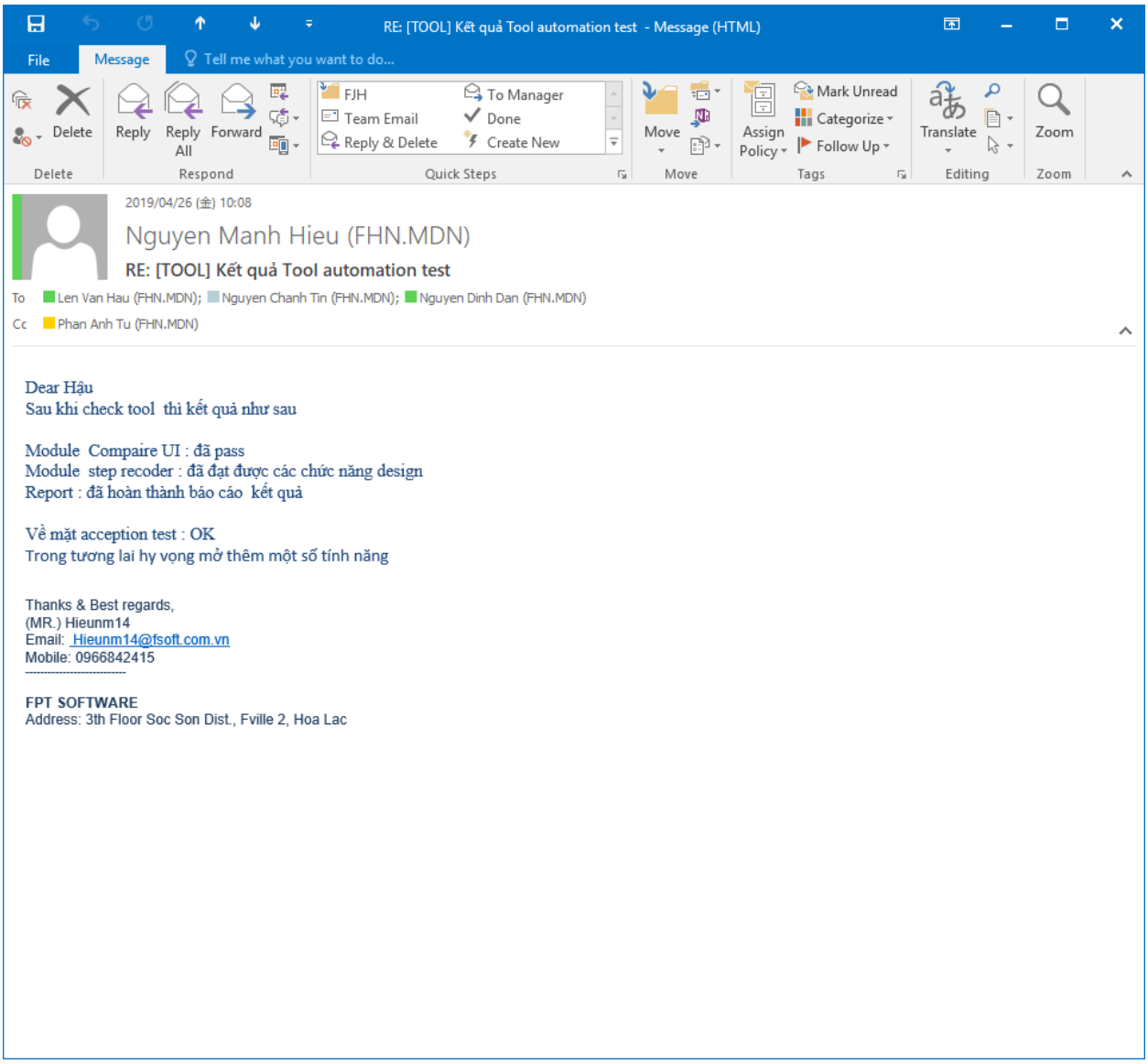
In regression testing we know that we executed all passed test cases but do we execute all the pass test cases? Well there are many criteria's which help to decide the test cases means whether we have to execute all the test cases or we are going to execute the test cases which are related to core functionality or a test cases for a particular modules.

5.4.4. Acceptance test

- These acceptance criteria are following this constraint:

No	Acceptance criteria	Note
1	100% test case executed	
2	92% pass rate test cases	
3	95% of medium severity defects have been closed	
4	100% of high severity defects have been closed	

- Mail & check list confirm result acceptance test of End user:



No.	Checklist	Yes	No	Opinion
	General			
1	Is there any problem when downloading and setting up our product?		x	
2	Is there any problem when using our product?		x	

3	Is there any feature that you do not understand how it works?		x	
4	What is (are) your problem(s) with the flow of pages?	No problem		
5	Which parts or features do you like most?	Compare UI		
6	Which parts or features do you dislike most?	N/A		
GUI				
1	Are texts on all pages checked with grammar and spelling errors?	x		
2	Are checkboxes and buttons displayed clearly in suitable positions?	x		
3	Is design style friendly and easy to use?	x		
4	What is (are) problem(s) with color of the product?	No problem		
5	What is (are) other problem with GUI that user meet?	No problem		
Functions				
1	Did the application set up suitable initial configuration based on your device specification?	x		
2	Is there any function that did not work or worked ineffectively?		x	
3	Is there any function user think that it was useless?		x	
4	Is load time of tool less than 3 seconds?		x	
5	What is (are) the problem(s) with computer RAM when using?	No problem		
6	What is (are) the problem(s) with computer hard disks memory when using?	No problem		

Table 83. Acceptance test form

Defect log

Excel is used to manage bugs of TOBI.

Processes:

- During the testing process, when a bug is found, testers access Defect Log file to log it. The bug must be described as detailed as possible with screenshots. The issue is then assigned to a developer, who is in charge of developing that module.
- Assigned developer checks out the issue in Defect Log file and fixes it. Once the bug is resolved, the assigned developer changes the issue status to “Resolved” and assigns it to a tester, whose responsibility is to check the bug again.
- Tester will then verify if the bug is completely resolved. If the bug is fixed, tester will change the status to “Closed”. If the bug is not fixed, tester will change the status to “Open” and assign it to developer again.

No.	Date Created	Function	Created By	Defect Description	Priority	Assign	Assigned Date	Estimated Time to F	Status	Expected result	Fix date	Actual Time to Fix	Root Cause	Reference Evidence
1	3/17/2019	Create New Project	GiangNT	Function of button Browse is not yet actived	Medium	HauLV	03/17/19	4	Closed	Function of button Browse is actived	03/19/19	4	Funtional was not yet implemented	Bug_01
2	3/17/2019	Create New Project	GiangNT	Create project with name is empty, message "Project name has existed already!" is displayed	Low	HauLV	03/17/19	1	Closed	Create project with name is a space, message "Project name must not be empty" is displayed	03/19/19	0.5	String Validation	Bug_02
3	3/17/2019	Create New Project	GiangNT	Create project with name is a space, message "Project name has existed already!" is displayed	Low	HauLV	03/17/19	1	Closed	Create project with name is a space, message "Project name is not valid" is displayed	03/19/19	0.5	String Validation	Bug_03
4	3/17/2019	Create New Project	GiangNT	Create project with name is a space and a dot, the program is closed	Low	HauLV	03/17/19	1	Closed	Create project with name is a space and a dot, message "Project name is not valid" is displayed	03/19/19	1	String Validation	
5	03/17/19	Create New Project	GiangNT	Create project with name is string of special characters, project is created	Low	HauLV	03/17/19	2	Closed	Create project with name is is string of special characters, message "Project name is not valid" is displayed	03/19/19	2	String Validation	Bug_04
6	03/17/19	Create New Project	GiangNT	Enter string "C:" or "C" into textbox "Location", project is created	Low	HauLV	03/17/19	1	Closed	Enter string "C" into textbox "Location", button "OK" is disabled	03/19/19	1	String Validation	Bug_05
7	3/28/2019	Run Test	GiangNT	Enter "abcdef", hold down the BackSpace for a few seconds to clear all text, but while testcase running, only "f" is deleted.	Medium	DongND	03/28/19	3	Opened	All text is deleted when testcase is running	03/30/19	3	Hook issue	

Figure 3. Defect log example

5.5. Test report

5.5.1. Iteration 1

Module Code	Phase 1 - TOBI - Results				
Test requirement	Run on desktop app				
Tester	GiangNT				
	Passed	Failed	Untested	N/A	Number of Test cases
	29	9	0	0	38

Figure 4. Test result of Window application: Iteration 1

Module Code	Phase 1 - TOBI - GUI			
Test requirement				
Tester	GiangNT			
	Passed	Failed	Untested	Number of Test cases
	7	1		8

Figure 5.Test GUI of Window application: Iteration 1

TEST REPORT

Project Name	*Tobi Automation Testing Tool*	Creator	GiangNT
Project Code	*TOBI*	Reviewer/Approver	HauLV
Document Code	*TOBI_Final_Report.docx*	Issue Date	*1/20/2019*
Notes			

No	Module code	Pass	Fail	Untested	N/A	Number of test cases
1	Phase 1- TOBI - Results	29	9	0	0	38
2	Phase 1- TOBI - GUI	7	1	0	0	8
	Sub total	36	10	0	0	46

Test coverage 100.00 %
Test successful coverage 78.26 %

Figure 68. Test report of iteration 1

5.5.2. Iteration 2

Module Code	Phase 2 - TOBI - Results				
Test requirement	Run on desktop app				
Tester	GiangNT				
	Passed	Failed	Untested	N/A	Number of Test cases
	33	5	0	0	38

Figure 7. Test result of Window application: Iteration 2

Module Code	Phase 2 - TOBI - GUI			
Test requirement				
Tester	GiangNT			
	Passed	Failed	Untested	Number of Test cases
	7	1		8

Figure 80. Test GUI of Window application: Iteration 2

TEST REPORT

Project Name	*Tobi Automation Testing Tool*	Creator	GiangNT
Project Code	*TOBI*	Reviewer/Approver	HauLV
Document Code	*TOBI_Final_Report.docx*	Issue Date	*2/28/2019*
Notes			

No	Module code	Pass	Fail	Untested	N/A	Number of test cases
1	Phase 1- TOBI - Results	29	9	0	0	38
2	Phase 1- TOBI - GUI	7	1	0	0	8
3	Phase 2- TOBI - Results	33	5	0	0	38
4	Phase 2- TOBI - GUI	7	1	0	0	8
	Sub total	76	16	0	0	92

Test coverage	100.00 %
Test successful coverage	82.61 %

Figure 91. Test report of iteration 2

5.5.3. Iteration 3

Module Code	Phase 3 - TOBI - Results				
Test requirement	Run on desktop app				
Tester	GiangNT				
	Passed	Failed	Untested	N/A	Number of Test cases
	30	8	0	0	38

Figure 192. Test result of Window application: Iteration 3

Module Code	Phase 3 - TOBI - GUI			
Test requirement				
Tester	GiangNT			
	Passed	Failed	Untested	Number of Test cases
	8	0		8

Figure 193.Test GUI of Window application: Iteration 3

TEST REPORT

Project Name	*Tobi Automation Testing Tool*	Creator	GiangNT
Project Code	*TOBI*	Reviewer/Approver	HauLV
Document Code	*TOBI_Final_Report.docx*	Issue Date	*2/28/2019*
Notes			

No	Module code	Pass	Fail	Untested	N/A	Number of test cases
1	Phase 1- TOBI - Results	29	9	0	0	38
2	Phase 1- TOBI - GUI	7	1	0	0	8
3	Phase 2- TOBI - Results	33	5	0	0	38
4	Phase 2- TOBI - GUI	7	1	0	0	8
5	Phase 3- TOBI - Results	30	8	0	0	38
6	Phase 3- TOBI - GUI	8	0	0	0	8
	Sub total	114	24	0	0	138

Test coverage	100.00 %
Test successful coverage	82.61 %

Figure 194. Test report of iteration 3

5.5.4. Iteration 4

Module Code	Phase 4 - TOBI - Results				
Test requirement	Run on desktop app				
Tester	GiangNT				
	Passed	Failed	Untested	N/A	Number of Test cases
	33	5	0	0	38

Figure 195. Test result of Window application: Iteration 4

Module Code	Phase 4 - TOBI - GUI			
Test requirement				
Tester	GiangNT			
	Passed	Failed	Untested	Number of Test cases
	8	0		8

Figure 196.Test GUI of Window application: Iteration 4

TEST REPORT

Project Name	*Tobi Automation Testing Tool*	Creator	GiangNT
Project Code	*TOBI*	Reviewer/Approver	HauLV
Document Code	*TOBI_Final_Report.docx*	Issue Date	*3/24/2019*
Notes			

No	Module code	Pass	Fail	Untested	N/A	Number of test cases
1	Phase 1- TOBI - Results	29	9	0	0	38
2	Phase 1- TOBI - GUI	7	1	0	0	8
3	Phase 2- TOBI - Results	33	5	0	0	38
4	Phase 2- TOBI - GUI	7	1	0	0	8
5	Phase 3- TOBI - Results	30	8	0	0	38
6	Phase 3- TOBI - GUI	8	0	0	0	8
7	Phase 4- TOBI - Results	33	5	0	0	38
8	Phase 4- TOBI - GUI	8	0	0	0	8
	Sub total	155	29	0	0	184

Test coverage	100.00 %
Test successful coverage	84.24 %

Figure 197. Test report of iteration 4

5.5.5. Iteration 5

Module Code	Phase 5 - TOBI - Results				
Test requirement	Run on desktop app				
Tester	GiangNT				
	Passed	Failed	Untested	N/A	Number of Test cases
	163	0	0	0	163

Figure 198. Test result of Window application: Iteration 5

Module Code	Phase 5 - TOBI - GUI			
Test requirement				
Tester	GiangNT			
	Passed	Failed	Untested	Number of Test cases
	8	0		8

Figure 199.Test GUI of Window application: Iteration 5

TEST REPORT

Project Name	*Tobi Automation Testing Tool*	Creator	GiangNT
Project Code	*TOBI*	Reviewer/Approve	AnhBN
Document Cod	*TOBI_Final_Report.docx*	Issue Date	*4/15/2019*
Notes			

No	Module code	Pass	Fail	Untested	N/A	Number of test cases
1	Phase 1 - TOBI - Results	29	0	0	0	29
2	Phase 1 - TOBI - GUI	7	0	0	0	7
1	Phase 2 - TOBI - Results	33	0	0	0	33
2	Phase 2 - TOBI - GUI	7	0	0	0	7
1	Phase 3 - TOBI - Results	30	0	0	0	30
2	Phase 3 - TOBI - GUI	8	0	0	0	8
1	Phase 4 - TOBI - Results	33	0	0	0	33
2	Phase 4 - TOBI - GUI	8	0	0	0	8
1	Phase 5 - TOBI - Results	163	0	0	0	163
2	Phase 5 - TOBI - GUI	8	0	0	0	8
	Sub total	326	0	0	0	326

Test coverage	100 %
Test successful coverage	100 %

Figure 200. Test report of iteration 5

CHAPTER 6 – SOFTWARE USER MANUAL

6.1. User Guidelines

6.2.1. Installment

Environment setting up: .NET framework 4.5.2

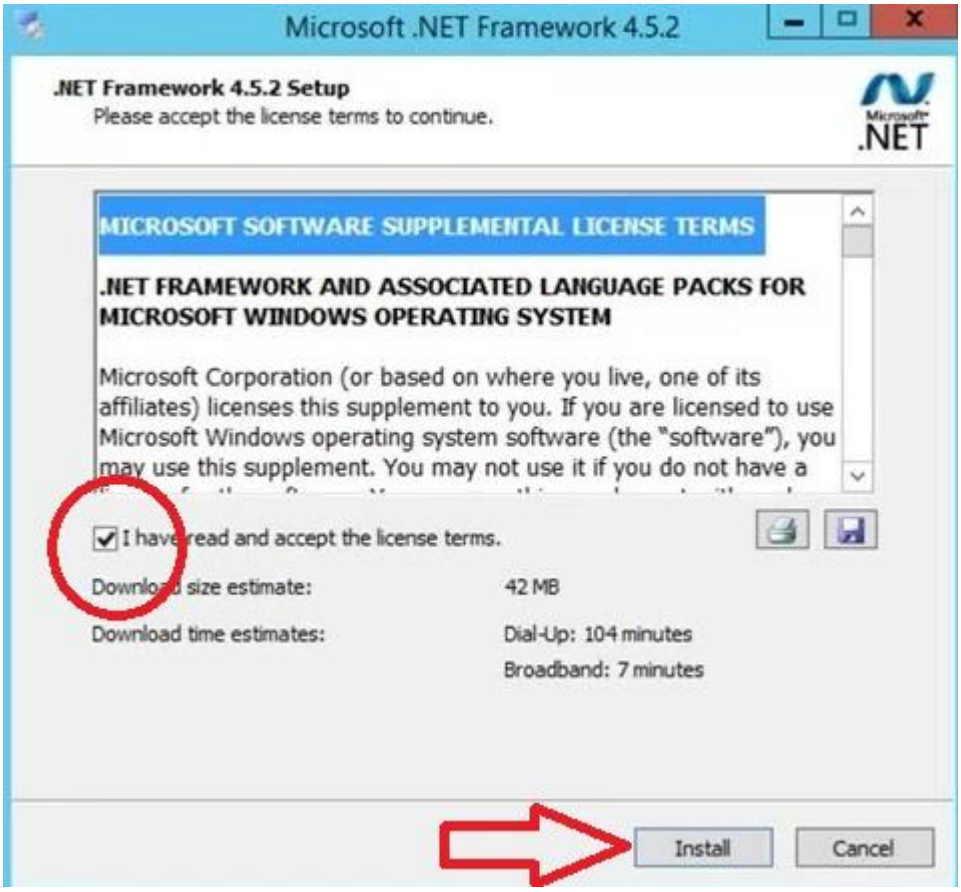


Figure 135. .NET 4.5.2 set up: Step 1

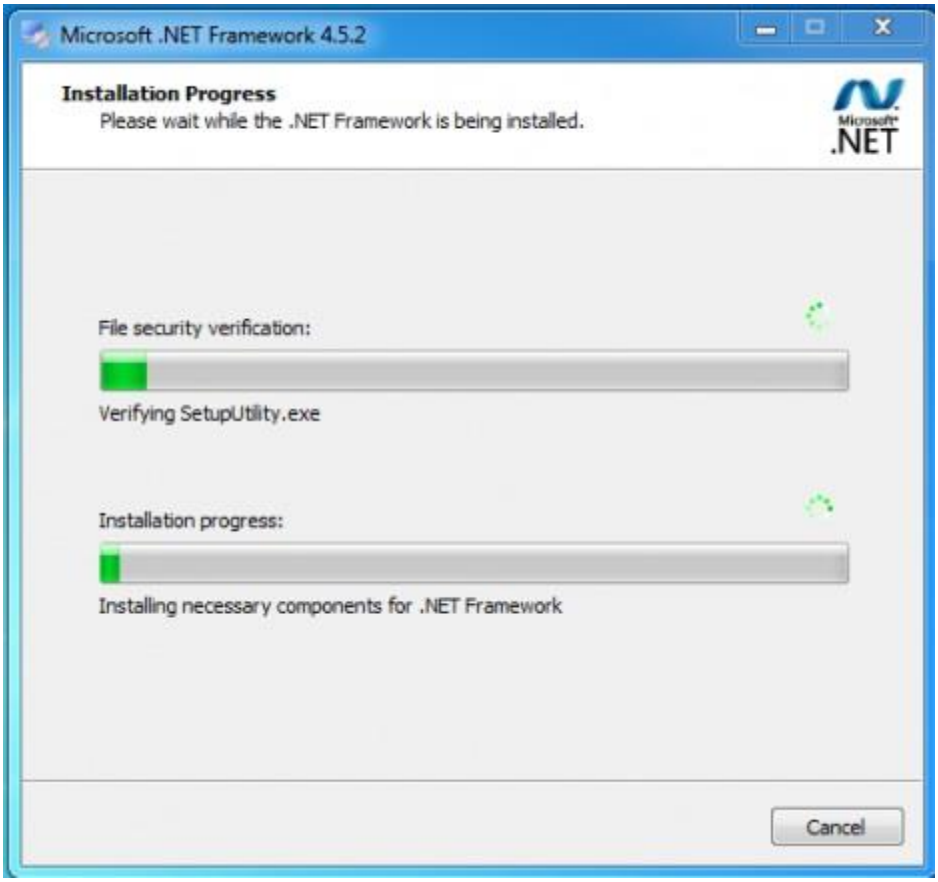


Figure 136. .NET 4.5.2 set up: Step 2

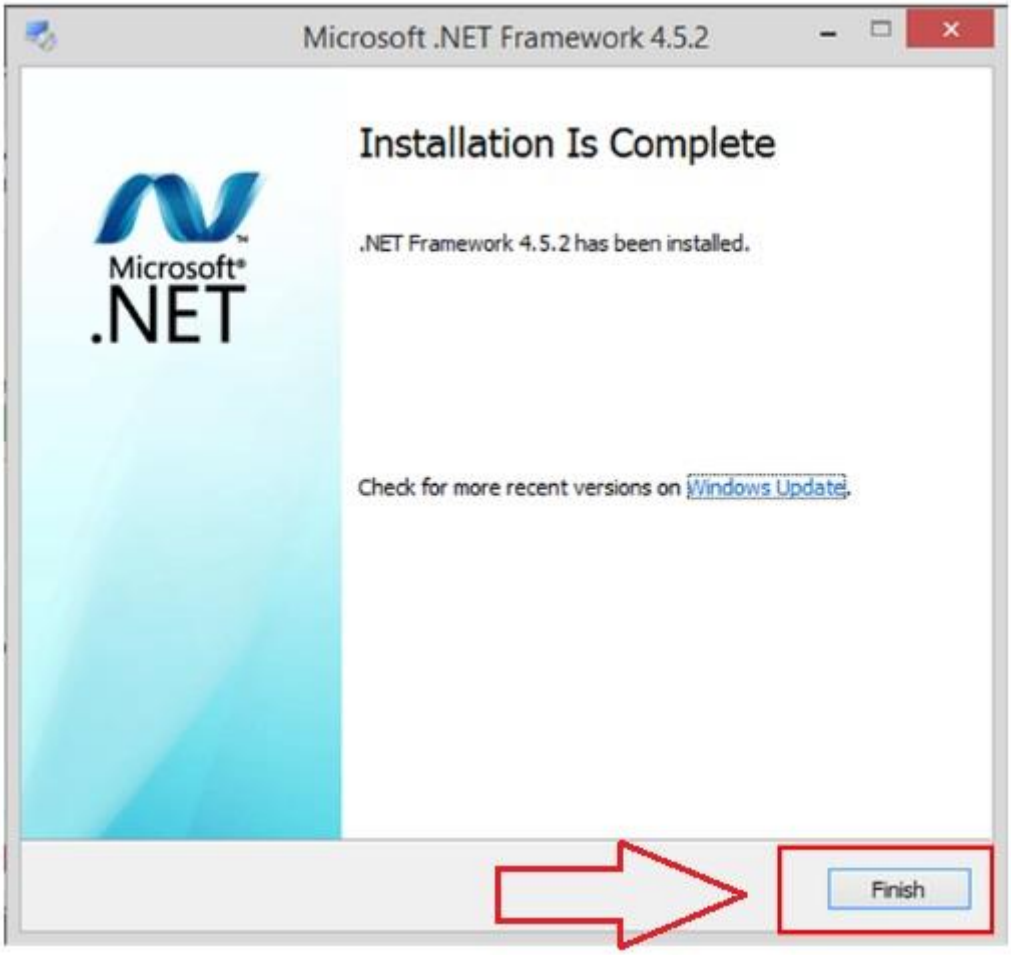


Figure 137. .NET 4.5.2 set up: Step 3

Environment setting up: Microsoft Visual C++ Redistributable for Visual Studio 2017

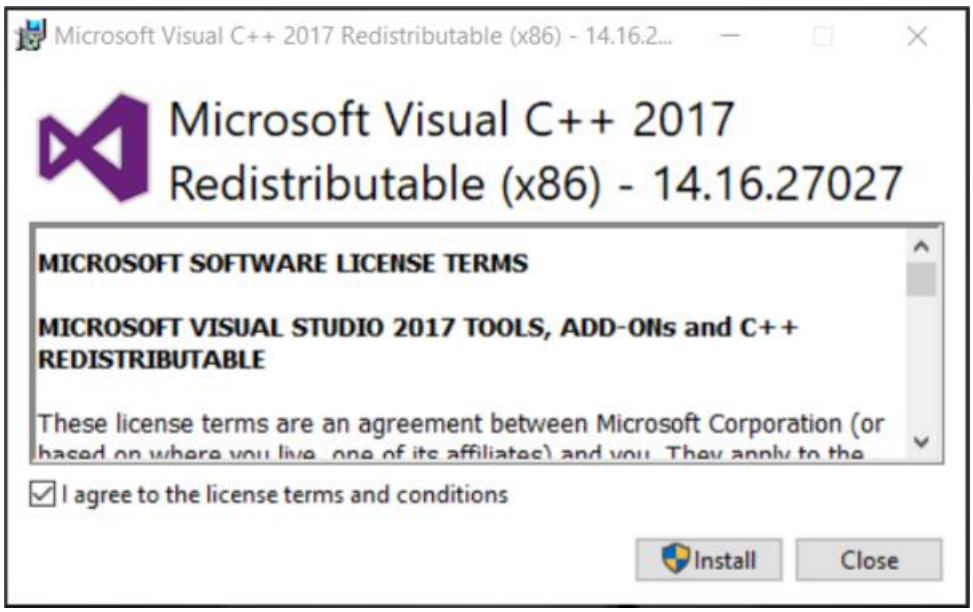


Figure 138. C++ Redistributable setup

TOBI Test Installation

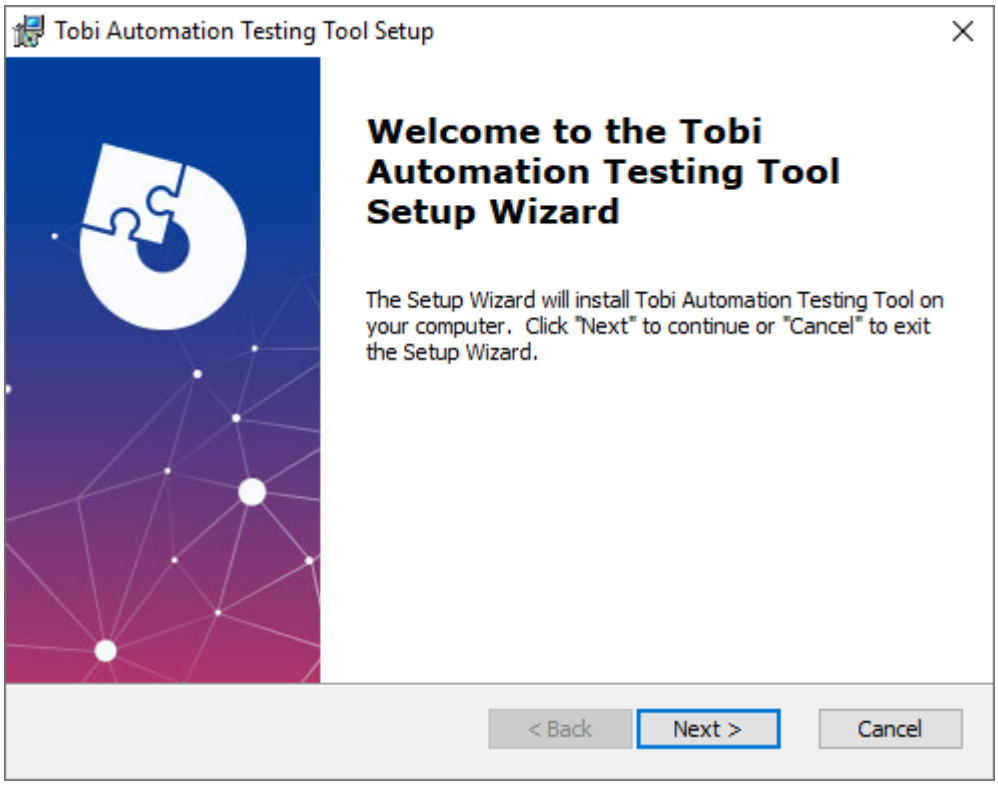


Figure 139. Window application installment: Step 1

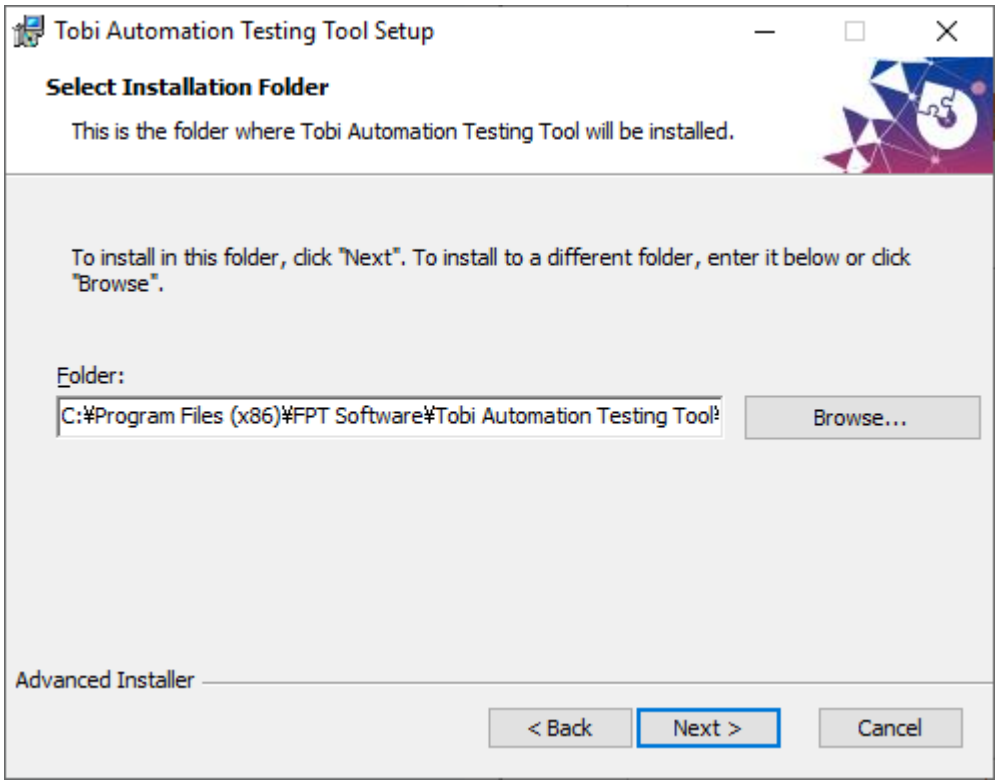


Figure 140. Window application installment: Step 2

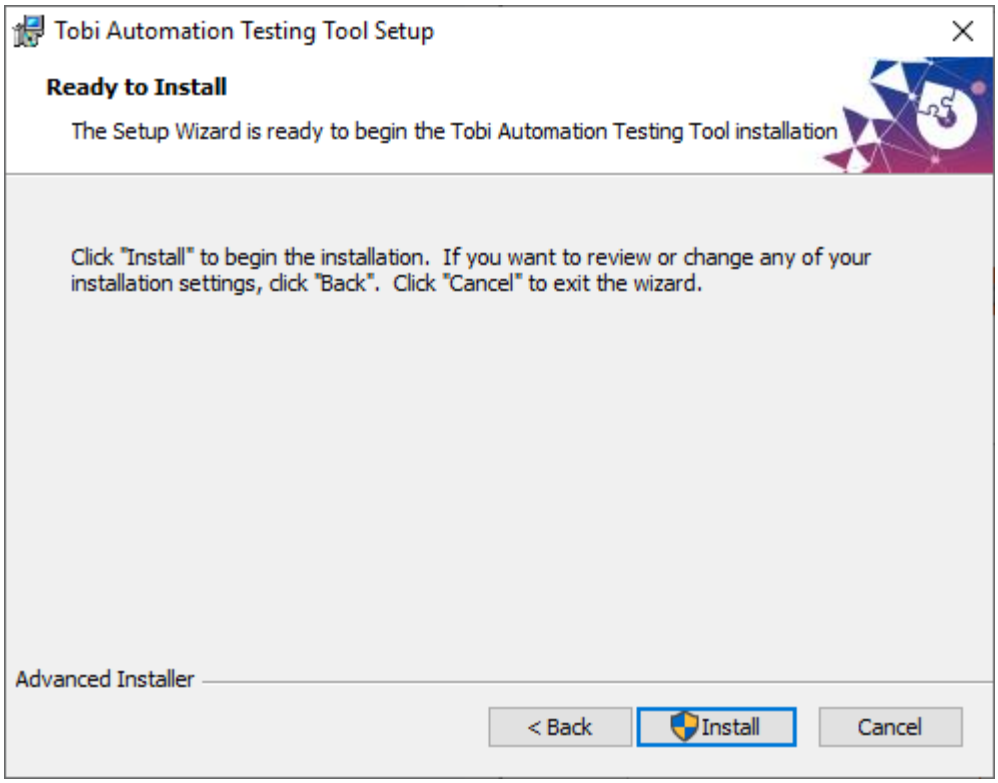


Figure 141. Window application installment: Step 3

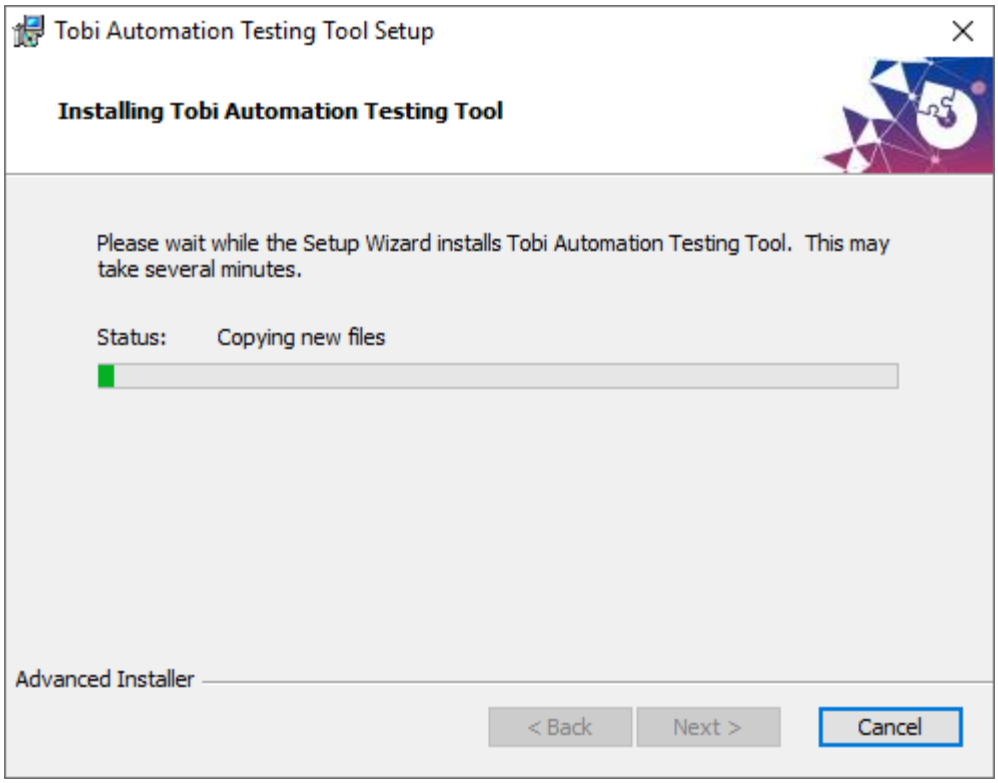


Figure 142. Window application installment: Step 4

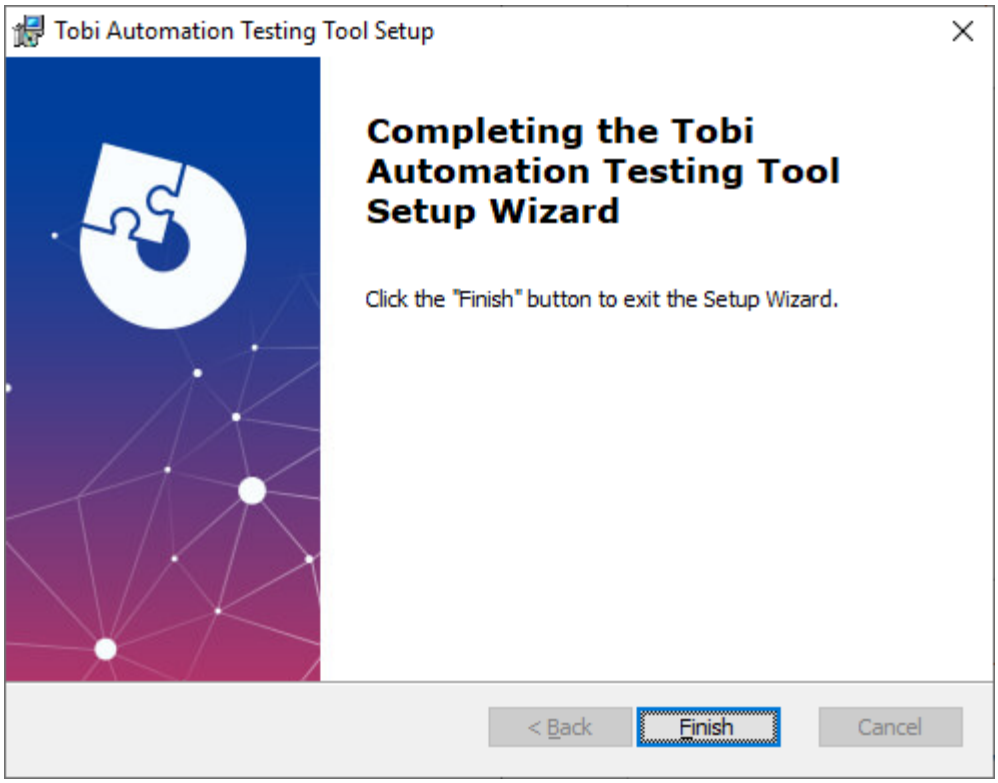


Figure 143. Window application installment: Step 5

6.2.2. Window Application Functions

6.2.2.1 Create Test Project

1. Open TOBI application

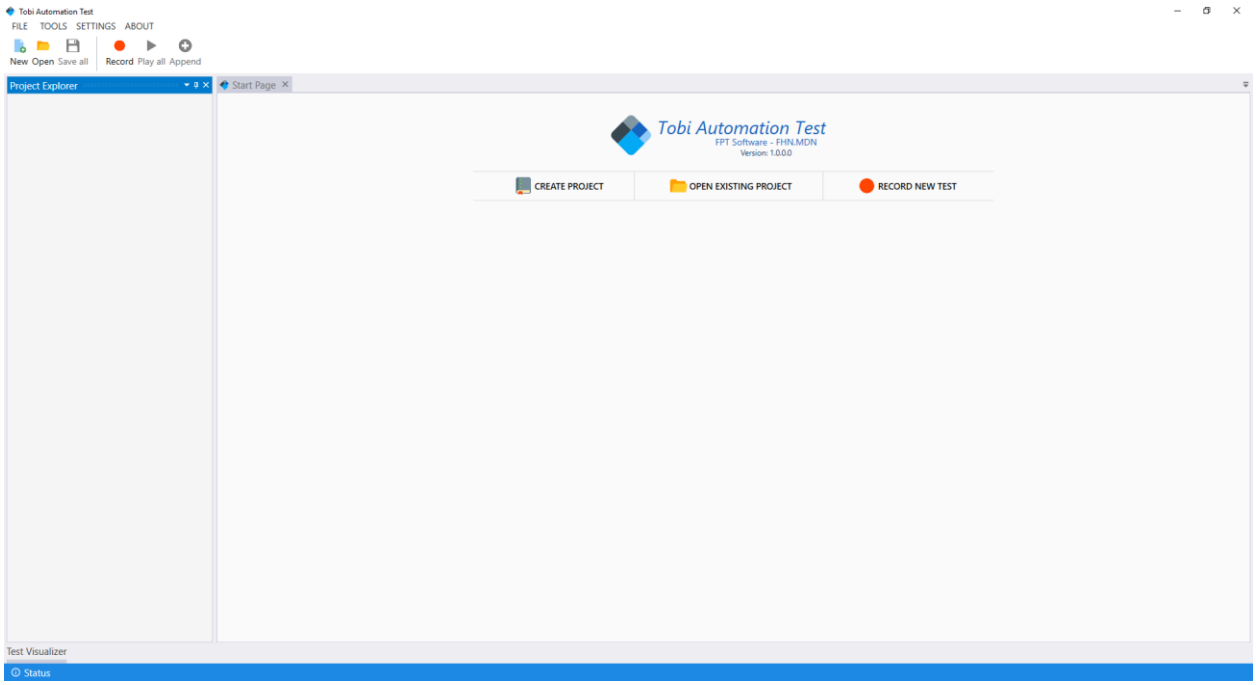


Figure 144: Main Window

2. Click on “New” button in toolbar to open “Create project” dialog
3. Fill information and click “OK” button

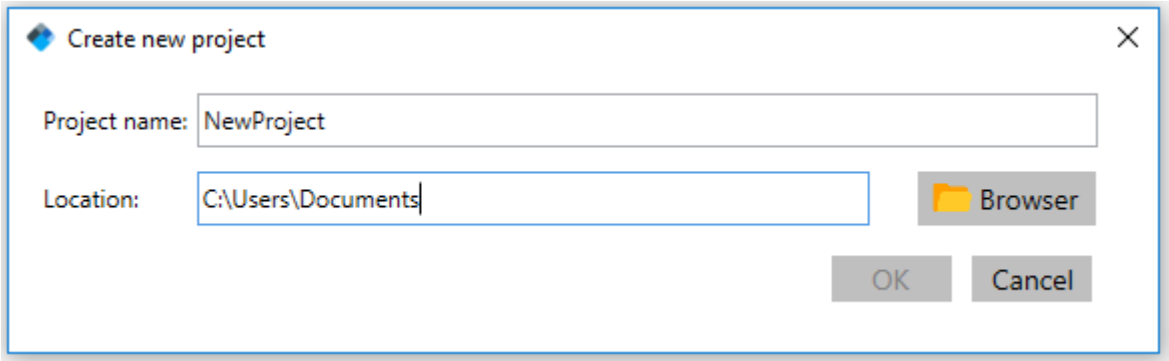


Figure 145: "Create project" dialog

6.2.2.2 Create Test Case

1. Click on “Record” button in toolbar to open “Record test case” dialog
2. Fill test case name
3. Click “OK” button

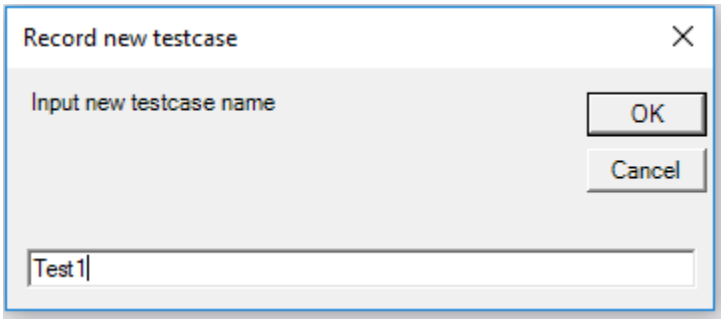


Figure 146: Create test case dialog

6.2.2.3 Start Recording

1. Click “Record” button in test case toolbar
2. System will display “Record” popup below the right corner of the screen

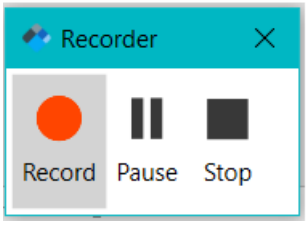
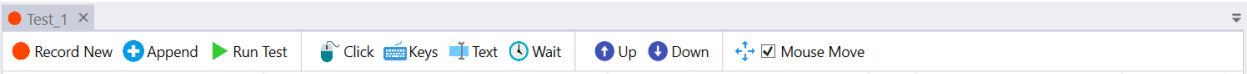


Figure 147: Recorder popup

3. User start record and click “Stop” when finish

6.2.2.4 Run Test Case

1. Click “Run Test” button in test case toolbar



2. System will minimize and run test case

3. After finished running, system will display test log window

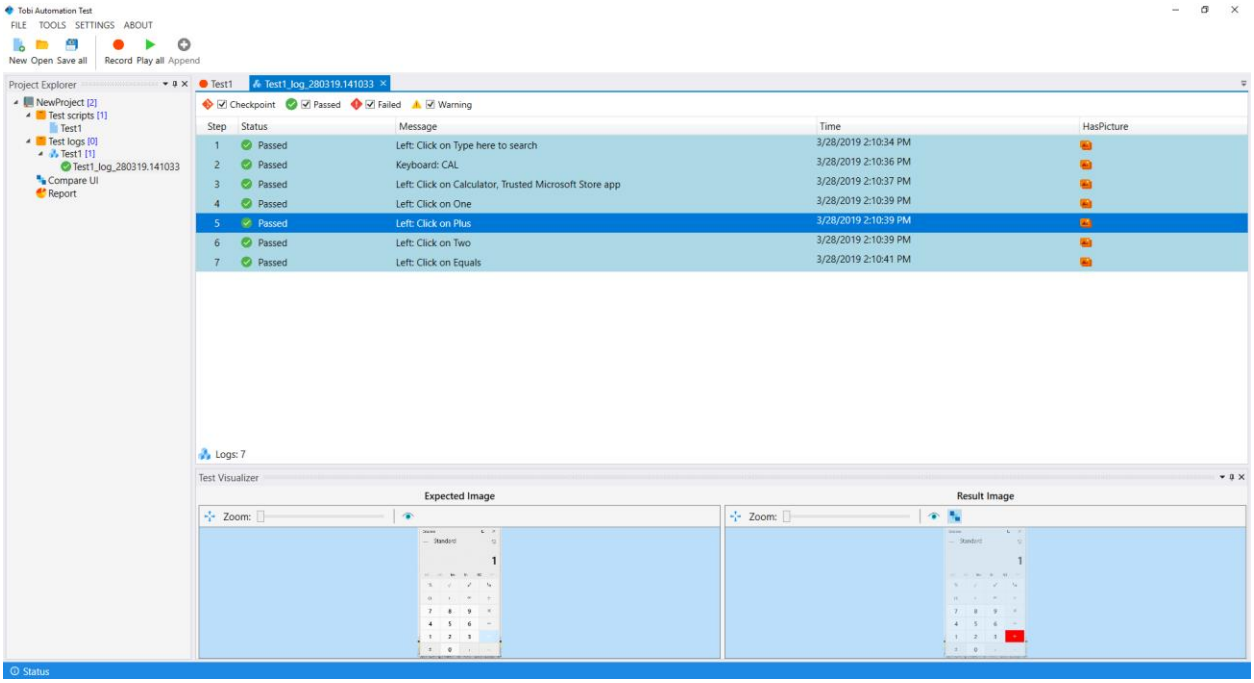


Figure 148: Test Log window

6.2.2.5 Compare Image

1. Click on “Compare UI” button in “Project Explorer”
2. Choose folder of image to compare
3. Click “Compare” button to start compare

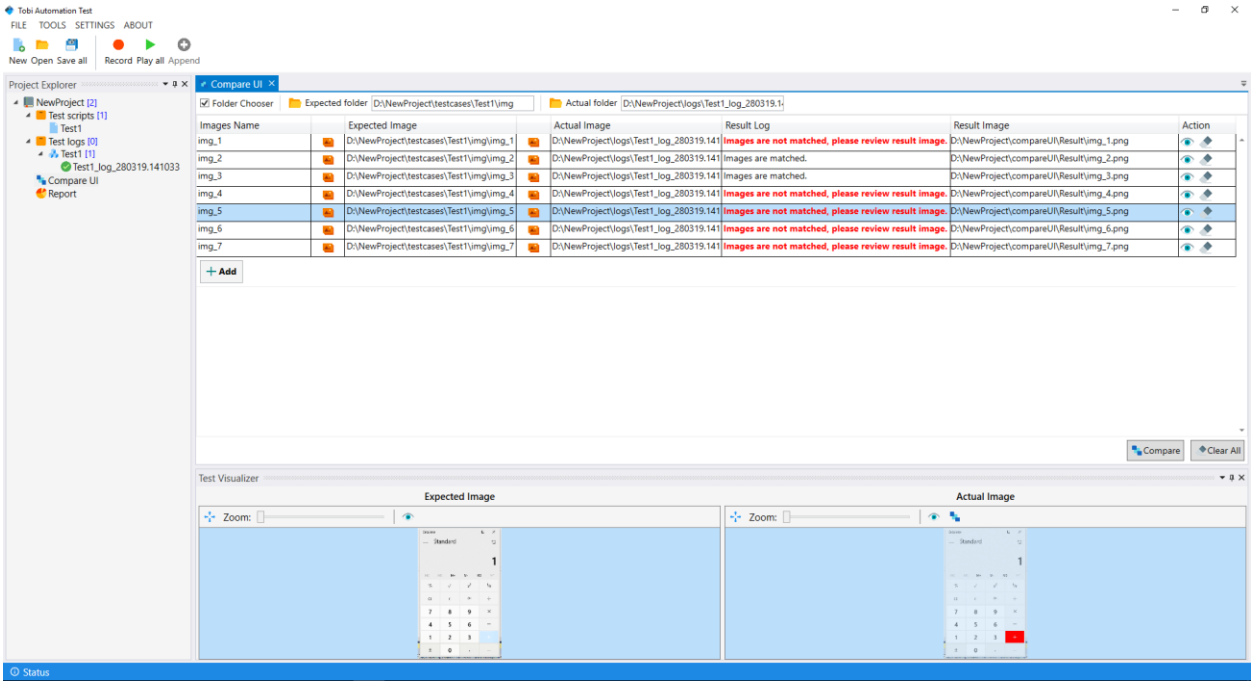


Figure 149: Compare UI window

6.2.2.6 Report

1. Click “Report” button in “Project Explorer”
2. System display “Report” window

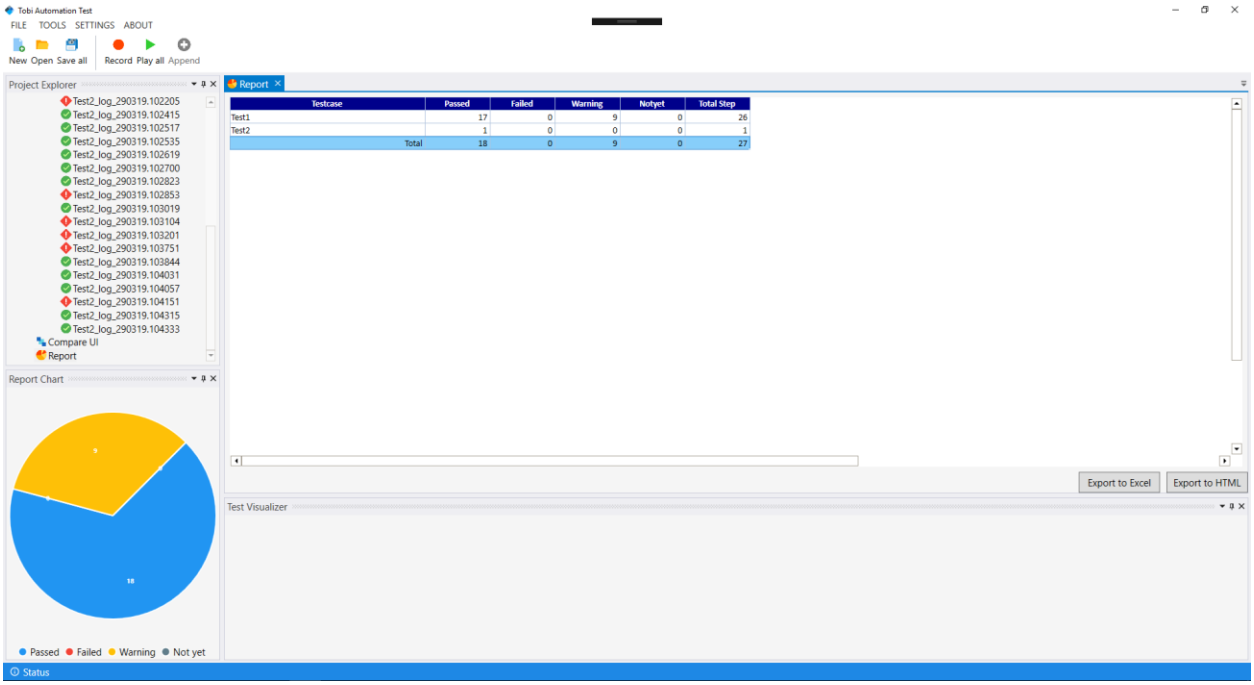


Figure 150: Report window

REFERENCES

1. Windows Automation API

a. ***UI Automation, n.d. Windows Automation API. [Online]***

Available at: https://docs.microsoft.com/en-us/windows/desktop/winauto/windows-automation-api-portal#purpose

2. WPF - Window Presentation Framework

a. ***Can, K., n.d. Introduction to WPF in Visual Studio. [Online]***

Available at: https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/introduction-to-wpf-in-vs

b. ***Microsoft, n.d. The Complete WPF Tutorial. [Online]***

Available at: https://wpf-tutorial.com/

c. ***Shivprasad koirala, n.d. WPF MVVM step by step (Basics to Advance Level). [Online]***

Available at: https://www.codeproject.com/Articles/819294/WPF-MVVM-step-by-step-Basics-to-Advance-Level

d. ***Wikipedia, n.d. Model-View-ViewModel. [Online]***

Available at: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel

e. ***Material Design, n.d. Material Design in XAML Toolkit. [Online]***

Available at: http://materialdesigninxaml.net/

f. ***Tran, D., 2017. Bitmap.LockBits Method. [Online]***

Available at: https://docs.microsoft.com/en-us/dotnet/api/system.drawing.bitmap.lockbits?view=netframework-4.8

3. MVVM Pattern

a. ***Jeremy Likness, n.d. Model-View-ViewModel (MVVM) Explained. [Online]***

Available at: https://csharperimage.jeremylikness.com/2010/04/model-view-viewmodel-mvvm-explained.html

4. Theme - FluentWPF

a. Source code, n.d. FluentWPF. [Online]

Available at: https://github.com/sourcechord/FluentWPF