HELWAN UNIVERSITY

Faculty of Engineering

Faculty of Engineering
Helwan University
Computer Department
Graduation Project Documentation

BUG TRACKING SYSTEM

Prepared by:
Mohamed Ashraf Kamal Eldin
Mohamed Magdy Sayed
Mohammed Hussein Elsayed

Under the Supervision of:
Prof. Manal Shoman

ACKNOWLEDGEMENT

First, we would like to express our deepest gratitude and appreciation to our advisor **Prof. Manal Shoman** for her support, patience, guidance and encouragement throughout our graduation project.

We would also like to thank our families, especially our parents, for their encouragement, patience, and assistance over the years.

We are forever indebted to our parents, who have always kept us in their prayers and pushed us to Success.

Finally, for our faculty for providing the suitable environment that led us to represent the best image that computer engineering graduates of Helwan University are supposed to represent.

ABSTRACT

Bug Tracking System has been introduced in order to keep track of the reported bug in the system. For many years, bug-tracking system is used only in some of the large software development houses. Most of the others never bothered with bug tracking at all, and instead simply relied on shared lists and email to monitor the status of bugs.

This procedure usually causes errors and lead to those bugs to be dropped or ignored by developers. So, we propose this document to discuss a detailed and complete specification with analysis of our Bug Tracking System which are used only internally in software house.

TABLE OF CONTENTS

CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1. CHAPTER 1: INTRODUCTION

The term “Bug” in context to software development project refers to any defect, fault, flaw that has been encountered in the working of a specific software application or in any product. The mechanism which is used in order to handle these bugs refers to as “Bug Tracking System”.

Bugs have existed as a problem in software development industry and they are normally inevitable in software development. Most software bugs arise from mistakes and errors made by people in a program's source code or its technical design. It was relatively difficult to manage bugs in simple word documents or remember everything in one’s head.

This system provides a platform for tester to report bug tickets and follow the latest progress for the tickets in a project. It combines and organizes information in single system and centralized data repository for changes and bugs in system. Without a proper system project stakeholder cannot access this information easily. It makes communication between teams more effective. It keeps all information that related to a ticket in a single system. It supports the developers and the testing team to collaborate with each other through the bug tracking system. They can educate each other on the nature of the bug and on the potential solution.

This system is focused on simple concepts and usability to provide an easy-to-use solution for small and medium businesses. The system provides user friendly interface and easy to use functions for tracking tickets. As a result, it simplifies every step of tracking bug, thus it can save time and work can be done efficiently.

CHAPTER 2: LITERATURE REVIEW

2. CHAPTER 2: LITERATURE REVIEW

This chapter provides the detail survey about various
developments in the field of bug tracking system. The survey is organized according to the year wise systematic enhancement in this field.

A. Gunes Koro and Jelf Tian [13] presented their survey report by using a methodology which included conducting a survey in order to predict defect handling process. A. Gunes Koro and Jelf Tian meet their objective by using the tool Bugzilla, survey tool. In their research they found out different defect handling process adopted by various open-source projects.

John Anvik, Lyndon Hiew Gail C.Murphy [11] have presented their research work by examining the two-open bug receptacle from eclipse and Firefox projects. They met their aim by using tool Bugzilla. In their research they come out with various challenges that the developers face due to duplicate bugs and bug triage problem.

Sascha Just, Rahul Premraj, Thomas Zimmermann [7] presented their research work by using a methodology using the card sorting technique that involved the developer and user of Ecllipse, Apache, Mozilla. They met their aim using the card sorting technique. After the completion of their research, they presented few suggestions on how one can improve the working of current bug tracking system by recognizing the bug duplicates etc.

Nicolas Bettenburg, Rahul Premraj, Thomas Zimmermann, Sunghun Kim [9] presented an experimental report on by proposing a methodology by conducting an experiment that took into account 16,511 master report and 27,838 reports submitted by the reporters. They met their aim by using the tool called infozilla. In their research they proved that duplicated bug report cannot always be considered as harmful, they in fact provide the developer with some useful information about the bug that were not even present in the master report [8]. Their research work throws light on various applications of infozilla in extracting structural information.

Jorge Aranda, Gina Vendia [6] presented an experimental report in which they adopted a methodology that included a survey conducted between the software professionals and studied various research around bug fixing. In their research they presented various ways by which one can get the complete set of information regarding bug‟s history and has also explained the use of automated analysis of bug record data in acquiring information about the bug.

Thomas Zimmermann et al. [1] presented an experimental report by proposing a methodology in which they conducted an experiment and considered the views of various researchers. Authors met their objective by using tool like cuezilla, macro recorder Whyline tool.

In their research they presented the list of items that must be acquired from the reporters in order to complete the bug report.

Swati Sen and Anita Ganpati [15] presented the report in which they used a methodology which was truly based in collecting data from various survey, research paper etc. Authors met their aim by using the tools like Bugzilla, Bug Genie, I tracker, Web Issues, Mantis. In their research they provided the pros and cons of various Bug Tracking tool by comparing each one of them in various characteristics.

In this work, we reviewed 8 papers on the topic of defect reporting published since 2004 through to 2013.We offer the following conclusions on the current state of the literature.

There is a disconnect between what is generally provided by the reporters in a defect report and what developers find useful. Organizations taking the time to closely evaluate what they are capturing in defect reports and trying to assist reporters in improving the quality of their report are paying off in a measurable way in terms of improved software quality.

Filtering defect reports results in a significant resource overhead. Automatic methods of assignment, categorization, and duplicate detection are being used with measured success. The advantages in this area are significant, with defect reports going to the developers that can resolve the issue the quickest, actual defect reports being separated from technical assistance and enhancement requests, and duplicates being aggregated into a master defect report for a singular issue.

CHAPTER 3: Analysis and Requirements

3. CHAPTER 3: ANALYSIS AND REQUIREMENTS

3.1. ANALYSIS

In order to arrive at the specific requirements, we wrote below, we did some steps to collect ideas and data about the proposed project, which are:

- We did a research about some systems serving the same goal.
- We did a small survey with couple of groups consisting of both developers and testers through social media.
- We did a lot discussions with our supervisor professor about the topic.

3.2. SYSTEM REQUIREMENTS

3.2.1. Functional Requirements

1. **Register as a user / Login:**

   Description: Secure system access.
2. **Assign/unassign users to/from roles:**

   Description: Role-based security.
3. **Create projects:**

   Description: Organization of resources
4. **Assign/unassign users to/from projects:**

   Description: Organization of resources
5. **Create requirement**

   Description: Organization of resources

6. **Assign/unassign users to/from requirements**

   Description: To help in identifying the responsible developer of each requirement

7. **Create test case**

   Description: To make the system more inclusive and attach every bug ticket to it for more insight

8. **Edit test case**

   Description: To store the result of the execution of the test case or edit another information

9. **List test cases by requirements assigned to tester**

   Description: Provide easy tracking of test cases to a particular tester

10. **sort test case list by title, status**

    Description: Ease of use and access

11. **Create tickets**

    Description: When a test case fails the system offer an option to create a bug ticket

12. **Assign tickets to developer**

    Description: A developer must be responsible for a ticket item, enforce accountability.

13. **Edit submitted tickets**

    Description: Make modifications to existing tickets

14. **Create ticket comments**

    Description: Add progress comments and other important information to the ticket history.

15. **Create ticket attachments**

Description: Add helpful visuals and other documentation to the ticket history.

16. **List comments, attachments per ticket**

Description: Organization of these resources.

17. **List history of a ticket's changes**

Description: Very important view for the developer on the project.

18. **List tickets based on the role of the user**

Description: Filter the ticket list based on the role to provide easy tracking

19. **Sort ticket list by title, owner, assignment, creation or recent update date/time, priority, status, and project**

Description: Ease of use and access.

20. **Full text search of all relevant fields**

Description: Ease of use and access.

21. **Notification function**

Description: Notify the user about new updates and changes to quickly respond to it.

3.2.2. Non-Functional Requirements

22. **Security**

Description: No one can access the system from outside the software house.

Priority: High.

23. **Usability**

Description: The system will have a user-friendly interface experience and easy to navigate.

Priority: High.

24.Performance

Description: Web pages of the system should load quickly to not break the user experience. Sorting and filtering of data should happen fast enough for user view.

Priority: Medium.

3.2.3. USER INTERFACE REQUIREMENTS

1. Login / Register:
   a. All users are directed to this page when accessing the site.
   b. From this page users can login or register as a new user.

2. Dashboard
   a. Users are directed here on successful login
   b. Display relevant information based on role
   c. Optional: Generate useful statistical information for the project manager

3. Role Management

   Admin users must have access to an interface for assigning users to and removing users from the following roles (roles should be seeded in database along with global admin user):

   • Administrator
   • Project Manager
   • Developer
   • Lead Developer
   • Tester
   • Lead Tester

4. Test Case Management

a. Paged listing of test case items (default 10 testcases per page, may be changed by user)
b. Test cases may be created and edited, not deleted

5. Ticket Management

a. Paged listing of ticket items (default 10 tickets per page, may be changed by user)
b. Users have access to ticket lists based on role:

- Testers see tickets which they own
- Developers see tickets to which they are assigned
- Project managers see tickets belonging to the projects for which they are responsible

c. Tickets may be created and edited, not deleted. Completed tickets receive a "Closed" status.
d. A ticket details page must be used to provide full detailed ticket history for each ticket
e. Provide interfaces for adding comments and attachments
f. Developers should be notified via an inbuilt notification function when assigned a new ticket

6. Requirement Management:

a. Project managers must have access to an interface for creating requirements
b. Project managers must have access to an interface for assigning dev and test leads
c. Dev and Test leads must have access to an interface for distributing requirements on his team.

7. Project Management

a. Project managers must have access to an interface for creating projects

b. Project managers must have access to an interface for assigning users to and removing users from a project.

8. User Profile
   a. Change password
   b. Edit profile information (name, email, etc.)

3.3. FUNCTIONAL REQUIREMENTS SPECIFICATION

3.3.1. System Stakeholders

- Administrator
- Project manager
- Tester
- Developer
- Tester lead
- Developer lead

3.3.2. Actors

a. Administrator

   - Assigning roles to the users of system.

   Objective:

   - Ensure role-based security

b. Project manager (Participating)

   - Create projects
   - Assign employees to a project
   - Monitoring all tickets in a project
   - Assign tickets to various developers

c. Tester (Participating)

- Submitting tickets
- Editing submitted tickets
- Monitoring tickets updates
- Create Test Case
- Edit Test Case

d. Developer (Participating)

- Monitoring tickets which are assigned to him and change the status of them.

e. Tester lead (Participating)

- Assign requirements to the testers which are assigned to him.
- Can do all Tester functions.

f. Developer lead (Participating)

- Assign requirements to the developers which are assigned to him.
- Can do all developer functions.

3.3.3. Use Cases

3.3.3.1. Login to the system

Each user can Login to the system by entering the email and password which make the system Provides Authentication and Authorization.

Requirements:

- Login

3.3.3.2. Role Assignment

Administrator gives Role for each user in the system based on his job.

Requirements:

- Assign/unassign users to/from roles

3.3.3.3. Create project

Project manager can create a new project and put groups of information on it like name, description and the state of the project.

Requirements:

- Create projects

3.3.3.4. Select the participating Employee to each project

Project manager can Select group of Testers and Developers to the project to work on it.

Requirements:

- Assign/unassign users to/from projects

3.3.3.5. Create Requirements to each project

Project manager creates list of Requirements that's serve the Client need.

Requirements:

- Create requirement

3.3.3.6. Select the Dev and test lead for each Requirement

Project manager can select the tester, developer leads to each requirement.

Requirements:

- Assign/unassign users to/from requirements.

3.3.3.7. Create test case

Tester can create group of testcases to each requirement to check the requirement if work as expected or not.

Requirements:

- Create test case

3.3.3.8. Create Bug ticket

When test case fail tester will write bug ticket and send it to developer to help to repair this bug.

Requirements:

- Create tickets

3.3.3.9. Assign Bug ticket

Tester can assign bug ticket to the responsible developer to repair the bug.

Requirements:

- Assign tickets to developer

3.3.3.10. Show the Assigned Bug

Developer can show all assigned bugs to him to start solving it.

Requirements:

- List Tickets by role

3.3.3.11. View/Manage test cases

Tester can view list of test cases based on requirements and can sort test cases based on Some attributes like the title, status, requirement, project in order to easy to use and access.

Requirements:

- List test cases by requirements assigned to tester
- Sort test case list by title, status
- Full text search of all relevant fields
- Edit Submitted Test Cases

3.3.3.12. View/Manage bug ticket

Tester can view list of bug ticket based on role, history and comment and can sort bug ticket based on Some attributes like the title, state, assigned, creating date, priority, project in order to easy to use and access. Tester can write comment in each ticket if he wants to put extra information and can edit the field of bug ticket in any time.

Requirements:

- List comments per ticket
- List attachments per ticket
- List history of a ticket’s changes
- List tickets based on the role of the user
- Sort ticket list by title , assignment, creation or recent update date/time, priority, status, and project
- Edit submitted tickets

- Create ticket comments
- Create ticket attachments
- Full text search of all relevant fields

3.3.3.13. Notification Function

Notify the user from the system about new updates and changes to quickly respond to it.

Requirements:

- Notification function

3.3.3.14. Distribute Requirement to each tester

Tester lead can distribute the requirements to his team

Requirements:

- Assign/unassign users to/from requirements

3.3.3.15. Distribute Requirement to each developer

Developer lead can distribute the requirement to his team

Requirements:

- Assign/unassign users to/from requirements

CHAPTER 4: Software Design

4. CHAPTER 4: SOFTWARE DESIGN

4.1. USE CASE DIAGRAM:

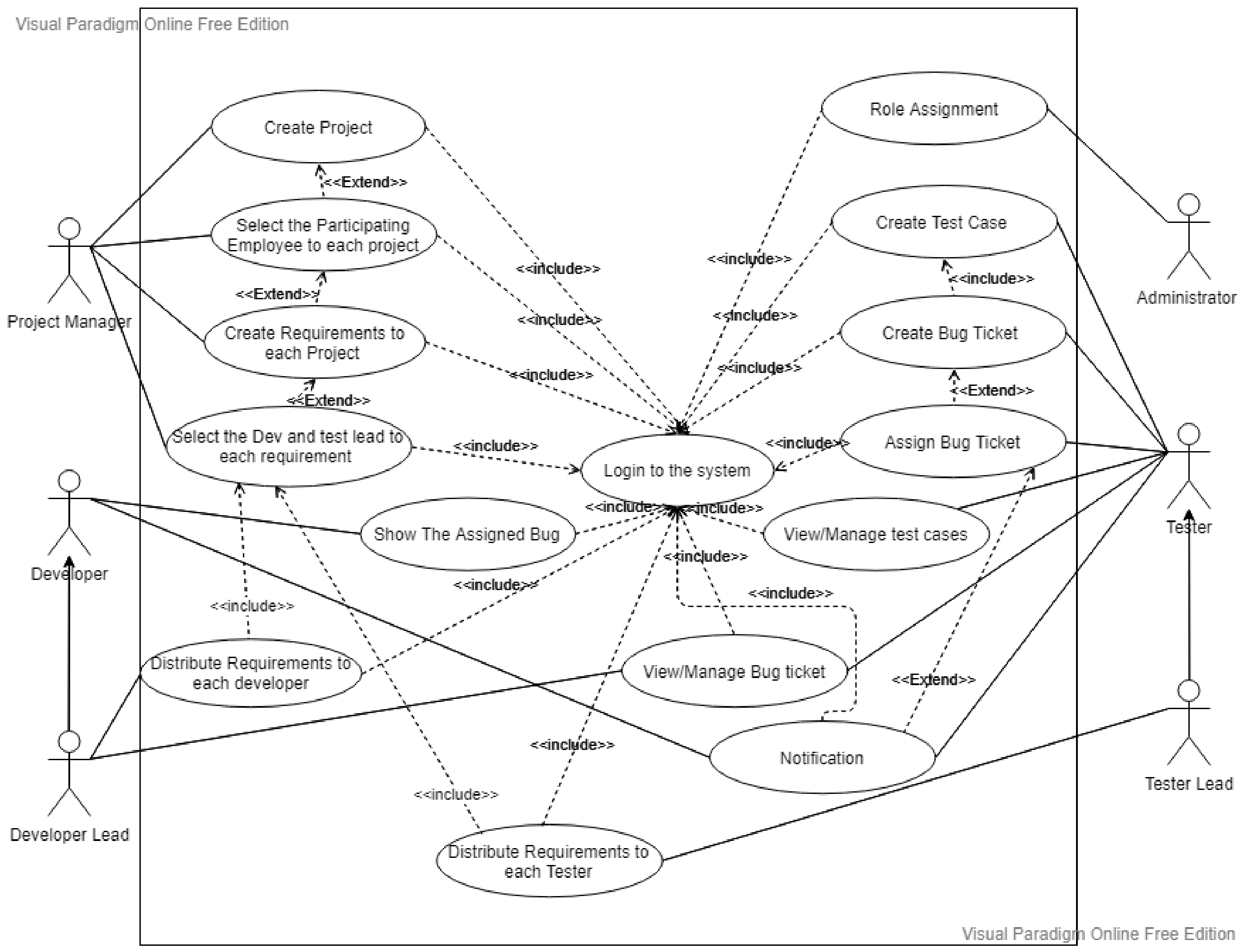


Figure 1:Use Case Diagram

4.2. System Sequences Diagrams

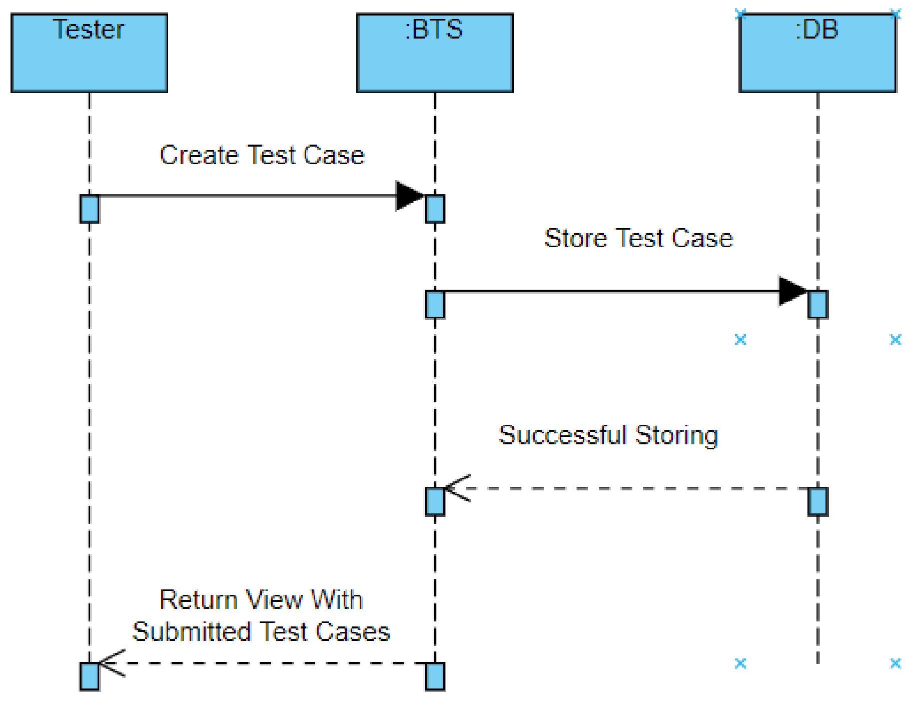


Figure 2:Sequence Diagram for Create test case

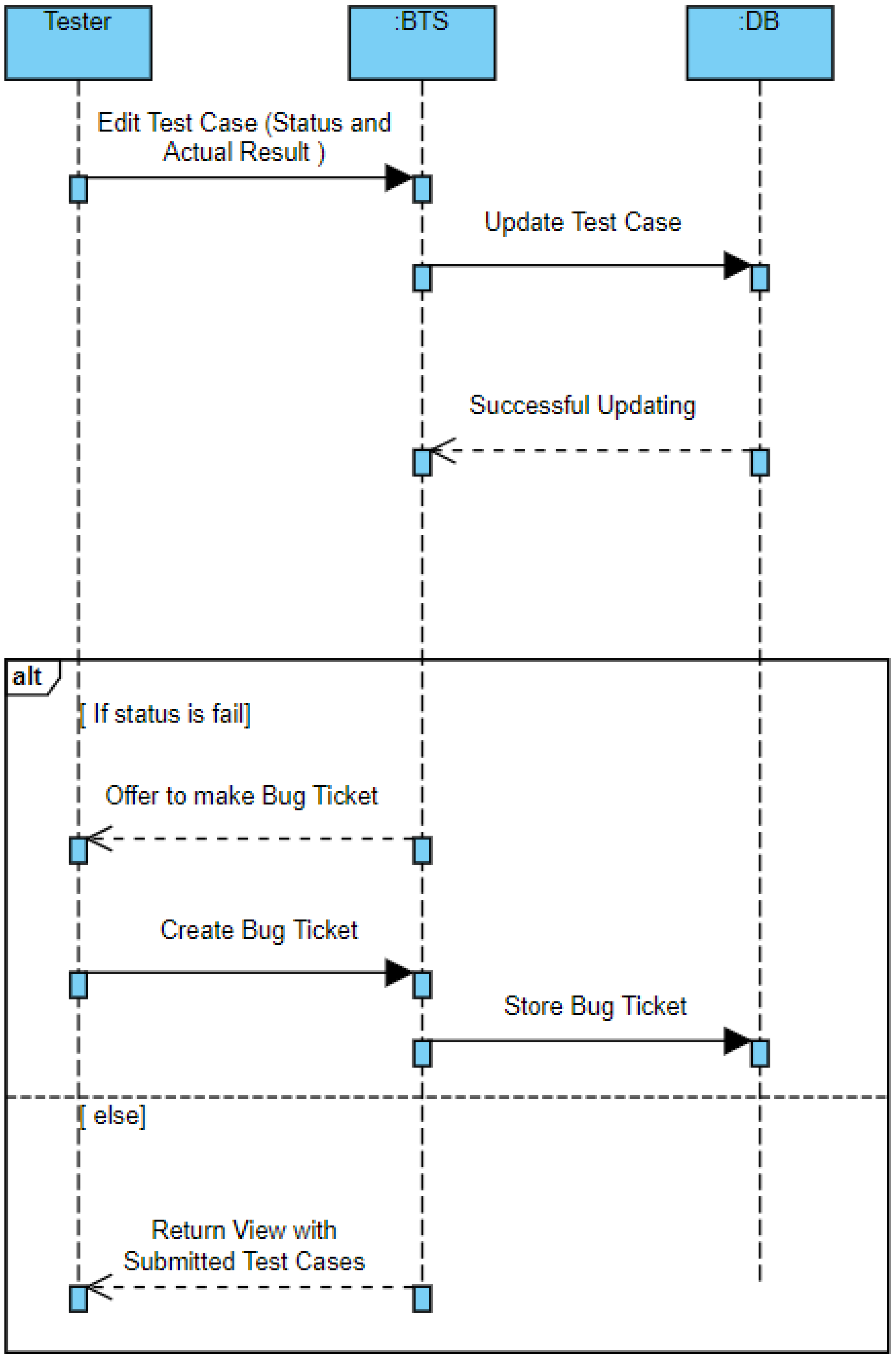


Figure 3: Sequence Diagram for creating bug ticket

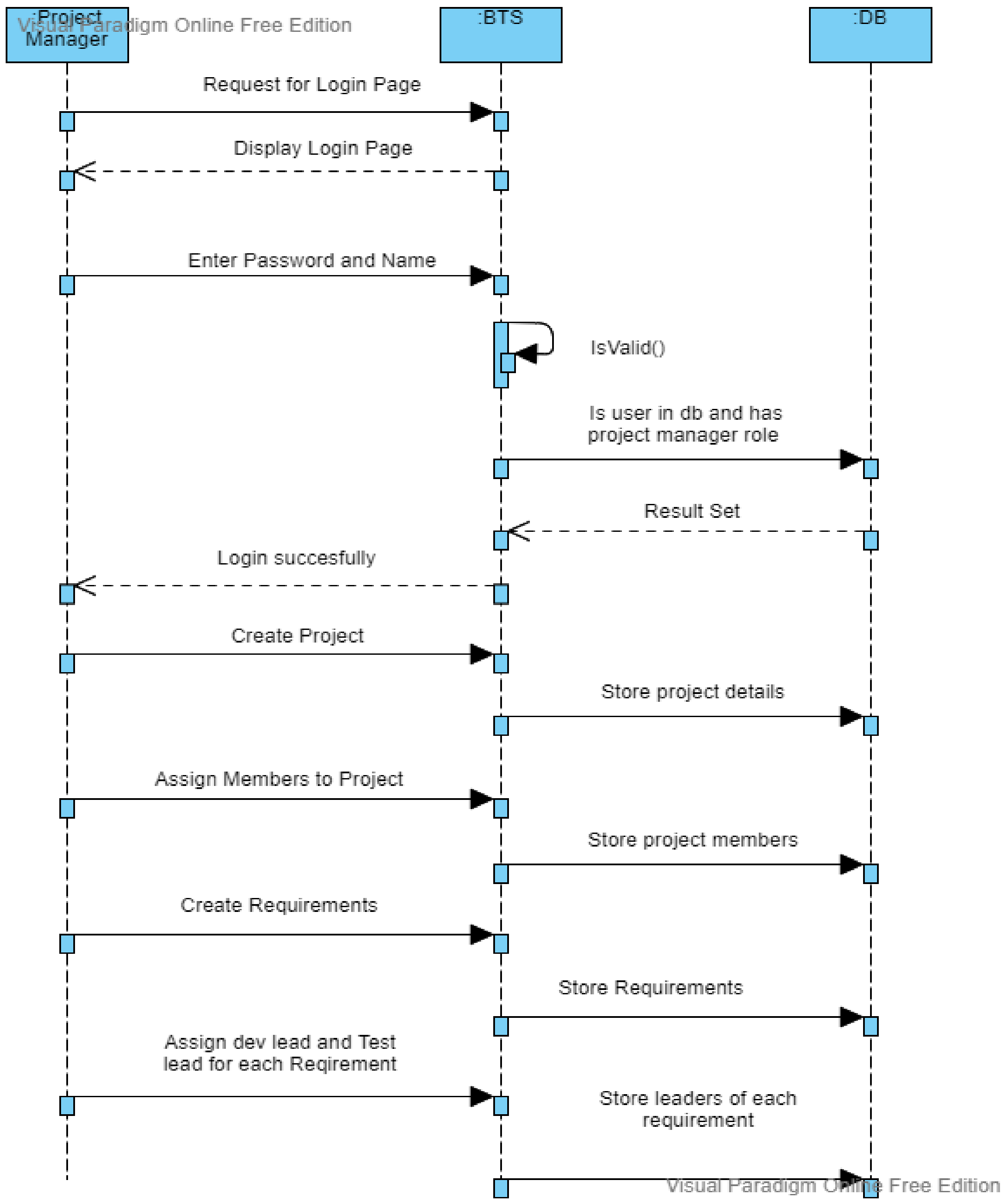


Figure 4:Sequence Diagram for Assign leads to Requirements

4.3. Class Diagram

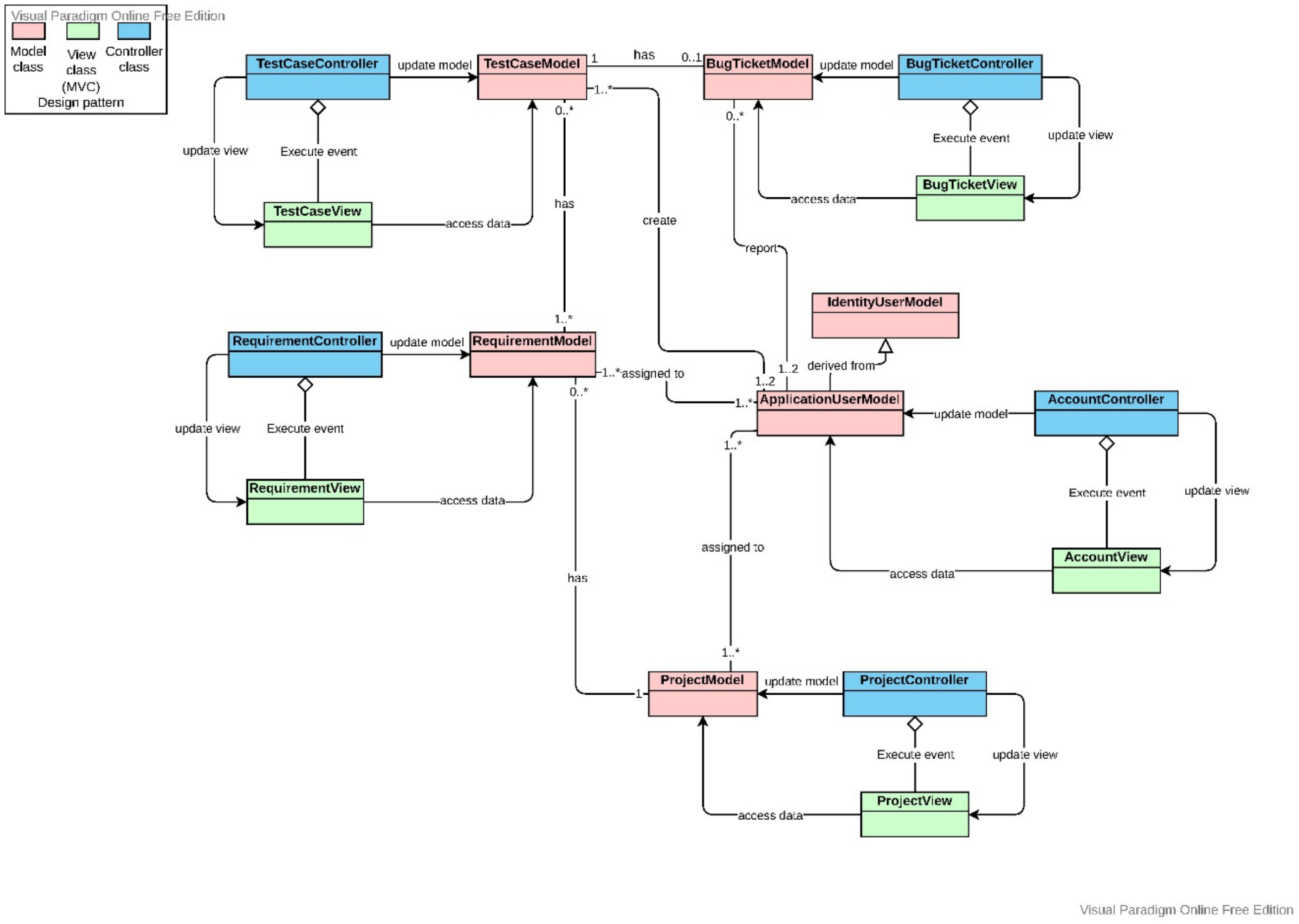


Figure 5:Class Diagram

In our system we have used the MVC design pattern (Model View Controller) to have a better separation of concerns and good design for the architecture of the source code.

The way MVC works is as follows: We have a model class which contain all attributes of the entity which it represents with no methods or implementation. Next, we have the view class and the controller class and they work with each other, the controller is the class responsible for handling the user inputs and update the model class with these input values while the view class is responsible for viewing the data on the interface by accessing these data values in the model class.

The way the controller and view classes work are by the user executing an event in the view (e.g., pressing a button) and then it sends a request to the action result method in the controller which responsible for doing the system function for that event like for example, getting something from the database and display it or receiving input from the user and inserting it in the database or just do some system function code logic. And most of the time the return of these action result methods is view class and any required data if needed to update the view after the execution of the event with the same view and new data, but sometimes the action result method return value is a redirection to another action result method -in the same controller or in a different one- which therefore will return a new view and update the interface.

4.4. RELATIONAL SCHEMA

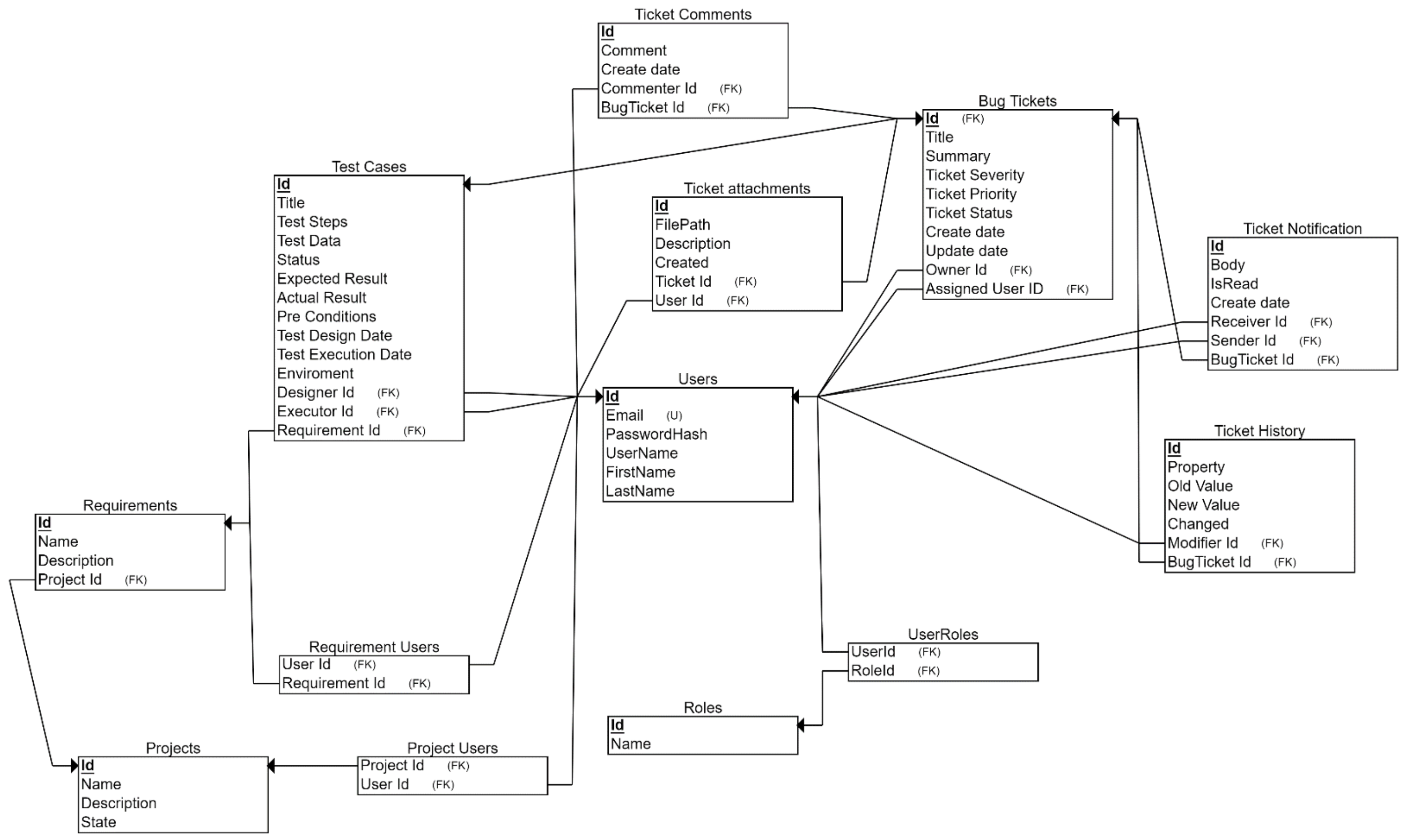


Figure 6: Relational Schema

4.5. WORK FLOWCHART

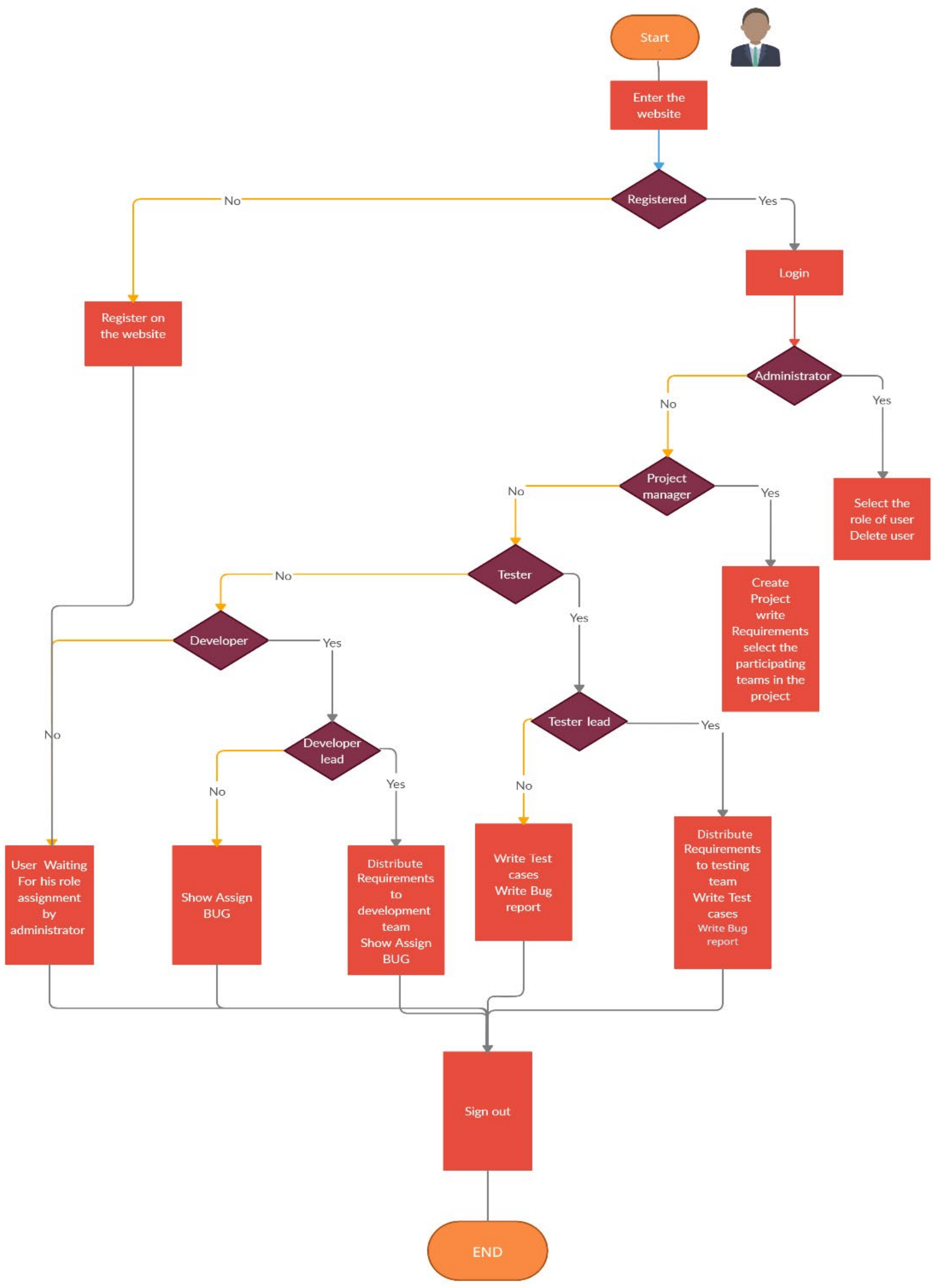

Start
Enter the website
Registered
No
Yes
Register on the website
Login
Administrator
No
Yes
Select the role of user Delete user
Project manager
No
Yes
Create Project write Requirements select the participating teams in the project
Tester
No
Yes
Developer
Yes
No
Tester lead
Yes
Developer lead
No
Yes
No
User Waiting For his role assignment by administrator
Show Assign BUG
Distribute Requirements to development team Show Assign BUG
Write Test cases Write Bug report
Distribute Requirements to testing team Write Test cases Write Bug report
Sign out
END

Figure 7:Workflow Chart

4.6. State Diagram

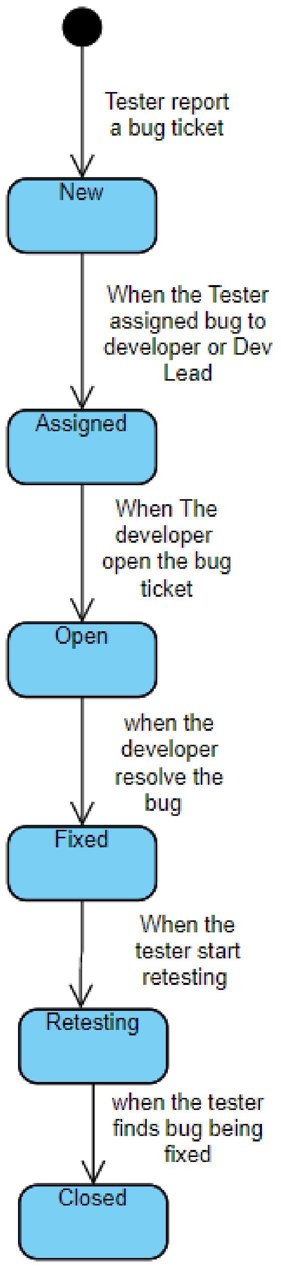

Tester report a bug ticket
New
When the Tester assigned bug to developer or Dev Lead
Assigned
When The developer open the bug ticket
Open
when the developer resolve the bug
Fixed
When the tester start retesting
Retesting
when the tester finds bug being fixed
Closed

Figure 8:State Diagram for Bug Life Cycle

CHAPTER 5: Implementation

5. CHAPTER 5: IMPLEMENTATION

5.1. Software Architecture:

5.1.1. Login:

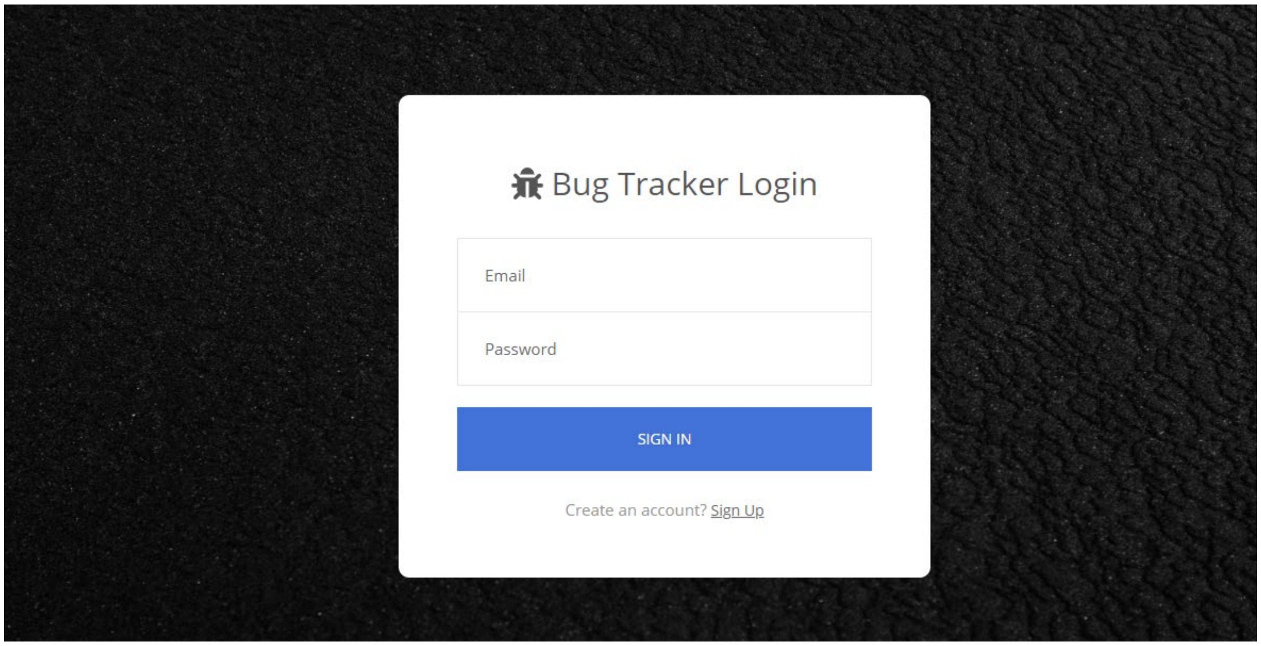


Figure 9: Login View

5.1.2. Registration:

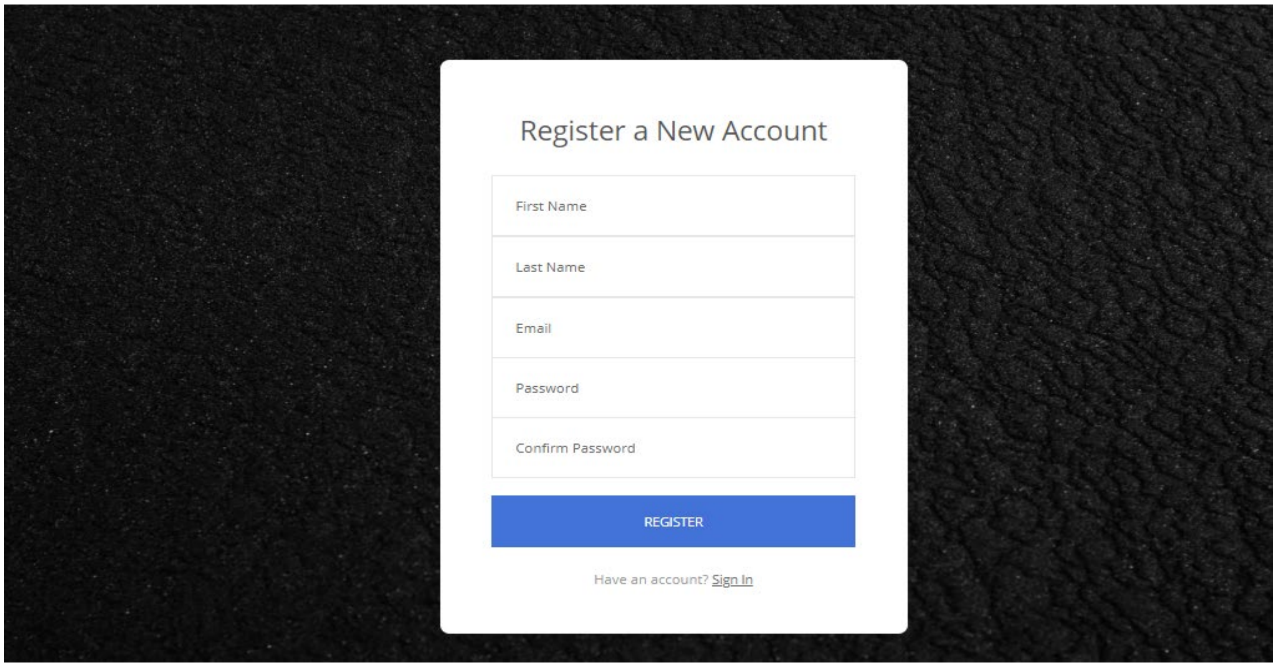
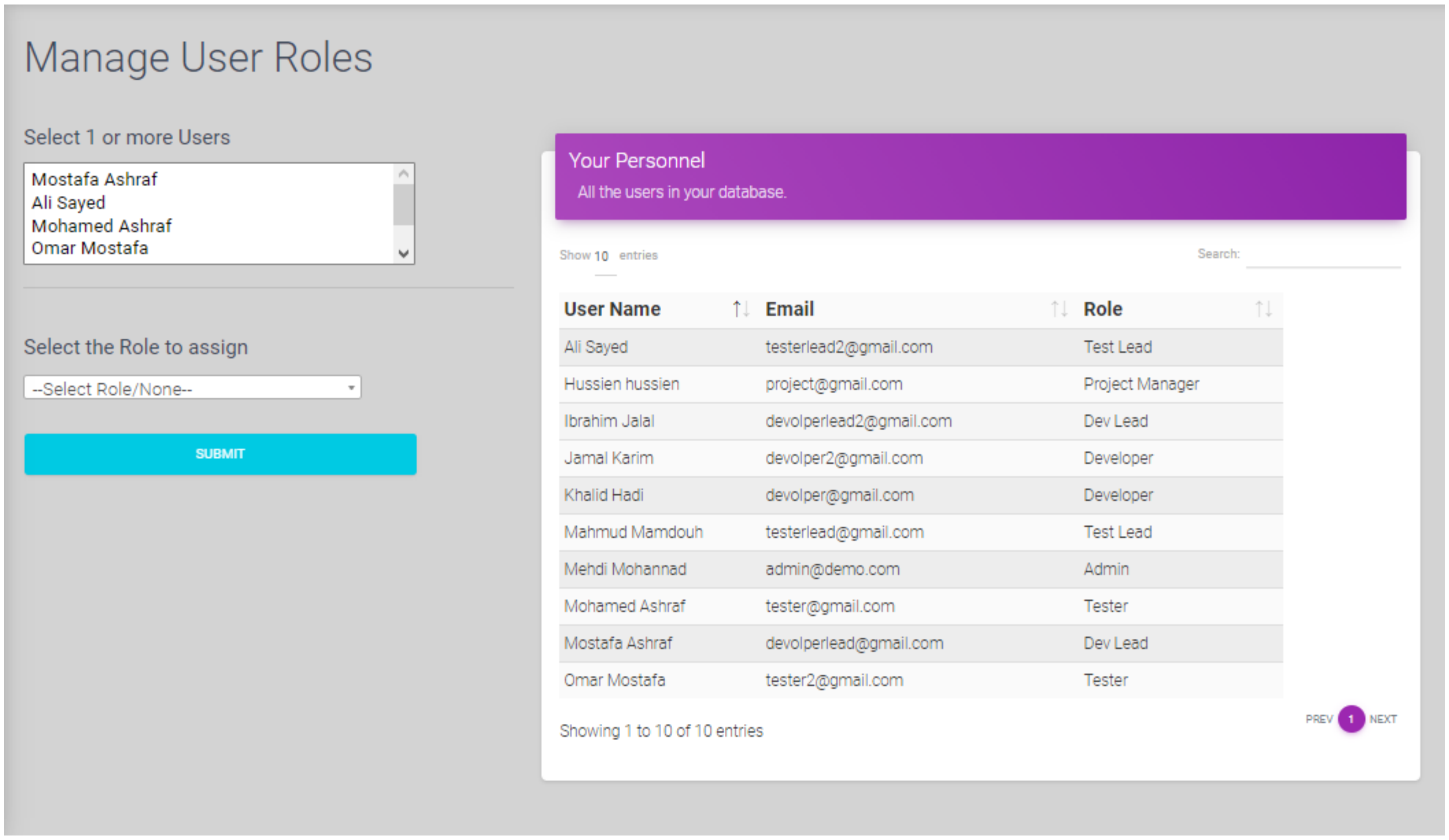


Figure 10:Registration View

5.1.3. Role Assigned:


Manage User Roles

Select 1 or more Users

Mostafa Ashraf
Ali Sayed
Mohamed Ashraf
Omar Mostafa

Select the Role to assign

--Select Role/None--

Your Personnel

All the users in your database.

Show 10 entries

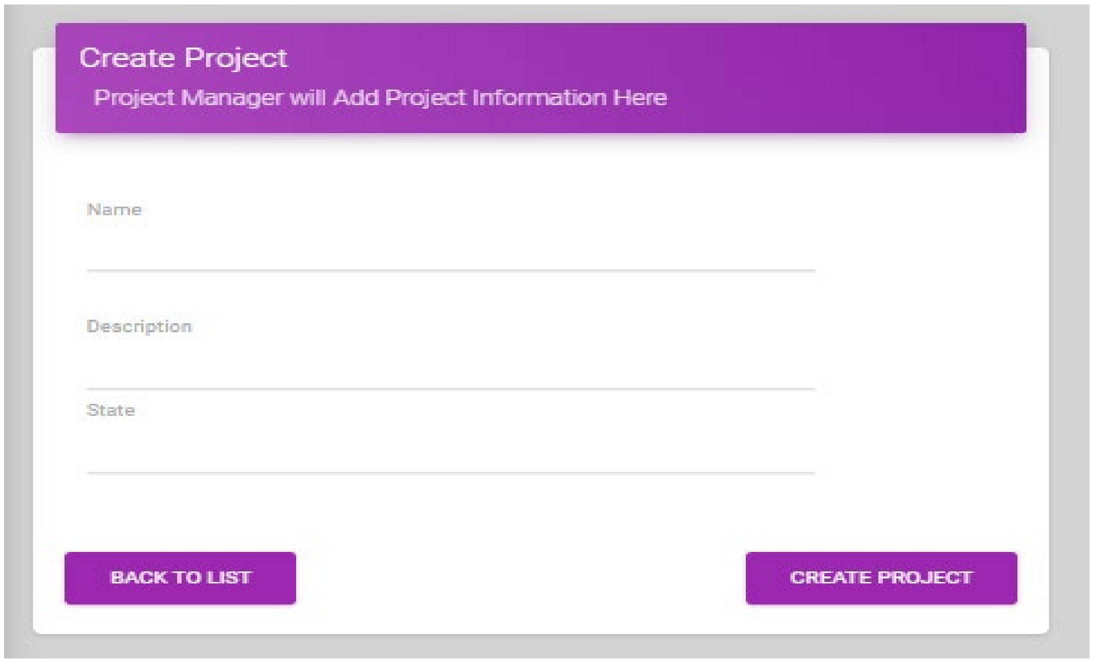
Search:

User Name	Email	Role
Ali Sayed	testerlead2@gmail.com	Test Lead
Hussien hussien	project@gmail.com	Project Manager
Ibrahim Jalal	devolperlead2@gmail.com	Dev Lead
Jamal Karim	devolper2@gmail.com	Developer
Khalid Hadi	devolper@gmail.com	Developer
Mahmud Mamdouh	testerlead@gmail.com	Test Lead
Mehdi Mohannad	admin@demo.com	Admin
Mohamed Ashraf	tester@gmail.com	Tester
Mostafa Ashraf	devolperlead@gmail.com	Dev Lead
Omar Mostafa	tester2@gmail.com	Tester

Showing 1 to 10 of 10 entries

Figure 11:Role Assigned View

5.1.4. Project Create:


Create Project

Project Manager will Add Project Information Here

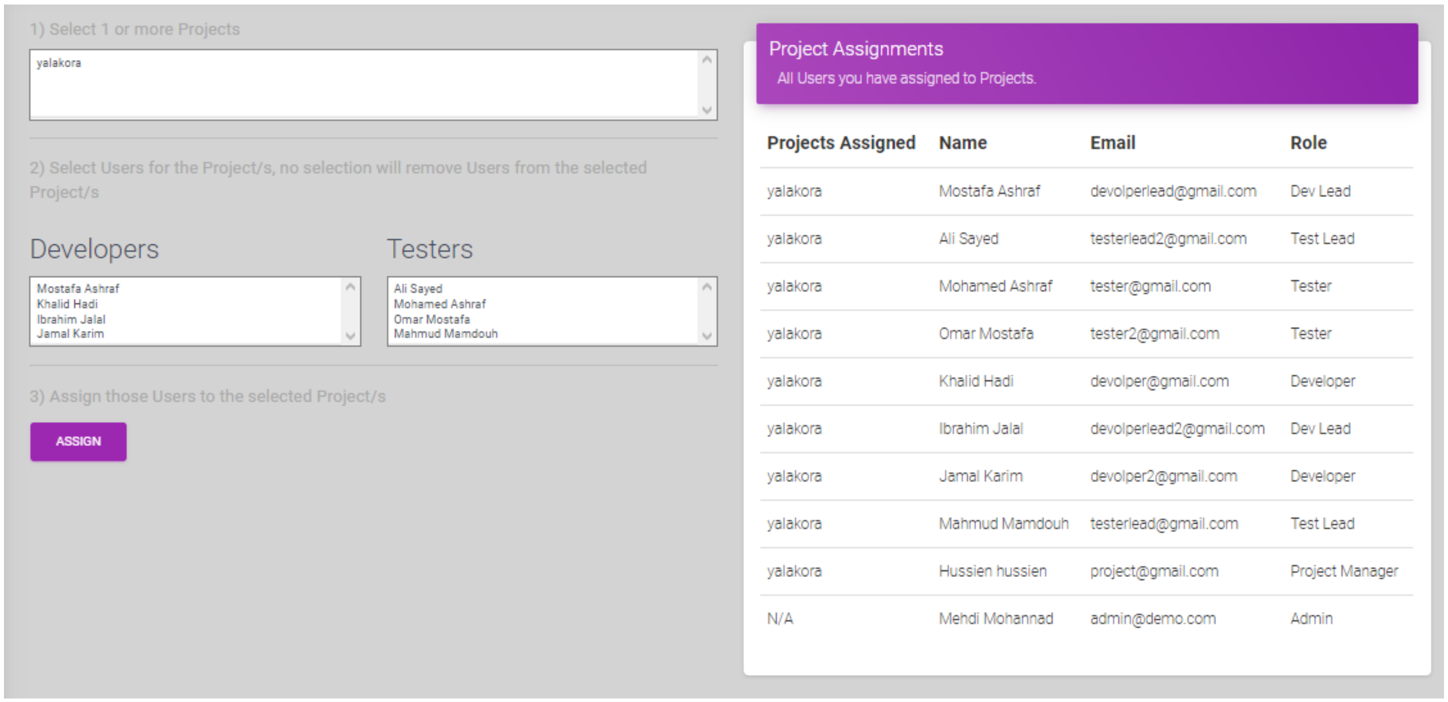
Name

Description

State

Figure 12:Create Project View

5.1.5. Manage Projects:



1) Select 1 or more Projects

yalakora

2) Select Users for the Project/s, no selection will remove Users from the selected Project/s

Developers

Mostafa Ashraf
Khalid Hadi
Ibrahim Jalal
Jamal Karim

Testers

Ali Sayed
Mohamed Ashraf
Omar Mostafa
Mahmud Mamdouh

3) Assign those Users to the selected Project/s

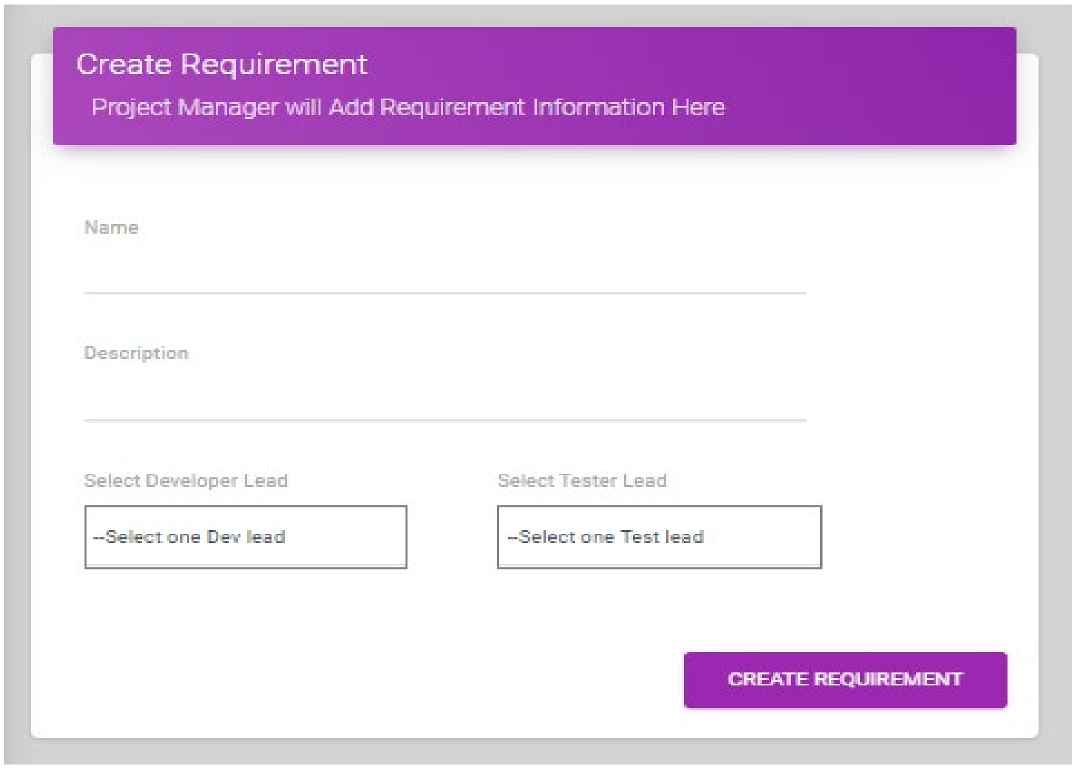
Project Assignments

All Users you have assigned to Projects.

Projects Assigned	Name	Email	Role
yalakora	Mostafa Ashraf	devolperlead@gmail.com	Dev Lead
yalakora	Ali Sayed	testerlead2@gmail.com	Test Lead
yalakora	Mohamed Ashraf	tester@gmail.com	Tester
yalakora	Omar Mostafa	tester2@gmail.com	Tester
yalakora	Khalid Hadi	devolper@gmail.com	Developer
yalakora	Ibrahim Jalal	devolperlead2@gmail.com	Dev Lead
yalakora	Jamal Karim	devolper2@gmail.com	Developer
yalakora	Mahmud Mamdouh	testerlead@gmail.com	Test Lead
yalakora	Hussien hussien	project@gmail.com	Project Manager
N/A	Mehdi Mohannad	admin@demo.com	Admin

Figure 13:Manage Project View

5.1.6. Create Requirement:



Create Requirement

Project Manager will Add Requirement Information Here

Name

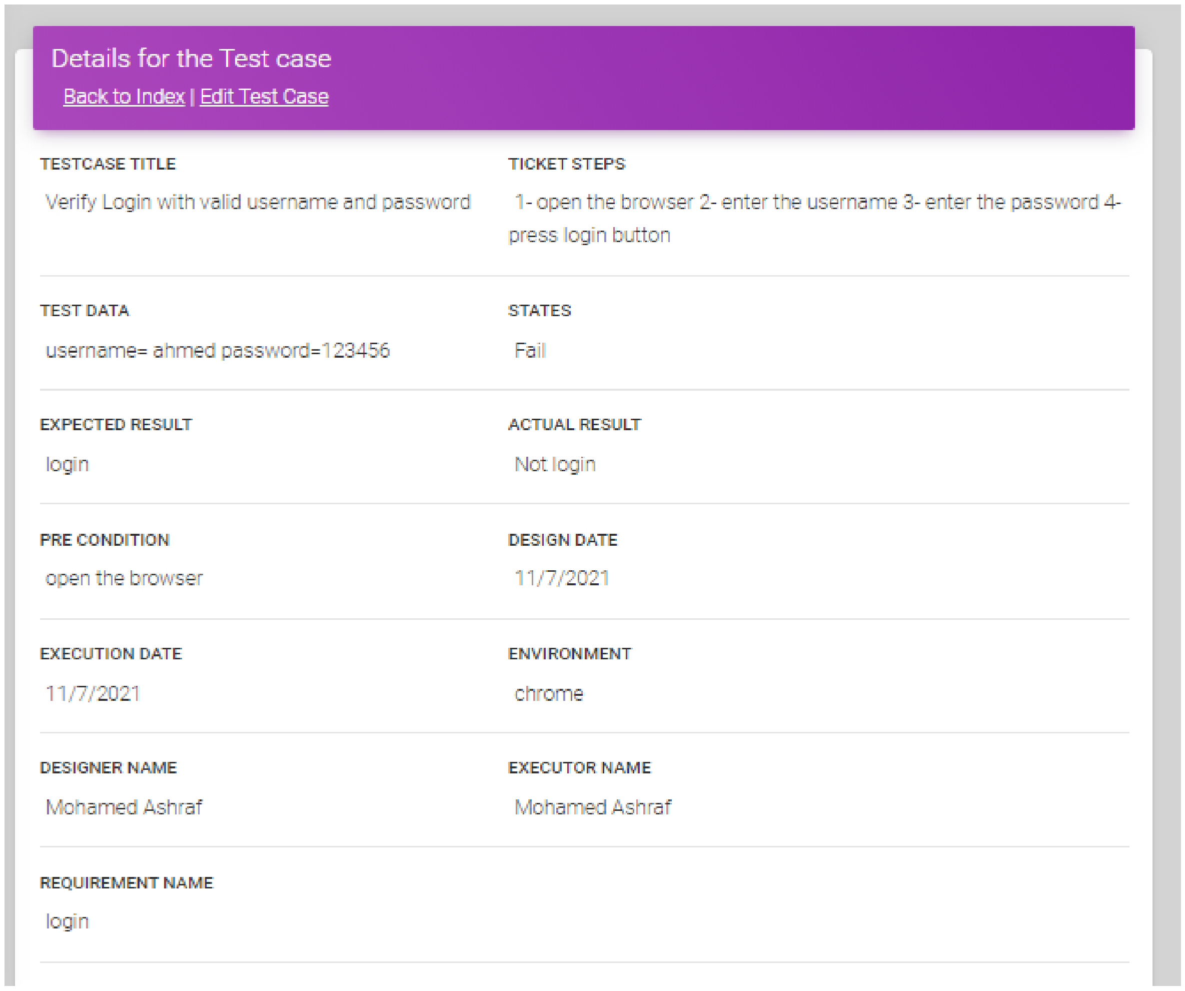
Description

Select Developer Lead: --Select one Dev lead

Select Tester Lead: --Select one Test lead

Figure 14:Create Requirement View

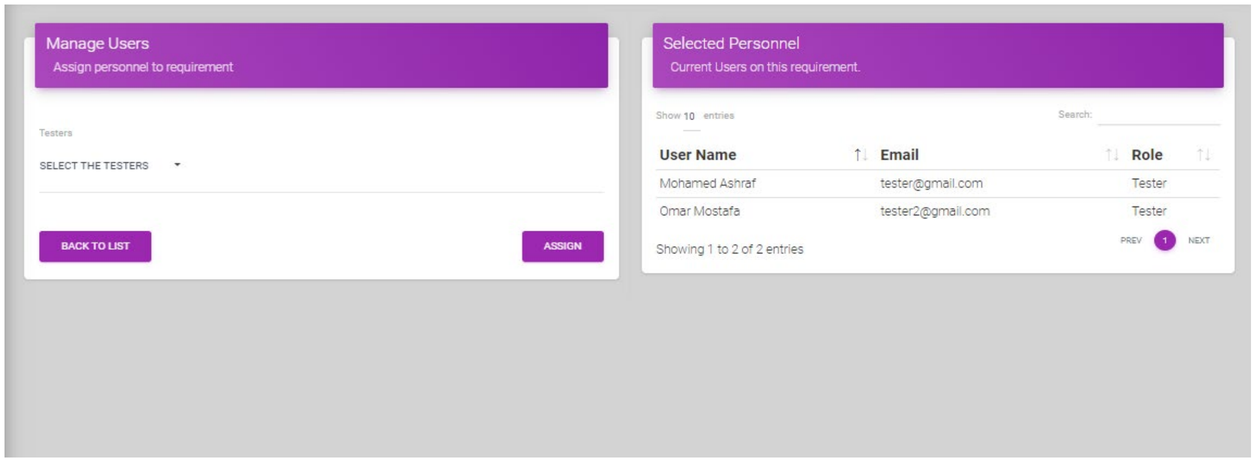
5.1.7. Detail of test case:


Details for the Test case
Back to Index | Edit Test Case

TESTCASE TITLE	TICKET STEPS
Verify Login with valid username and password	1- open the browser 2- enter the username 3- enter the password 4- press login button
TEST DATA	STATES
username= ahmed password=123456	Fail
EXPECTED RESULT	ACTUAL RESULT
login	Not login
PRE CONDITION	DESIGN DATE
open the browser	11/7/2021
EXECUTION DATE	ENVIRONMENT
11/7/2021	chrome
DESIGNER NAME	EXECUTOR NAME
Mohamed Ashraf	Mohamed Ashraf
REQUIREMENT NAME	
login	

Figure 15:Details of Test Case View

5.1.8. Assign Requirement:


Manage Users
Assign personnel to requirement

Testers
SELECT THE TESTERS

BACK TO LIST
ASSIGN

Selected Personnel
Current Users on this requirement.

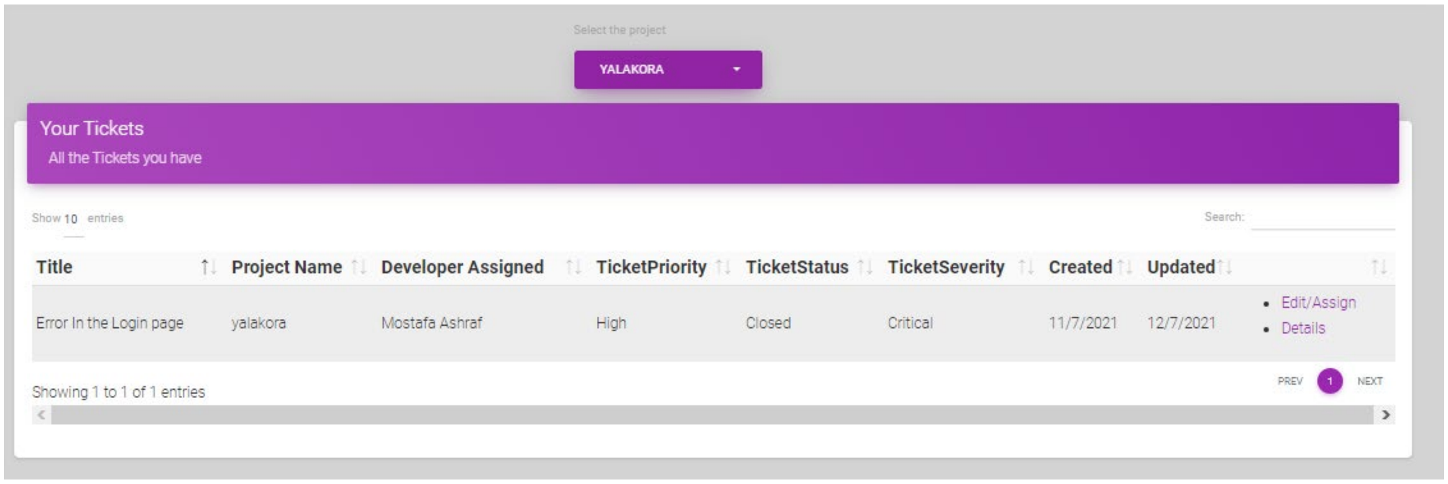
Show 10 entries
Search:

User Name	Email	Role
Mohamed Ashraf	tester@gmail.com	Tester
Omar Mostafa	tester2@gmail.com	Tester

Showing 1 to 2 of 2 entries
PREV 1 NEXT

Figure 16:Assign Requirement View

5.1.9. Ticket Index:



Select the project

YALAKORA

Your Tickets

All the Tickets you have

Show 10 entries

Search:

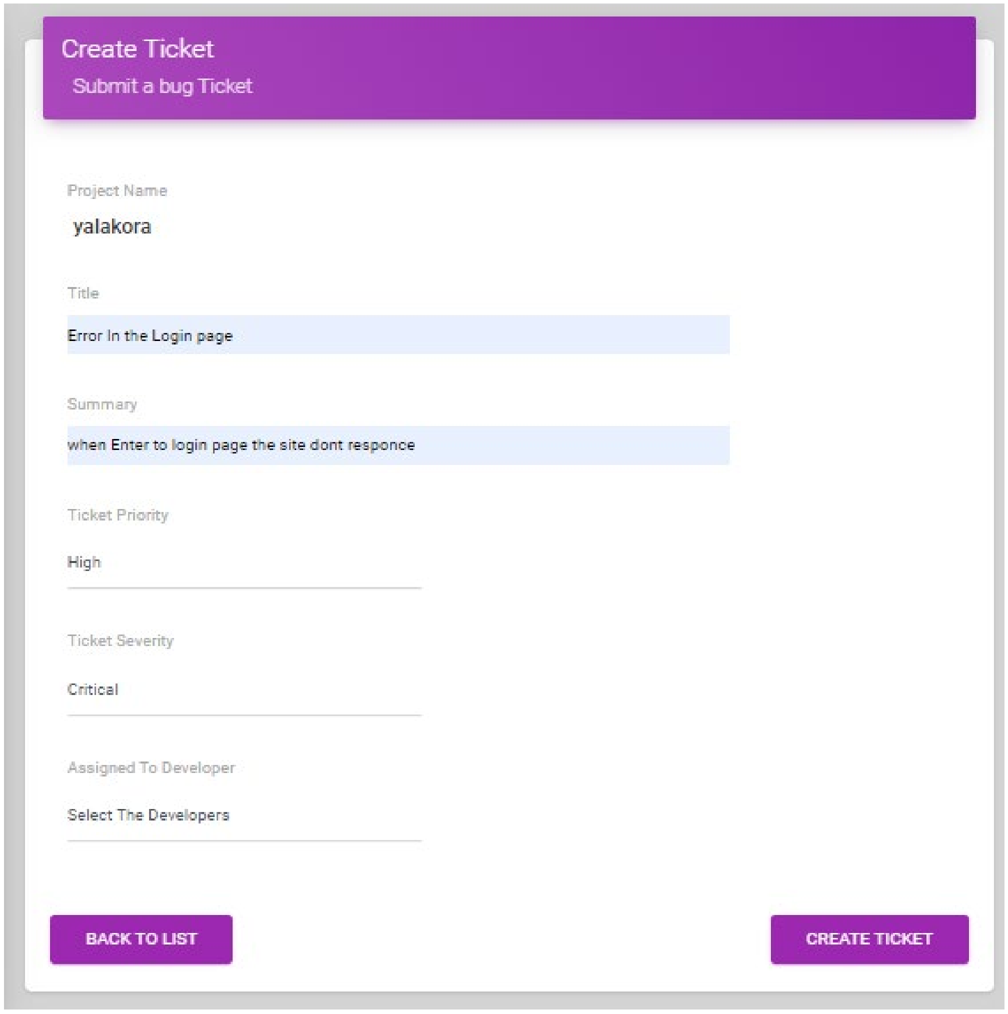
Title	Project Name	Developer Assigned	TicketPriority	TicketStatus	TicketSeverity	Created	Updated	
Error In the Login page	yalakora	Mostafa Ashraf	High	Closed	Critical	11/7/2021	12/7/2021	• Edit/Assign • Details

Showing 1 to 1 of 1 entries

PREV 1 NEXT

Figure 17:Ticket Index View

5.1.10. Create ticket:



Create Ticket

Submit a bug Ticket

Project Name

yalakora

Title

Error In the Login page

Summary

when Enter to login page the site dont responce

Ticket Priority

High

Ticket Severity

Critical

Assigned To Developer

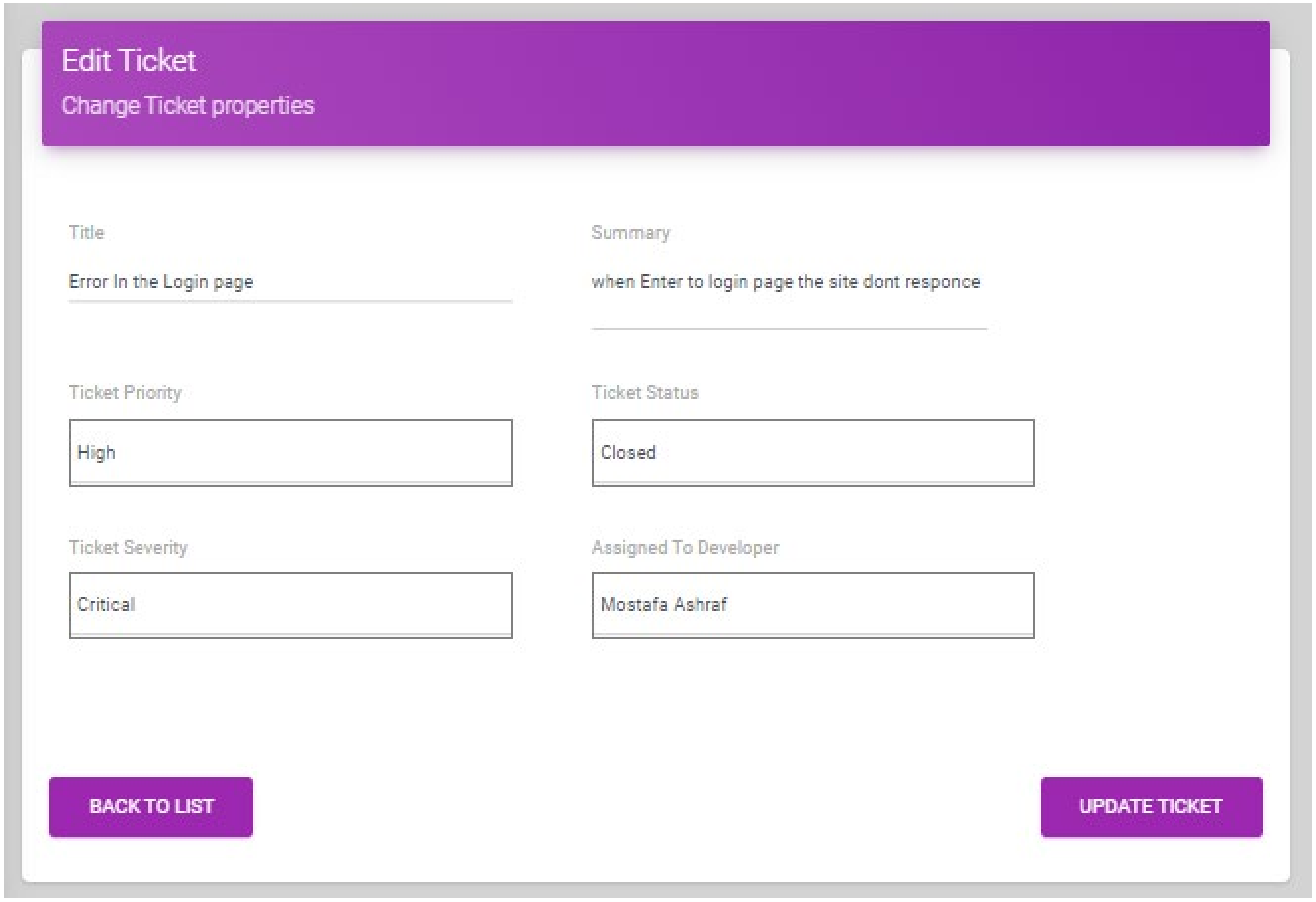
Select The Developers

BACK TO LIST

CREATE TICKET

Figure 18:Create Ticket View

5.1.11. Edit Ticket:


Edit Ticket

Change Ticket properties

Title: Error In the Login page

Summary: when Enter to login page the site dont responce

Ticket Priority: High

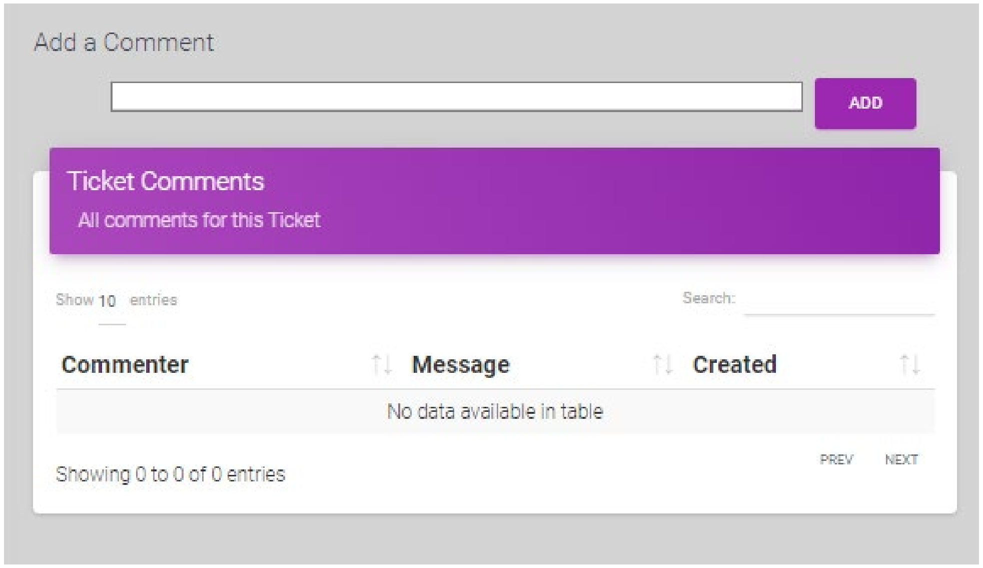
Ticket Status: Closed

Ticket Severity: Critical

Assigned To Developer: Mostafa Ashraf

Figure 19:Edit Ticket View

5.1.12. Ticket Comments:


Add a Comment

Ticket Comments

All comments for this Ticket

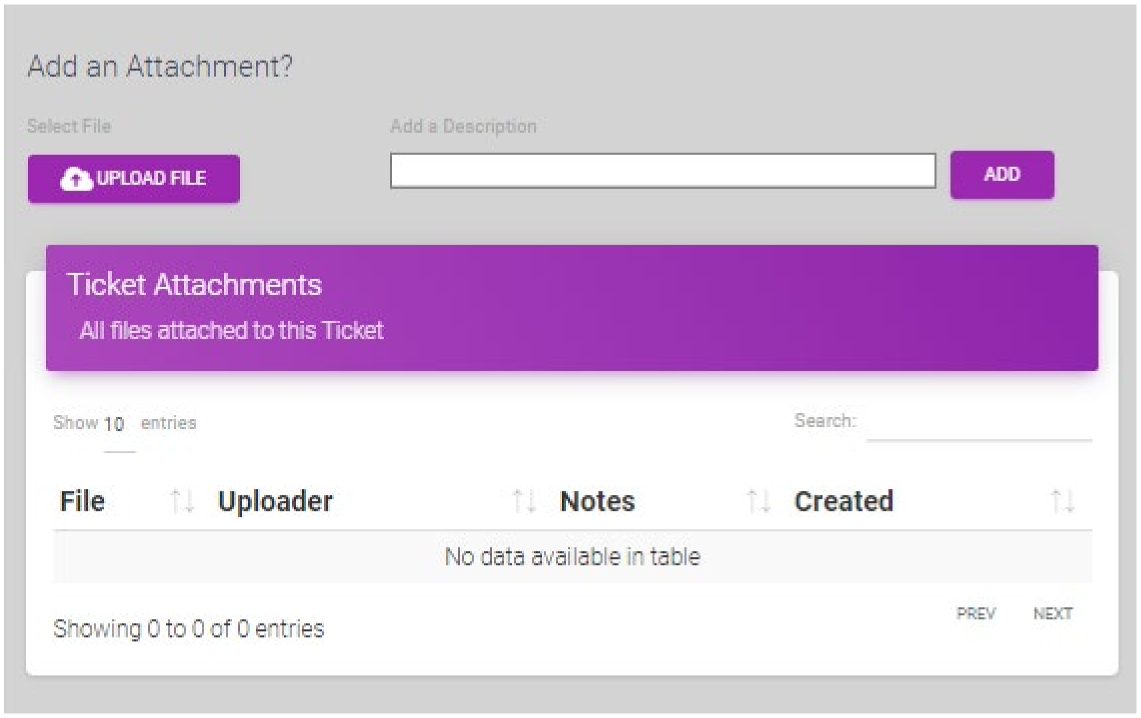
Show 10 entries Search:

Commenter	Message	Created
No data available in table		

Showing 0 to 0 of 0 entries

Figure 20:Ticket Comment View

5.1.13. Ticket Attachments:


Add an Attachment?

Select File

Add a Description

Ticket Attachments

All files attached to this Ticket

Show 10 entries

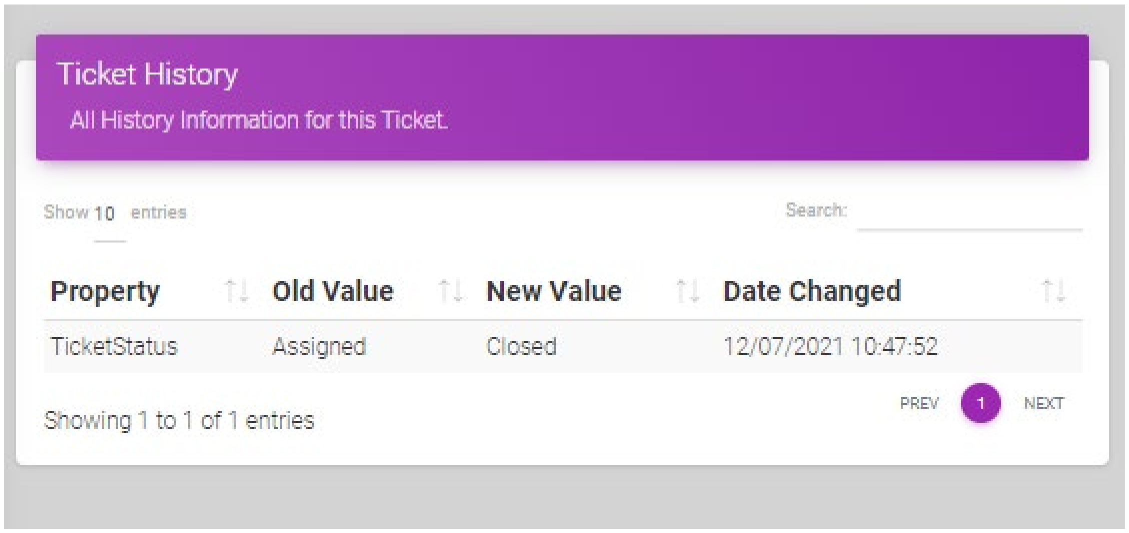
Search:

File	Uploader	Notes	Created
No data available in table			

Showing 0 to 0 of 0 entries

Figure 21:Ticket Attachment View

5.1.14. Ticket History:


Ticket History

All History Information for this Ticket.

Show 10 entries

Search:

Property	Old Value	New Value	Date Changed
TicketStatus	Assigned	Closed	12/07/2021 10:47:52

Showing 1 to 1 of 1 entries

Figure 22:Ticket History View

CHAPTER 6: Testing

6. CHAPTER 6: TESTING

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.

There are many different types of software tests, each with specific objectives and strategies:

- **Acceptance testing:** Verifying whether the whole system works as intended.
- **Integration testing:** Ensuring that software components or functions operate together.
- **Unit testing:** Validating that each software unit performs as expected. A unit is the smallest testable component of an application.
- **Functional testing:** Checking functions by emulating business scenarios, based on functional requirements. Black-box testing is a common way to verify functions.

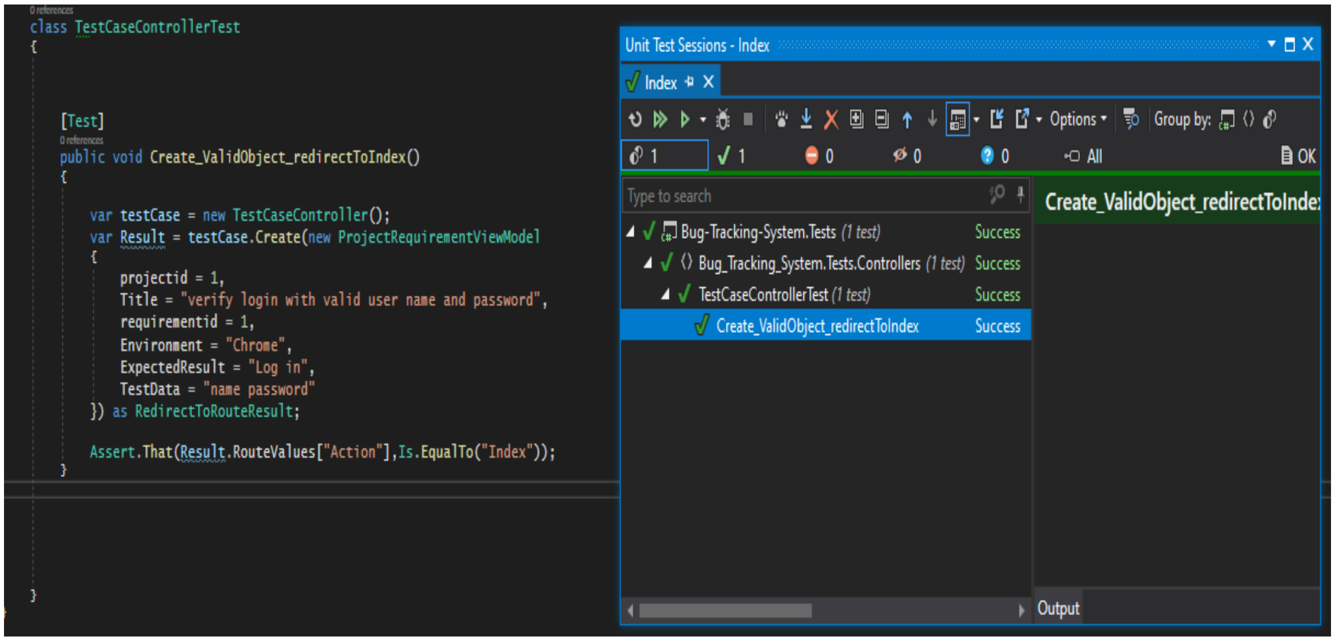
6.1. UNIT TESTING:

Component testing (also known as unit or module testing) focuses on components that are separately testable.

The type of testing used to make unit testing is black box testing (Equivalence partitioning).

Create Test case:

TC#	Project Requirement view model: Object	output
1.	Valid object	Output
2.	Invalid object	Error
3.	No Entered value	Error



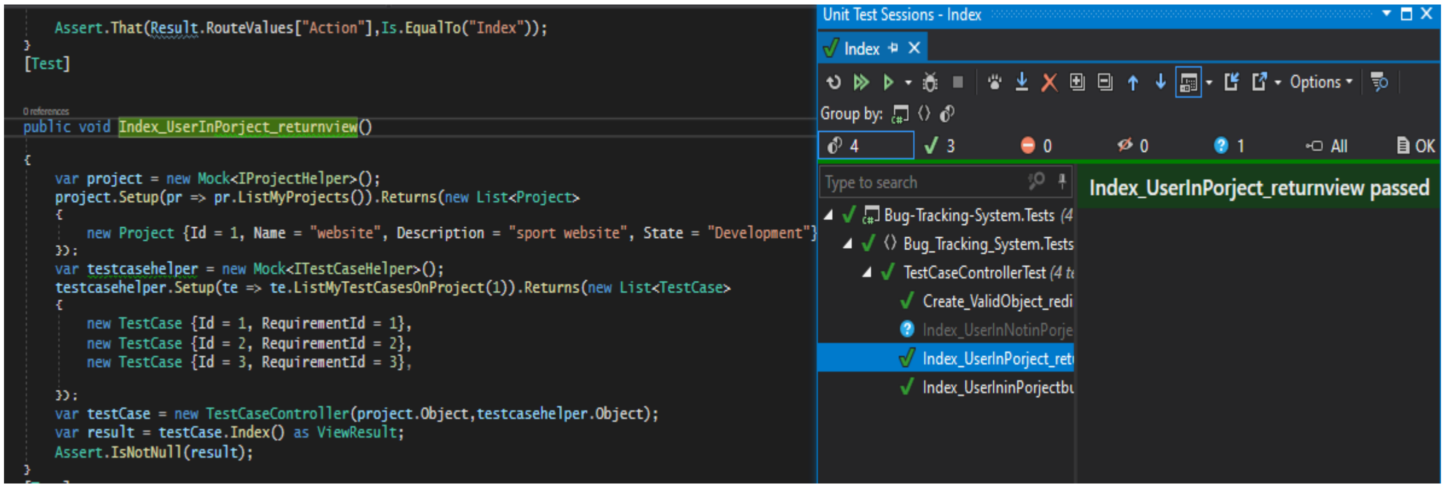
```
0 references
class TestCaseControllerTest
{


    [Test]
    0 references
    public void Create_ValidObject_redirectToIndex()
    {

        var testCase = new TestCaseController();
        var Result = testCase.Create(new ProjectRequirementViewModel
        {
            projectid = 1,
            Title = "verify login with valid user name and password",
            requirementid = 1,
            Environment = "Chrome",
            ExpectedResult = "Log in",
            TestData = "name password"
        }) as RedirectToRouteResult;

        Assert.That(Result.RouteValues["Action"],Is.EqualTo("Index"));
    }




}
```

```
Unit Test Sessions - Index
Index
1    ✓ 1    0    0    0    All    OK
Bug-Tracking-System.Tests (1 test)                        Success
  Bug_Tracking_System.Tests.Controllers (1 test)          Success
    TestCaseControllerTest (1 test)                       Success
      Create_ValidObject_redirectToIndex                  Success
Create_ValidObject_redirectToInde
Output
```

Figure 23:Show NUnit test for Create Test case

Test Case Index:

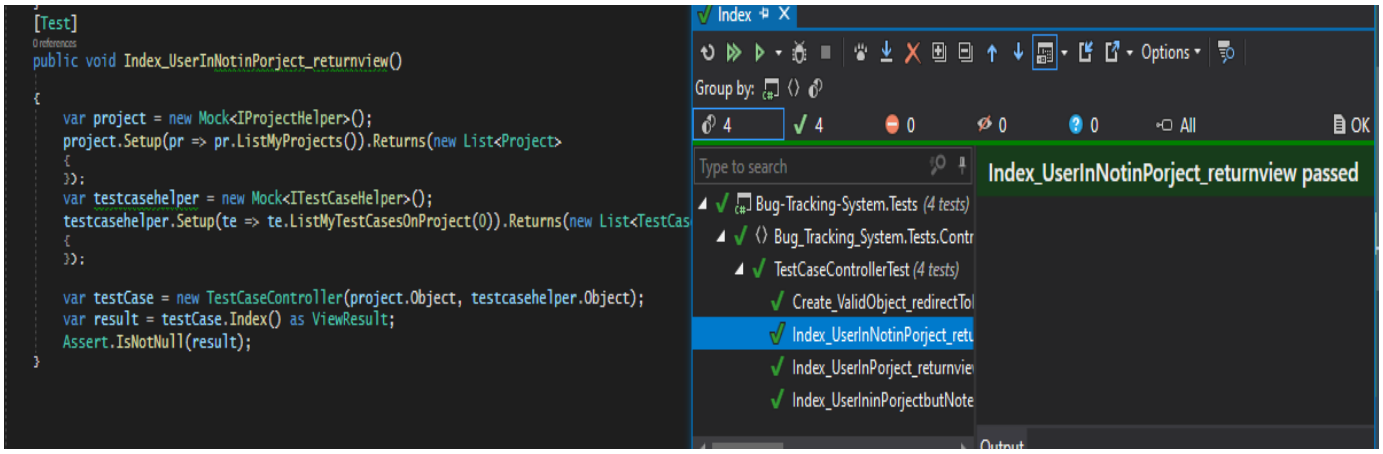
TC#	Route Value	output
1.	Valid	Output
2.	Invalid	Error
3.	No Entered Value	Error



```
        Assert.That(Result.RouteValues["Action"],Is.EqualTo("Index"));
    }
    [Test]

    0 references
    public void Index_UserInPorject_returnview()

    {
        var project = new Mock<IProjectHelper>();
        project.Setup(pr => pr.ListMyProjects()).Returns(new List<Project>
        {
            new Project {Id = 1, Name = "website", Description = "sport website", State = "Development"}
        });
        var testcasehelper = new Mock<ITestCaseHelper>();
        testcasehelper.Setup(te => te.ListMyTestCasesOnProject(1)).Returns(new List<TestCase>
        {
            new TestCase {Id = 1, RequirementId = 1},
            new TestCase {Id = 2, RequirementId = 2},
            new TestCase {Id = 3, RequirementId = 3},

        });
        var testCase = new TestCaseController(project.Object,testcasehelper.Object);
        var result = testCase.Index() as ViewResult;
        Assert.IsNotNull(result);
    }
```

```
Unit Test Sessions - Index
Index
Group by:
4    ✓ 3    0    0    1    All    OK
Bug-Tracking-System.Tests (4
  Bug_Tracking_System.Tests
    TestCaseControllerTest (4 t
      Create_ValidObject_redi
      Index_UserInNotinPorje
      Index_UserInPorject_retu
      Index_UserIninPorjectbu
Index_UserInPorject_returnview passed
```

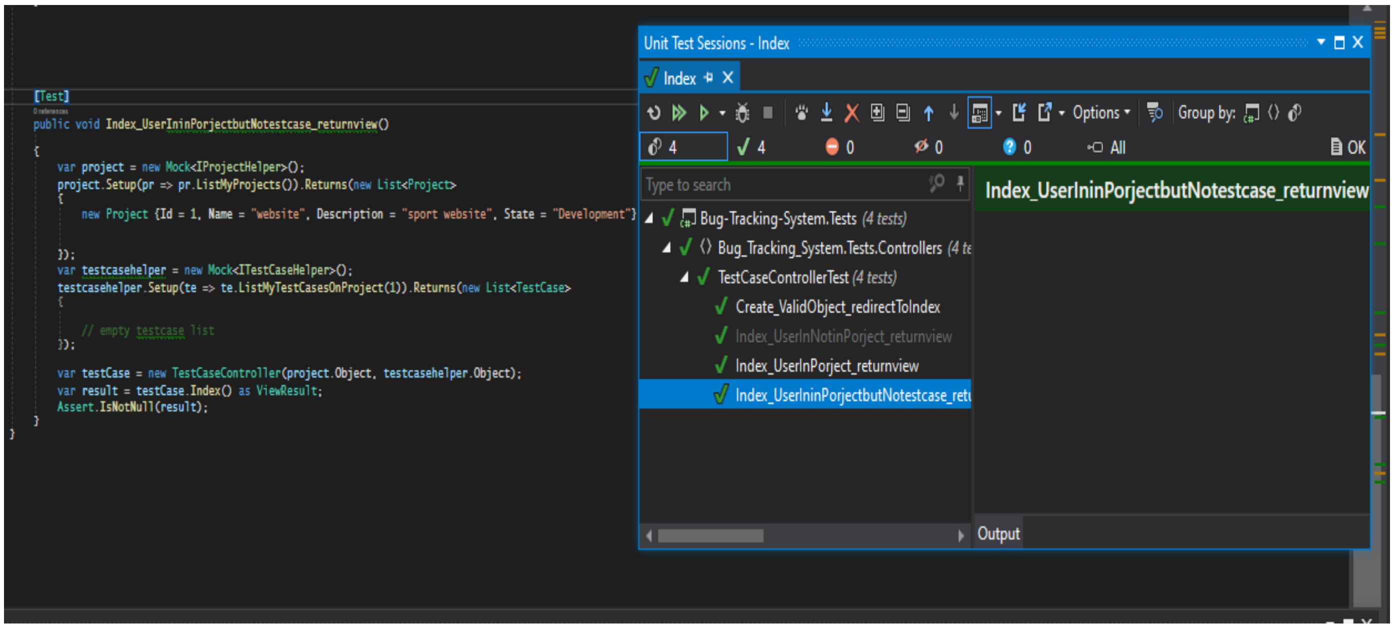
Figure 24:Show NUnit test for Test case Index when user in project and have testcases



```
[Test]
0 references
public void Index_UserInNotinPorject_returnview()

{
    var project = new Mock<IProjectHelper>();
    project.Setup(pr => pr.ListMyProjects()).Returns(new List<Project>
    {
    });
    var testcasehelper = new Mock<ITestCaseHelper>();
    testcasehelper.Setup(te => te.ListMyTestCasesOnProject(0)).Returns(new List<TestCas
    {
    });

    var testCase = new TestCaseController(project.Object, testcasehelper.Object);
    var result = testCase.Index() as ViewResult;
    Assert.IsNotNull(result);
}
```

```
Index_UserInNotinPorject_returnview passed

Bug-Tracking-System.Tests (4 tests)
  Bug_Tracking_System.Tests.Contr
    TestCaseControllerTest (4 tests)
      Create_ValidObject_redirectTo
      Index_UserInNotinPorject_retu
      Index_UserInPorject_returnvie
      Index_UserIninPorjectbutNote
```

Figure 25:Show NUnit test for Test case Index when user is not in project



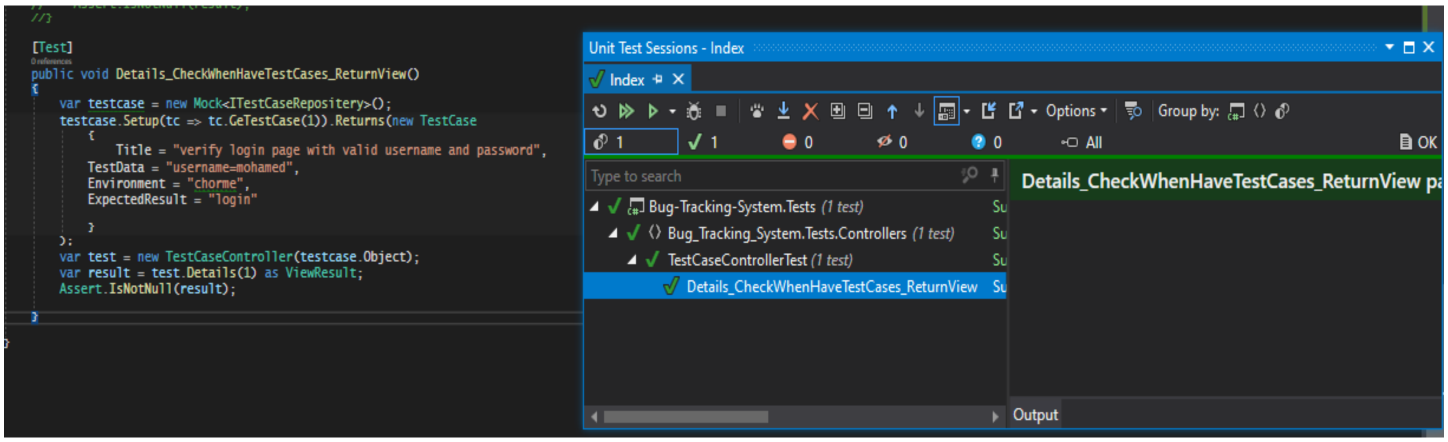
```
[Test]
0 references
public void Index_UserIninPorjectbutNotestcase_returnview()

{
    var project = new Mock<IProjectHelper>();
    project.Setup(pr => pr.ListMyProjects()).Returns(new List<Project>
    {
        new Project {Id = 1, Name = "website", Description = "sport website", State = "Development"}

    });
    var testcasehelper = new Mock<ITestCaseHelper>();
    testcasehelper.Setup(te => te.ListMyTestCasesOnProject(1)).Returns(new List<TestCase>
    {

        // empty testcase list
    });

    var testCase = new TestCaseController(project.Object, testcasehelper.Object);
    var result = testCase.Index() as ViewResult;
    Assert.IsNotNull(result);
}
}
```

```
Index_UserIninPorjectbutNotestcase_returnview

Bug-Tracking-System.Tests (4 tests)
  Bug_Tracking_System.Tests.Controllers (4 te
    TestCaseControllerTest (4 tests)
      Create_ValidObject_redirectToIndex
      Index_UserInNotinPorject_returnview
      Index_UserInPorject_returnview
      Index_UserIninPorjectbutNotestcase_retu
```

Figure 26:Show NUnit test for Test case Index when user is in project but not have test case

Test Case Detail:

TC#	Id	output
1	Valid	Output
2	Invalid	Error
3	No Entered value	Error



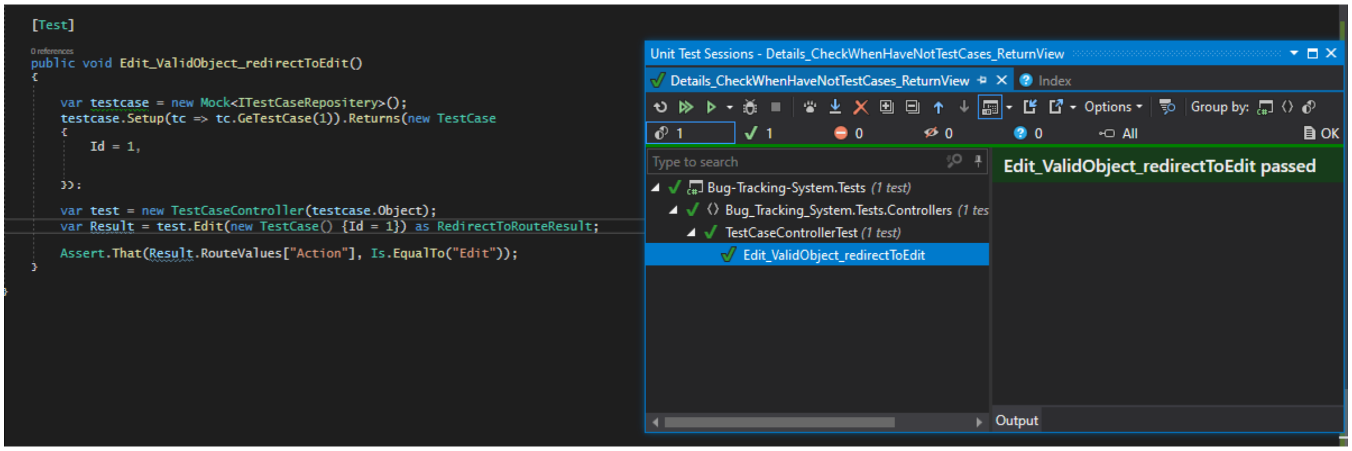
```
    //}

    [Test]
    0 references
    public void Details_CheckWhenHaveTestCases_ReturnView()
    {
        var testcase = new Mock<ITestCaseRepositery>();
        testcase.Setup(tc => tc.GeTestCase(1)).Returns(new TestCase
            {
                Title = "verify login page with valid username and password",
            TestData = "username=mohamed",
            Environment = "chorme",
            ExpectedResult = "login"

            }
        );
        var test = new TestCaseController(testcase.Object);
        var result = test.Details(1) as ViewResult;
        Assert.IsNotNull(result);

    }
}
```

```
Unit Test Sessions - Index
Index
1    ✓ 1    0    0    0    All    OK
Bug-Tracking-System.Tests (1 test)    Su
  Bug_Tracking_System.Tests.Controllers (1 test)    Su
    TestCaseControllerTest (1 test)    Su
      Details_CheckWhenHaveTestCases_ReturnView    Su
Details_CheckWhenHaveTestCases_ReturnView pa
Output
```

Figure 27:Show NUnit test for Test case Detail when have valid Id

Edit Test case:

TC#	Test Case: Object	output
1	Valid object	Output
2	Invalid object	Error
3	No Entered value	Error



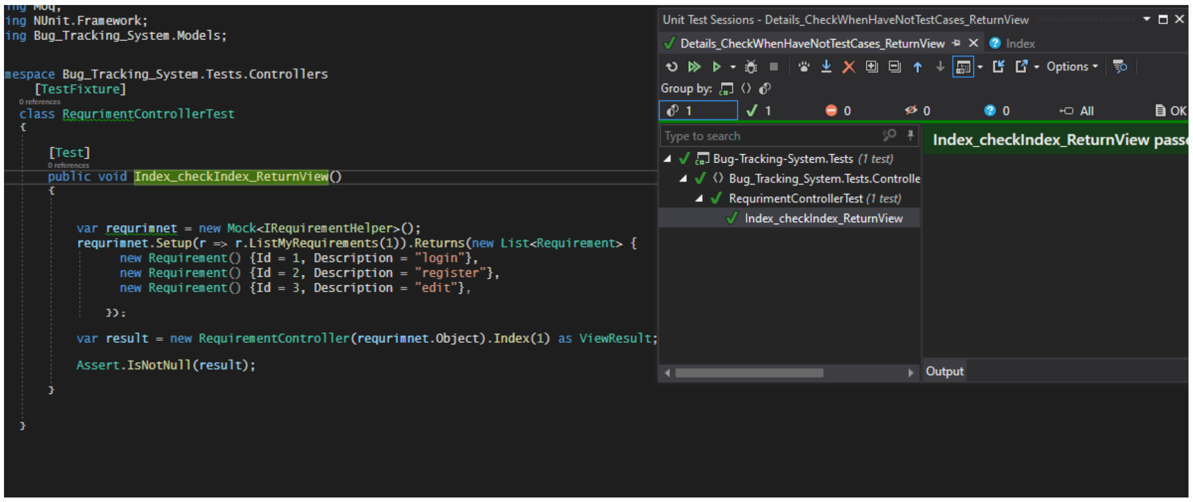
```
    [Test]

    0 references
    public void Edit_ValidObject_redirectToEdit()
    {

        var testcase = new Mock<ITestCaseRepositery>();
        testcase.Setup(tc => tc.GeTestCase(1)).Returns(new TestCase
        {
            Id = 1,


        });

        var test = new TestCaseController(testcase.Object);
        var Result = test.Edit(new TestCase() {Id = 1}) as RedirectToRouteResult;

        Assert.That(Result.RouteValues["Action"], Is.EqualTo("Edit"));
    }
}
```

```
Unit Test Sessions - Details_CheckWhenHaveNotTestCases_ReturnView
Details_CheckWhenHaveNotTestCases_ReturnView    Index
1    ✓ 1    0    0    0    All    OK
Bug-Tracking-System.Tests (1 test)
  Bug_Tracking_System.Tests.Controllers (1 tes
    TestCaseControllerTest (1 test)
      Edit_ValidObject_redirectToEdit
Edit_ValidObject_redirectToEdit passed
Output
```

Figure 28:Show NUnit test for Test case Edit

Requirement Index:

TC#	Id	output
1.	Valid	Output
2.	Invalid	Error
3.	Not enter	Invalid



```
ing Moq;
ing NUnit.Framework;
ing Bug_Tracking_System.Models;

espace Bug_Tracking_System.Tests.Controllers
    [TestFixture]
  0 references
  class RequrimentControllerTest
  {

      [Test]
      0 references
      public void Index_checkIndex_ReturnView()
      {


          var requrimnet = new Mock<IRequirementHelper>();
          requrimnet.Setup(r => r.ListMyRequirements(1)).Returns(new List<Requirement> {
                new Requirement() {Id = 1, Description = "login"},
                new Requirement() {Id = 2, Description = "register"},
                new Requirement() {Id = 3, Description = "edit"},

              });

          var result = new RequirementController(requrimnet.Object).Index(1) as ViewResult;

          Assert.IsNotNull(result);

      }


  }
```

Unit Test Sessions - Details_CheckWhenHaveNotTestCases_ReturnView
Details_CheckWhenHaveNotTestCases_ReturnView
Index
Options
Group by:
1
✓ 1
0
0
0
All
OK
Type to search

```
✓ Bug-Tracking-System.Tests (1 test)
  ✓ Bug_Tracking_System.Tests.Controlle
    ✓ RequrimentControllerTest (1 test)
      ✓ Index_checkIndex_ReturnView
```

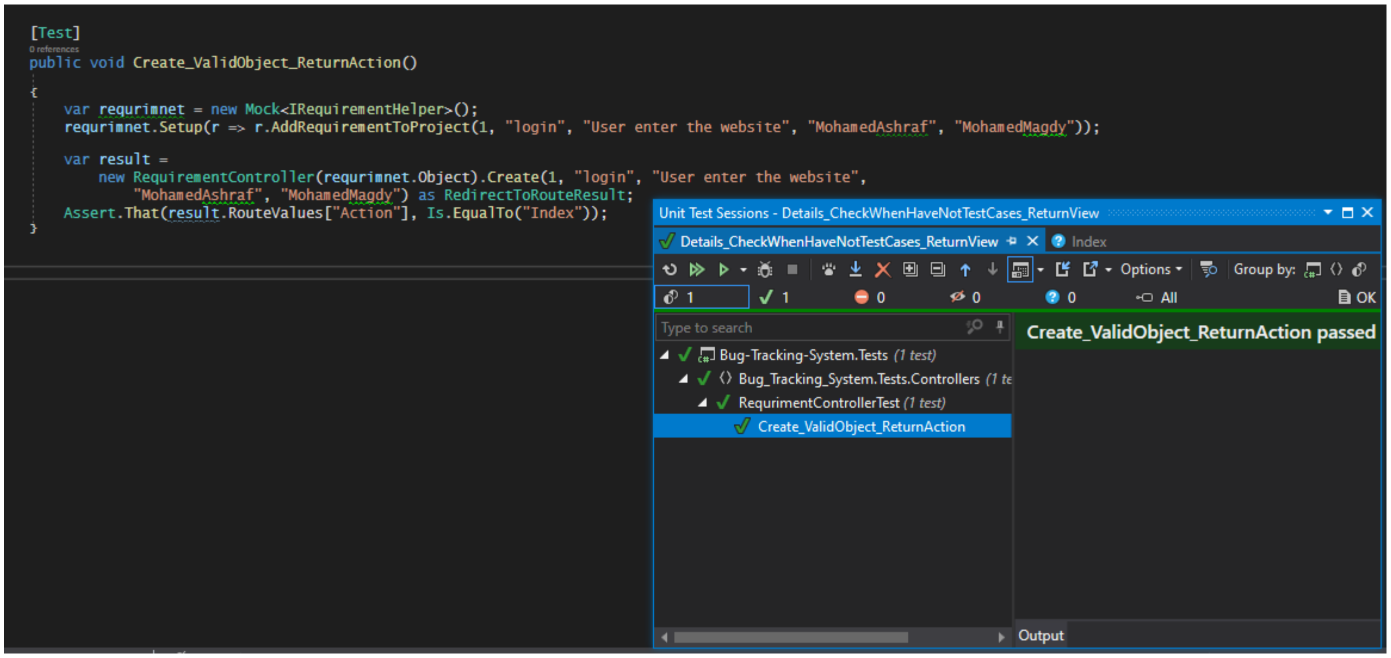
Index_checkIndex_ReturnView passe

Output

Figure 29:Show NUnit test for Requirement Index

Create Requirement:

TC#	Id	Name	Description	Dev leads	Test leads	output
1.	Valid	Valid	Valid	Valid	Valid	Output
2.	Invalid	Any	Any	Any	Any	Error
3.	Any	Invalid	Any	Any	Any	Error
4.	Any	Any	Invalid	Any	Any	Error
5.	Any	Any	Any	Invalid	Any	Error
6.	Any	Any	Any	Any	Invalid	Error
7.	Invalid	Invalid	Invalid	Invalid	Invalid	Error



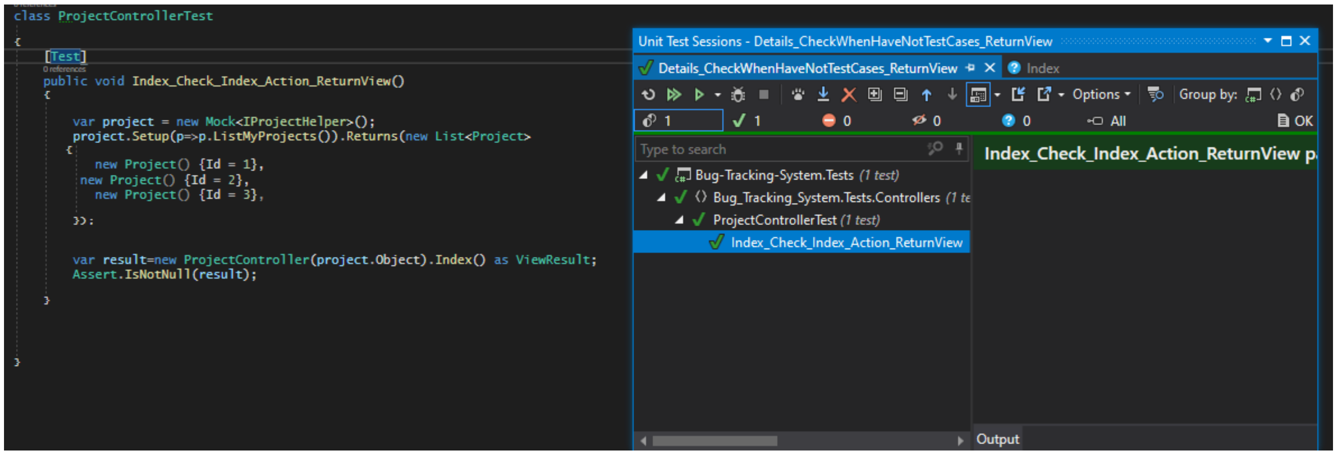
```
[Test]
0 references
public void Create_ValidObject_ReturnAction()
{
    var requrimnet = new Mock<IRequirementHelper>();
    requrimnet.Setup(r => r.AddRequirementToProject(1, "login", "User enter the website", "MohamedAshraf", "MohamedMagdy"));

    var result =
        new RequirementController(requrimnet.Object).Create(1, "login", "User enter the website",
            "MohamedAshraf", "MohamedMagdy") as RedirectToRouteResult;
    Assert.That(result.RouteValues["Action"], Is.EqualTo("Index"));
}
```

```
Unit Test Sessions - Details_CheckWhenHaveNotTestCases_ReturnView
Details_CheckWhenHaveNotTestCases_ReturnView    Index
1   ✓ 1   0   0   0   All   OK
Create_ValidObject_ReturnAction passed
Bug-Tracking-System.Tests (1 test)
  Bug_Tracking_System.Tests.Controllers (1 te
    RequrimentControllerTest (1 test)
      Create_ValidObject_ReturnAction
```

Figure 30:Show NUnit test for Create Requirement

Project Index:

TC#	Route Value	output
1.	Valid	Output
2.	Invalid	Error
3.	No Entered Value	Error



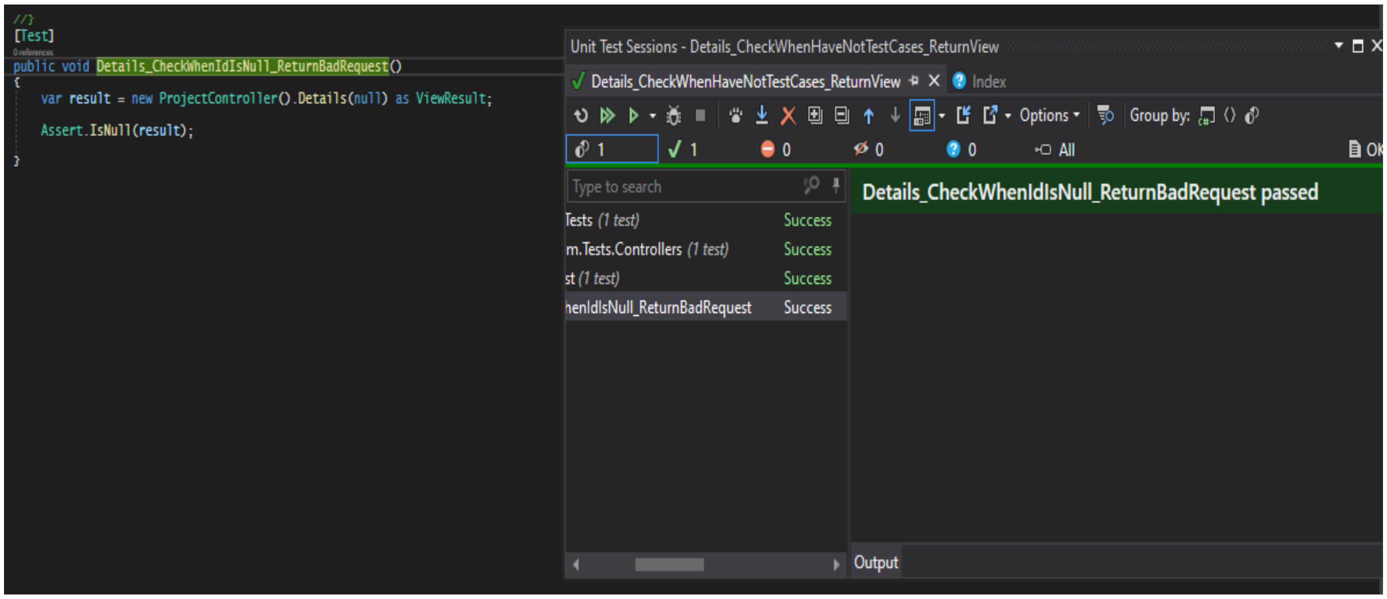
```
class ProjectControllerTest
{
    [Test]
    0 references
    public void Index_Check_Index_Action_ReturnView()
    {

        var project = new Mock<IProjectHelper>();
        project.Setup(p=>p.ListMyProjects()).Returns(new List<Project>
       {
           new Project() {Id = 1},
         new Project() {Id = 2},
           new Project() {Id = 3},

        });


        var result=new ProjectController(project.Object).Index() as ViewResult;
        Assert.IsNotNull(result);

    }


}
```

```
Unit Test Sessions - Details_CheckWhenHaveNotTestCases_ReturnView
Details_CheckWhenHaveNotTestCases_ReturnView    Index
1    1    0    0    0    All    OK
Bug-Tracking-System.Tests (1 test)
  Bug_Tracking_System.Tests.Controllers (1 te
    ProjectControllerTest (1 test)
      Index_Check_Index_Action_ReturnView
Index_Check_Index_Action_ReturnView p
```

Figure 31:Show NUnit test for Project Index

Project Detail:

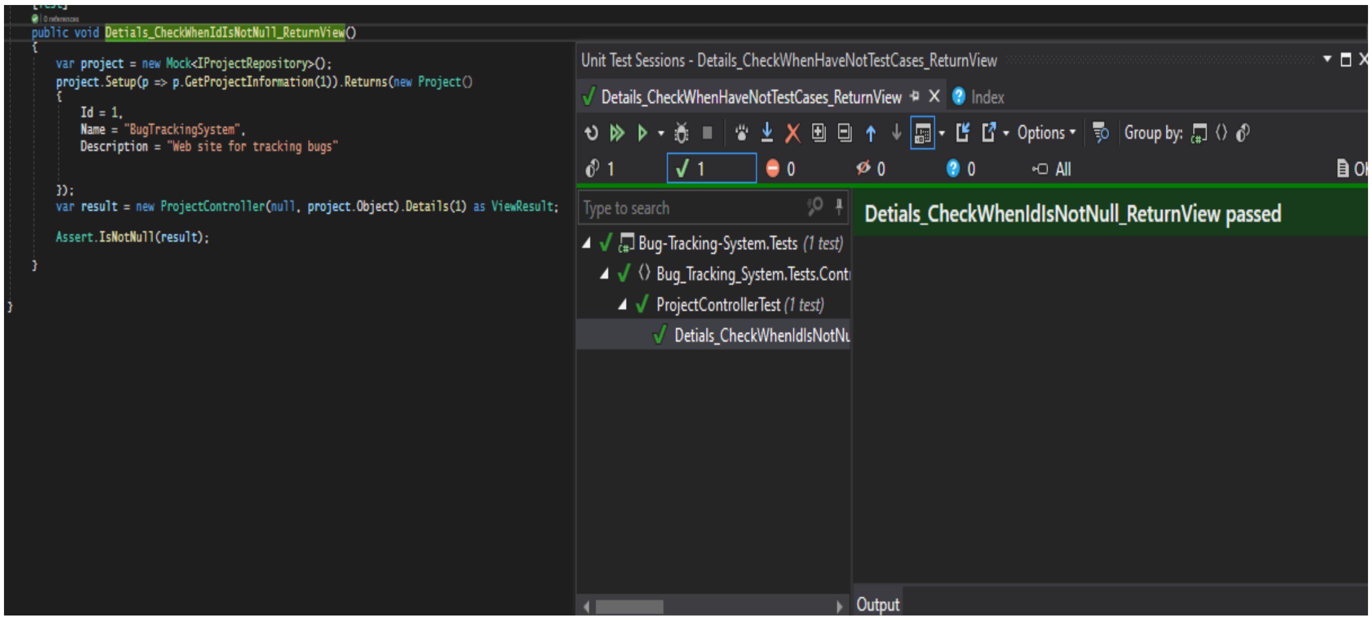
TC#	Id	output
1	Valid	Output
2	Invalid	Error
3	No Entered value	Error



```
//}
[Test]
0 references
public void Details_CheckWhenIdIsNull_ReturnBadRequest()
{
    var result = new ProjectController().Details(null) as ViewResult;

    Assert.IsNull(result);

}
```

```
Unit Test Sessions - Details_CheckWhenHaveNotTestCases_ReturnView
Details_CheckWhenHaveNotTestCases_ReturnView    Index
Options    Group by:
1   ✓ 1   0   0   0   All
Details_CheckWhenIdIsNull_ReturnBadRequest passed
Tests (1 test)                          Success
m.Tests.Controllers (1 test)            Success
st (1 test)                             Success
henIdIsNull_ReturnBadRequest            Success
Output
```

Figure 33:Show NUnit test for Project Detail when Id is Invalid Value



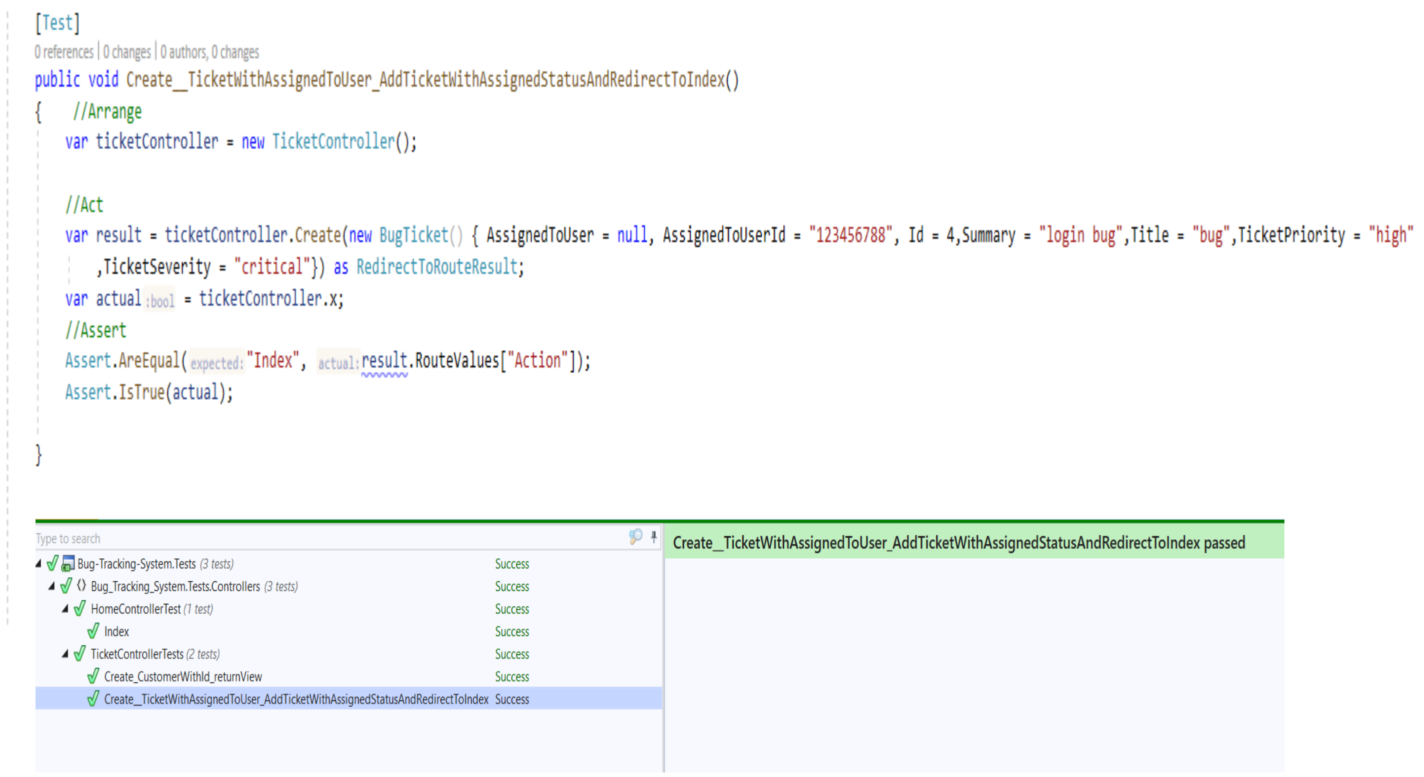
```
0 references
public void Detials_CheckWhenIdIsNotNull_ReturnView()
{
    var project = new Mock<IProjectRepository>();
    project.Setup(p => p.GetProjectInformation(1)).Returns(new Project()
    {
        Id = 1,
        Name = "BugTrackingSystem",
        Description = "Web site for tracking bugs"


    });
    var result = new ProjectController(null, project.Object).Details(1) as ViewResult;

    Assert.IsNotNull(result);

}

}
```

```
Unit Test Sessions - Details_CheckWhenHaveNotTestCases_ReturnView
Details_CheckWhenHaveNotTestCases_ReturnView    Index
Options    Group by:
1   ✓ 1   0   0   0   All
Detials_CheckWhenIdIsNotNull_ReturnView passed
✓ Bug-Tracking-System.Tests (1 test)
  ✓ Bug_Tracking_System.Tests.Conti
    ✓ ProjectControllerTest (1 test)
      ✓ Detials_CheckWhenIdIsNotNu
Output
```

Figure 32:Show NUnit test for Project Detail When Id is Valid Value

Create Ticket:

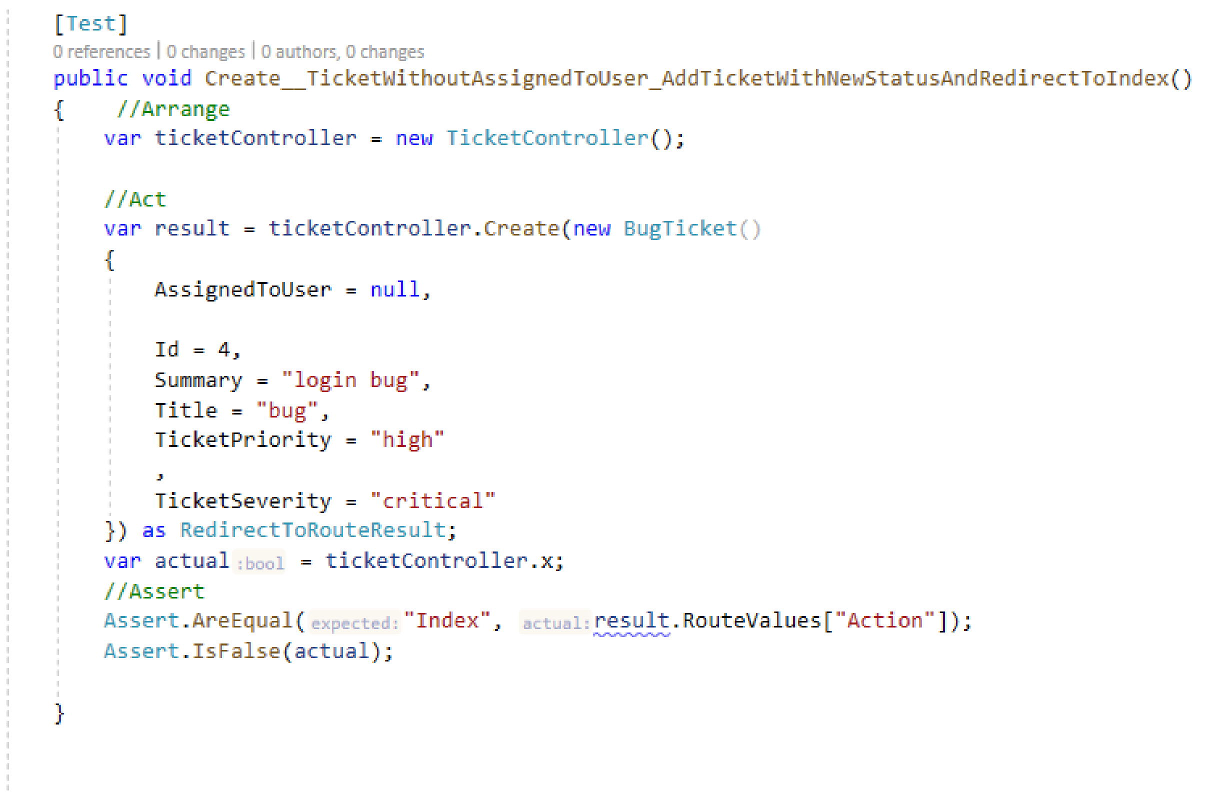
TC#	Bug Ticket: Object	output
1	Valid	Output
2	Invalid	Error
3	No Entered value	Error



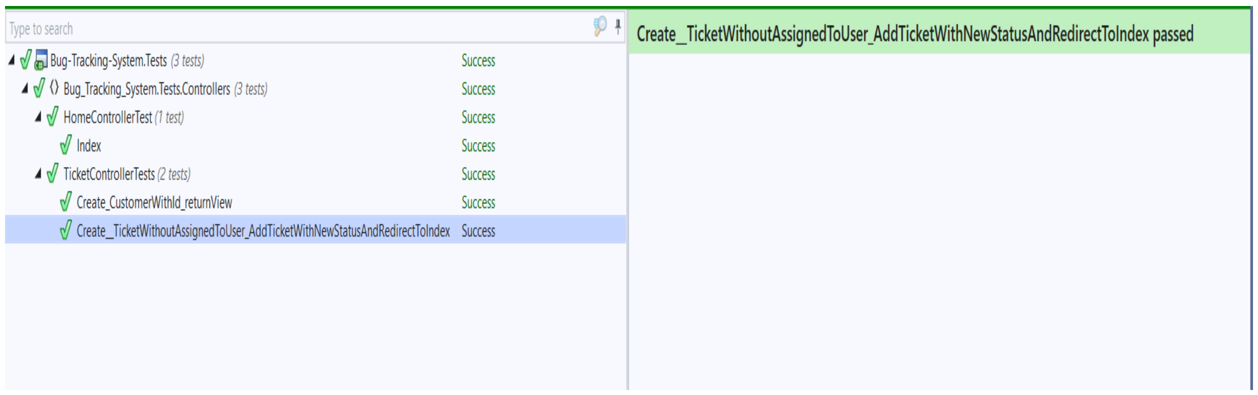
```
[Test]
0 references | 0 changes | 0 authors, 0 changes
public void Create__TicketWithAssignedToUser_AddTicketWithAssignedStatusAndRedirectToIndex()
{    //Arrange
    var ticketController = new TicketController();

    //Act
    var result = ticketController.Create(new BugTicket() { AssignedToUser = null, AssignedToUserId = "123456788", Id = 4,Summary = "login bug",Title = "bug",TicketPriority = "high"
        ,TicketSeverity = "critical"}) as RedirectToRouteResult;
    var actual:bool = ticketController.x;
    //Assert
    Assert.AreEqual(expected:"Index", actual:result.RouteValues["Action"]);
    Assert.IsTrue(actual);

}
```

```
Type to search
◢ ✓ Bug-Tracking-System.Tests (3 tests)                                                   Success
  ◢ ✓ {} Bug_Tracking_System.Tests.Controllers (3 tests)                                  Success
    ◢ ✓ HomeControllerTest (1 test)                                                       Success
        ✓ Index                                                                           Success
    ◢ ✓ TicketControllerTests (2 tests)                                                   Success
        ✓ Create_CustomerWithId_returnView                                                Success
        ✓ Create__TicketWithAssignedToUser_AddTicketWithAssignedStatusAndRedirectToIndex  Success

Create__TicketWithAssignedToUser_AddTicketWithAssignedStatusAndRedirectToIndex passed
```

Figure 34:Show NUnit test for Create Ticket and assigned to developer



```
[Test]
0 references | 0 changes | 0 authors, 0 changes
public void Create__TicketWithoutAssignedToUser_AddTicketWithNewStatusAndRedirectToIndex()
{    //Arrange
    var ticketController = new TicketController();

    //Act
    var result = ticketController.Create(new BugTicket()
    {
        AssignedToUser = null,

        Id = 4,
        Summary = "login bug",
        Title = "bug",
        TicketPriority = "high"
        ,
        TicketSeverity = "critical"
    }) as RedirectToRouteResult;
    var actual :bool = ticketController.x;
    //Assert
    Assert.AreEqual( expected: "Index",  actual: result.RouteValues["Action"]);
    Assert.IsFalse(actual);

}
```

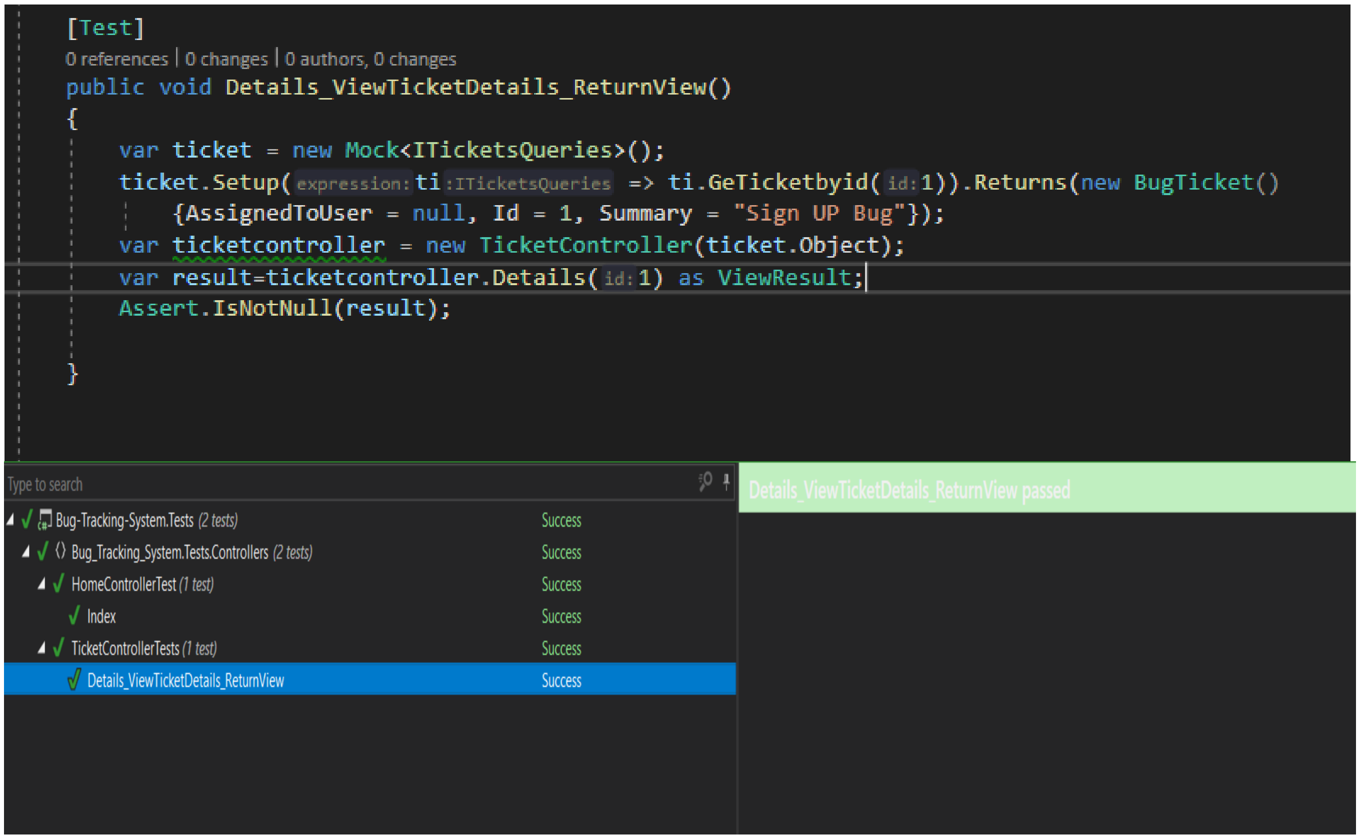


```
Type to search
Bug-Tracking-System.Tests (3 tests)                                         Success
  Bug_Tracking_System.Tests.Controllers (3 tests)                           Success
    HomeControllerTest (1 test)                                             Success
      Index                                                                 Success
    TicketControllerTests (2 tests)                                         Success
      Create_CustomerWithId_returnView                                      Success
      Create_TicketWithoutAssignedToUser_AddTicketWithNewStatusAndRedirectToIndex  Success

Create_TicketWithoutAssignedToUser_AddTicketWithNewStatusAndRedirectToIndex passed
```

Figure 35:Show NUnit test for Create Ticket and Not assigned to developer

Ticket Detail:

TC#	Id	output
1	Valid	Output
2	Invalid	Error
3	No Entered value	Error



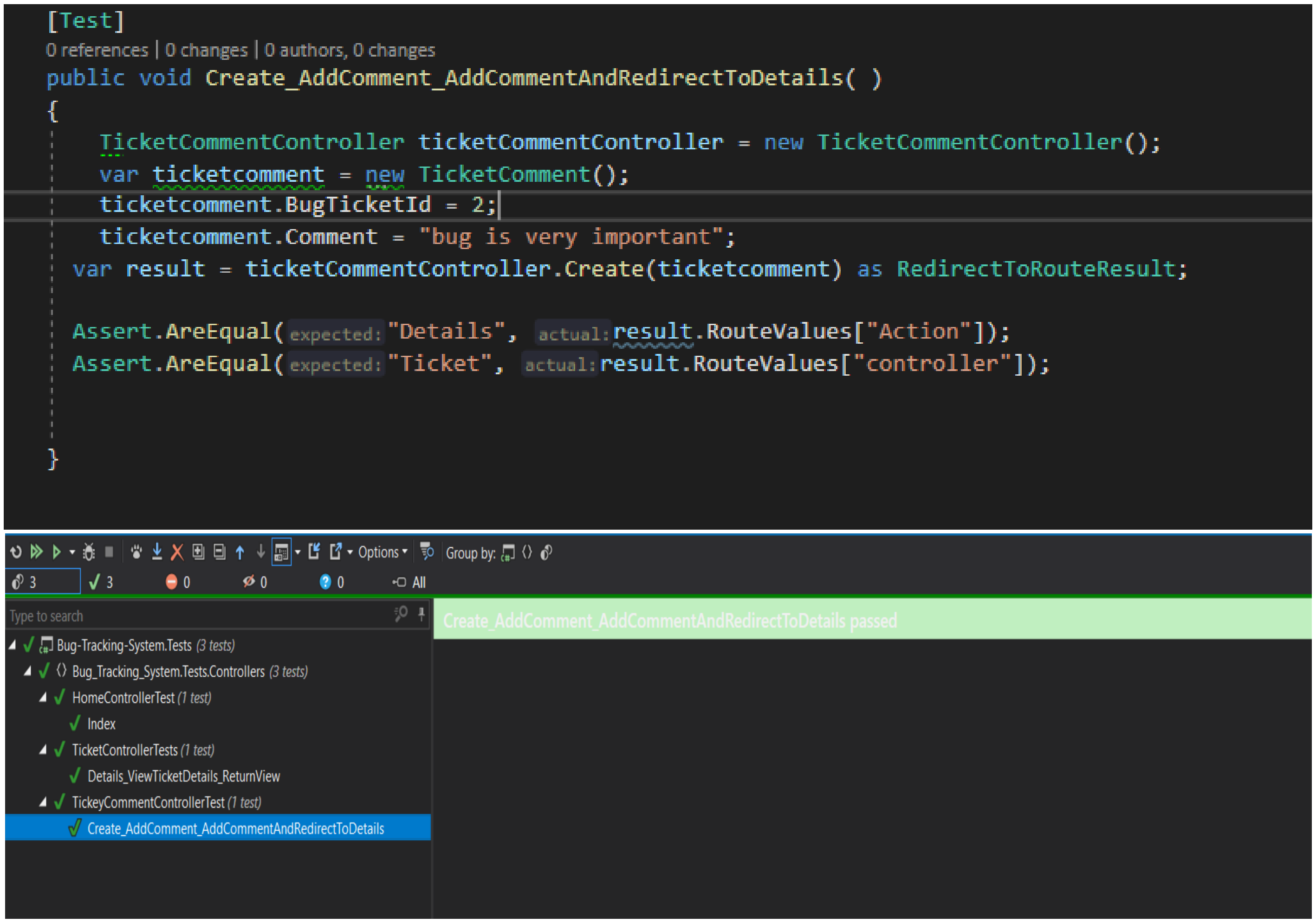
```
[Test]
0 references | 0 changes | 0 authors, 0 changes
public void Details_ViewTicketDetails_ReturnView()
{
    var ticket = new Mock<ITicketsQueries>();
    ticket.Setup(expression:ti:ITicketsQueries => ti.GetTicketbyid(id:1)).Returns(new BugTicket()
        {AssignedToUser = null, Id = 1, Summary = "Sign UP Bug"});
    var ticketcontroller = new TicketController(ticket.Object);
    var result=ticketcontroller.Details(id:1) as ViewResult;
    Assert.IsNotNull(result);

}
```

```
Type to search
Bug-Tracking-System.Tests (2 tests)                          Success
  Bug_Tracking_System.Tests.Controllers (2 tests)            Success
    HomeControllerTest (1 test)                              Success
      Index                                                  Success
    TicketControllerTests (1 test)                           Success
      Details_ViewTicketDetails_ReturnView                   Success

Details_ViewTicketDetails_ReturnView passed
```

Figure 36:Show NUnit test for Ticket Detail

Ticket Comment:

TC#	Ticket Comment: Object	output
1	Valid	Output
2	Invalid	Error
3	No Entered value	Error



```
[Test]
0 references | 0 changes | 0 authors, 0 changes
public void Create_AddComment_AddCommentAndRedirectToDetails( )
{
    TicketCommentController ticketCommentController = new TicketCommentController();
    var ticketcomment = new TicketComment();
    ticketcomment.BugTicketId = 2;
    ticketcomment.Comment = "bug is very important";
  var result = ticketCommentController.Create(ticketcomment) as RedirectToRouteResult;

  Assert.AreEqual(expected: "Details", actual: result.RouteValues["Action"]);
  Assert.AreEqual(expected: "Ticket", actual: result.RouteValues["controller"]);


}
```

```
3   3   0   0   0   All
Type to search
Bug-Tracking-System.Tests (3 tests)
  Bug_Tracking_System.Tests.Controllers (3 tests)
    HomeControllerTest (1 test)
      Index
    TicketControllerTests (1 test)
      Details_ViewTicketDetails_ReturnView
    TickeyCommentControllerTest (1 test)
      Create_AddComment_AddCommentAndRedirectToDetails

Create_AddComment_AddCommentAndRedirectToDetails passed
```

Figure 37:Show NUnit test for Comment

6.2. REQUIREMENTS TESTING:

1. **Register as a user / Login:**

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_01	**Successful Employee Login to The System**	**Valid Email and password**	**1.Enter Email 2.Enter Password 3.Click Login Button**	**Email= testlead1@demo.com Password=123456**	**Login Successfully**	**Login Successfully**
TC_02	Make sure that message will appear to tell you that you should enter a valid email when you enter email not in database.	**Invalid email and password**	**1.Enter Email 2.Enter Password 3.Click Login**	**Email= testlead1111@demo.com Password=123456**	Message will appear to tell you that this email is not valid, there is no such user in the database with this email and return to log in page again	**Massage appeared**
TC_03	Make sure that message will appear to tell you that you should enter a valid password when you enter wrong password	**valid Email and Invalid password**	**1.Enter Email 2.Enter Password 3.Click Login**	**Email= testlead1@demo.com Password=12345666**	**Message will appear to tell you that this password is wrong, and return to log in page**	**Massage appeared**

TC_04	Make a successful sign up.	Valid First Named Valid Last Named valid Email Valid Password	1. Enter First Name 2.Enter Last Name 3.Enter Email 4.Enter password 5. click sign up	First name = Mohamed Last name =Magdy Email=devlead1@demo.com Password=123456	Signed up successfully , home page will open	Signed up successfully , home page open

2. Assign/unassign users to/from roles:

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_05	Make sure that admin can assign role to user	Some users and roles in database Logged as admin	1. select user 2. select Role 3. click submit button		Change Role from N/A to Role That is Assigned	Change Role from N/A to Role That is Assigned

3. **Create projects**

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_06	Make sure that project manager Can create project	Logged as Project manager	1.Enter Project Name 2.Enter Description 3.Enter state 4.Press Create Button		Creating Projects Successfully	Creating Projects Successfully

4. **Assign/unassign users to/from projects:**

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_07	Make sure that project manager Can assign user to project	Logged as Project manager	1.Select project 2. Select users 3.click assign button		Assigning users Successfully	Assigning users Successfully

5. **Create requirement**

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_08	Make sure that project manager Can Create Requirements	Logged as Project manager	1.Enter name 2. Enter Description 3.select dev and test lead 4. click create		Creating Requirements Successfully	Creating Requirement Successfully

6. Assign/unassign users to/from requirements

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_09	Make sure that project manager Can assign users to requirement	Logged as Project manager	1.Select one Dev Lead 2. Select one Test Lead 3. click assign		The user appears in current users in this requirement table	The user appears in current users in this requirement table

7. Create test case

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_10	Make sure that Tester and Test Lead Can Create Test Case	Logged as Tester or Test Lead Assigned to project Assigned to Requirement	1.select project 2.select requirement 3.Enter Title 4.Enter Test Steps 5. Enter Test Data 6.Enter Expected Result 7. Enter preconditions 8.Enter Environment 9. Click Create		Create Test Case Successfully	Success

8. **List test cases by requirements assigned to tester**

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_11	Make sure that Tester and Test Lead Can List Testcases by Requirements	Logged as Tester or Test Lead Assigned to project Assigned to requirement Some testcases in the databases	1.Select project from drop down menu 2. Select Requirement from drop down menu		List Test Cases by requirements Successfully	List Test Cases by requirements Successfully

9. **Sort test case list by title and status**

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_12	Make sure that Tester and Test Lead Can Sort Testcase List by title, status.	Logged as Tester or Test Lead Assigned to project Assigned to requirement Some testcases in the databases	1-Go to index page 2-Press on the label (title or status) in the table		Sort Test Case list by Title and status Successfully	Sort Test Case list by Title and status Successfully

10.Create tickets /Assign tickets to developer

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_13	Make sure that Tester and Test Lead Can Create Tickets	Logged as Tester or Test Lead Assigned to project Assigned to Requirement There is a testcase with fail status	1.Enter Title 2.Enter Summary 3. Select Ticket Priority 4. Select Ticket Severity 5. Select Assigned Developer 6. Click Create		Create Ticket Successfully	Create Ticket Successfully

11.Edit submitted tickets

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_14	Make sure that Tester, Test Lead, developer and dev lead Can edit submitted ticket	Logged as Tester, Developer, Test Lead or Developer lead	1.Edit Title 2. Edit Summary 3. Edit Ticket Priority 4.click Update		Edit Ticket Successfully	Edit Ticket Successfully

12.Create ticket comments

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_15	Make sure that Tester, Test Lead, developer and dev lead Can Create Ticket Comment	Logged as Tester, Developer, Test Lead or Developer lead	1.Add Comment 2. Click ADD		Create Comment Successfully	Create Comment Successfully

13.List comments per ticket

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_16	Make sure that Tester, Test Lead, developer and dev lead Can List Comments Per Ticket	Logged as Tester, Developer, Test Lead or Developer lead	1-Click Ticket Index 2-Click Ticket Details		List Comments per Ticket Successfully	List Comments per Ticket Successfully

14.List history of a ticket's changes

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_17	Make sure that Tester, Test Lead, developer and dev lead Can List history of a ticket's changes	Logged as Tester, Developer, Test Lead or Developer lead	1-Click Ticket Index 2-Click Ticket Details		List history Of a ticket's changes Successfully	List history Of a ticket's changes Successfully

15.List tickets based on the role of the user

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_18	Make sure that Each user sees only his tickets that assigned to him	Logged as Tester, Developer, Test Lead or Developer lead	Go to Index Page		List tickets Based on the role of the user Successfully	List tickets Based on the role of the user Successfully

16. Sort ticket list by title, owner, assignment, creation or recent update date/time, priority, status, and project

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_20	Make sure that Tester, Test Lead, Developer and Dev Lead Can Sort Ticket List by title, owner, assignment, creation or recent update date/time, priority, status, and project	Logged as Tester, Developer, Test Lead or Developer lead	Go to Index Page Press on the label Of attribute want sort by it in the table		List tickets by title, owner, assignment, creation or recent update date/time, priority, status, and project Successfully	List tickets by title, owner, assignment, creation or recent update date/time, priority, status, and project Successfully

17. Full text search of all relevant fields

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_21	Make sure that user can search text of all relevant fields successfully	Logged as any user in the system	1-Go to Index Page 2-Enter Text in search bar		Full text search of all relevant fields Successfully	Full text search of all relevant fields Successfully

18.**Notification function**

Testcase Id	Test Objective	Preconditions	Steps	Test data	Expected Result	Actual Result
TC_22	Make sure that When tester assign ticket to Developer, system will notify him.	Logged as Developer There is ticket assigned to him	1- login to system 2- Press Notify Icon		Notification function works well	Notification function works well

CHAPTER 7: Development tools and technologies

7. CHAPTER 7: DEVELOPMENT TOOLS AND TECHNOLOGIES

7.1. IDEs

Visual Studio Community

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

Figure: 38 Visual Studio

7.2. Database

Microsoft SQL Server

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

Figure: 39 Microsoft SQL Server

7.3. Libraries and Frameworks

7.3.1. Asp.net MVC

ASP.NET MVC is a web application framework developed by Microsoft that implements the model–view–controller (MVC) pattern.
It is open-source software, apart from the ASP.NET Web Forms component, which is proprietary.

Figure: 40 ASP.NET MVC framework

7.3.1.1. Why use it?

- Very popular in the market
- Updated frequently by Microsoft
- Separations of concerns

7.3.2. Entity Framework

Entity Framework (EF) is an open-source object–relational mapping (ORM) framework for ADO.NET. It was originally shipped as an integral part of .NET Framework. Starting with Entity Framework version 6, it has been delivered separately from the .NET Framework.

7.3.2.1. Why use it?

- Code First Migrations
- Maintainability
- Simplification of Queries

Figure: 41 Microsoft Entity Framework

7.3.3. Asp.Net Identity Framework

The ASP.NET Identity system is designed to replace the previous ASP.NET Membership and Simple Membership systems. It includes profile support, OAuth integration, works with OWIN, and is included with the ASP.NET templates shipped with Visual Studio 2013.

7.3.3.1. Why use it?

- Easy to customize
- More Secure

CHAPTER 8: Conclusion and future improvement

8. CHAPTER 8: CONCLUSION AND FUTURE IMPROVEMENT

8.1. Conclusion

In conclusion, we tried look for an elegant and user friendly solution to help in the process of software development to increase the communication between teams and increase their productivity by creating this system, which will greatly help them by having a repository for most of the data related to testing and bug tracking to reduce the human error and friction in communicating these data and information and also help to generate useful information for the project managers to help in their decision making and planning for their projects.

8.2. Future Improvements

We have done quite an effort to be able to achieve our goal, but still there are a lot of room for improvement and enhancements, and we suggest some of them below:

- ❖ Integrate with an automation testing tool
- ❖ Improving the performance more by implementing some components in a more efficient ways.
- ❖ Adding a real time chatting function.

9. REFERENCES

[1] T. Zimmermann, R. Premraj, J. Sillito, S. Breu, “Improving Bug Tracking System”, 31st International Conference On Software Engineering, 2009.

[6] J. Aranda, G. Vendia, “The Secret Life Of Bugs”, In ICSE‟09 Proceedings Of The 31st International Conference On Software Engineering, 2009.

[7] S. Just, R. Premraj, T. Zimmermann, “Towards the Next Generation of Bug Tracking System”, in proceedings Of the IEEE Symposium on Visual languages And Human Centric Computing, 2008.

[8] N. Bettenburg, R. Premraj, T. Zimmermann, S. Kim, “Duplicate Bug Report Considered Harmful…Really? In ICSM‟08 Proceedings of 24th IEEE International Conference On Software Maintenance,2008.

[9] N. Bettenburg, R. Premraj, “Extracting Structural Information from Bug Report”, Proceedings of 5th Working Conference on Mining Software Repositories, 2008.

[11] J. Anvik, L. Hiew G. C. Murphy, “Coping with an Open Bug Repository”, OOPSLA Workshop on Ecllipse Technology Exchange, 2005.

[13] A. Gunes Koro and J. Tian, “Defect Handling In Medium And Large Open Source Projects”, IEEE Software, Vol. 21, 2004.

[15] S. Sen, A. Ganpati, “Proposed Framework for Bug Tracking System in OSS Domain”, International Journal of Advance Research in Computer Science and Engineering, Vol. 3, 2013.