



# **Savitribai Phule Pune University**

**A**

**Project Report**

**On**

**“Test Pass/Fail Criteria and judge the acceptance of application developed”**

Submitted by

Mr. Swapnil Rajendra Take

UNDER THE GUIDANCE BY

**Prof. Pachhade R.C.**



**DEPARTMENT OF COMPUTER ENGINEERING**  
**VISHWABHARATI ACADEMY'S COLLEGE OF ENGINEERING SAROLA BADDI**  
**AHMEDNAGAR 414201**

**ACADEMIC YEAR 2022-23**



**DEPARTMENT OF COMPUTER ENGINEERING  
VISHWABHARATI ACADEMY'S COLLEGE OF ENGINEERING**

**Sarola Baddi, Ahmednagar**

**CERTIFICATE**

This is to certify that **Swapnil Rajendra Take** has successfully completed his Report on **“Test Pass/Fail Criteria and judge the acceptance of application developed”** at Vishwabharti Academy's College of Engineering, Ahmednagar in the partial fulfillment of the Graduate Degree course in B.E. at the Department of Computer Engineering, in the academic Year 2022-2023

Semester-VII as prescribed by the Savitribai Phule Pune University

Prof. Pachhade R.C.

Project Guide

Prof. Joshi S.G.

Head of Department

Prof. Dhongde V.S.

Principal

Date:

Place: Ahmednagar

## **Acknowledgement**

We would like to extend our sincere appreciation and indebtedness to the teacher of the Computer Department Prof. Pachhade R.C. for providing the technical, informative support, valuable guidance and constant inspiration and encouragement as a project guide which has brought this stage one project report in this form.

We would also like to express our gratitude to Prof. Dhongade V.S. for his constant source of encouragement and friendly guidance throughout the project work And at the end we would like to express our gratitude to all staff member who have directly or indirectly contributed in their own way and all my friends Computer Department for their suggestions and constructive criticism.

Mr. Swapnil Rajendra Take

# Table of Contents

1. Unit Testing
2. Integration Testing
3. Extreme Programming & Unit Testing
4. Sample Programs
5. Screenshots
6. Test Plans
7. Test Results

## **Title:**

Create a small web-based application by selecting relevant system environment/platform and programming languages. Narrate concise Test Plan consisting features to be tested and bug taxonomy. Prepare Test Cases inclusive of Test Procedures for identified Test Scenarios. Perform selective Black-box and White-box testing covering Unit and Integration test by using suitable Testing tools. Prepare Test Reports based on Test Pass/Fail Criteria and judge the acceptance of application developed.

## **Problem Definition:**

Perform Desktop Application testing using unittest library in python.

## **Objective**

We are going to learn how to Prepare Test Cases inclusive of Test Procedures for identified TestScenarios.

Perform selective Black-box and White-box testing covering Unit and Integration test by usingsuitable Testing tools.

Prepare Test Reports based on Test Pass/Fail Criteria.

## **Theory**

### **Test Plan for Application Testing**

The Test Plan document is derived from the Product Description, Software Requirement Specification SRS, or Use Case Documents. The focus of the test is what to test, how to test, when to test, and who will test. Test plan document is used as a communication medium betweentest team and test managers.

A standard test plan for Application Testing should define following features;

- Define the scope of testing
- Define objective of testing

- Approach for testing activity
- Schedule for testing
- Bug tracking and reporting

### **UNIT TESTING :**

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

### **INTEGRATION TESTING :**

INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing

### **Extreme Programming & Unit Testing :**

Unit testing in Extreme Programming involves the extensive use of testing frameworks. A unit test framework is used in order to create automated unit tests. Unit testing frameworks are not unique to extreme programming, but they are essential to it. Below we look at some of what extreme programming brings to the world of unit testing:

- Tests are written before the code
- Rely heavily on testing frameworks
- All classes in the applications are tested
- Quick and easy integration is made possible

### **Unittest library python :**

The unittest unit testing framework was originally inspired by JUnit and has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

## Application Tested :

A bookstore application with a simple graphical user interface (**GUI**) is built to support all 4 essential CRUD functionalities. These operations are tested using some test cases.

## Test Plan :

The main aim is to check if we are getting correct output for each given input and if the test is being executed correctly .

- We start with giving varied inputs to perform CRUD operations on database.
- Test cases include authentication for different values for Author, Book name, ISBN code and publishing year fields for our database.
- For this we test our file, a separate tab opens up wherein we write our test code for the different operations, assertEquals is used here.
- Finally we get the test reports, which indicate the overall performance of our tests, i.e. either test pass or test fail.

## SAMPLE UNIT TESTS:

```
def test_nonNumericYear_INSERT(self):
```

```
    flag = database.insert("bookt", "i am auth", "abcd", "IBN000df")
```

```
    self.assertFalse(flag, "Non Numeric year should not be allowed.")
```

```
def test_floatYear_INSERT(self):
```

```
    flag = database.insert("bookt", "i am auth", "1967.45", "IBN000df")
```

```
    self.assertFalse(flag, "Float year should not be allowed.")
```

```
def test_emptyFields_INSERT(self):
```

```
    flag = database.insert("", "", 1920, "")
```

```
self.assertFalse(flag,"Incomplete or blank arguments in insert function are not valid.")
```

```
def test_shortTitle_INSERT(self):
```

```
    flag = database.insert("b","i am auth","1967","IBN000df")
```

```
    self.assertFalse(flag,"Title less than 2 letters should not be allowed.")
```

## **SAMPLE INTEGRATION TESTS:**

```
def test_INSERT(self):
```

```
    flag = self.wrapper.insert(self.title,self.author,self.year,self.isbn)
```

```
    self.assertTrue(flag,"INSERT operation failed.")
```

```
    print("INSERT integration passed.")
```

```
def test_SEARCH(self):
```

```
    flag = self.wrapper.search(self.title,self.author,self.year,self.isbn)
```

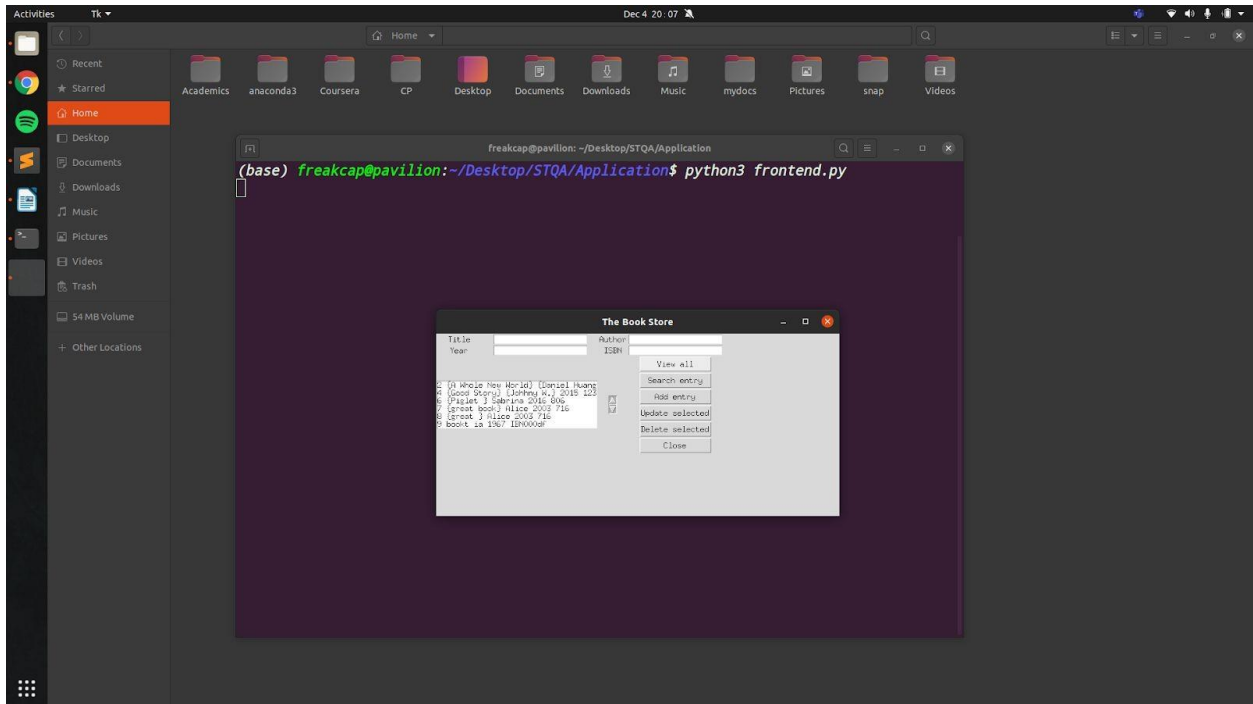
```
    self.assertTrue(flag,"SEARCH operation failed.")
```

```
    print("SEARCH integration passed.")
```

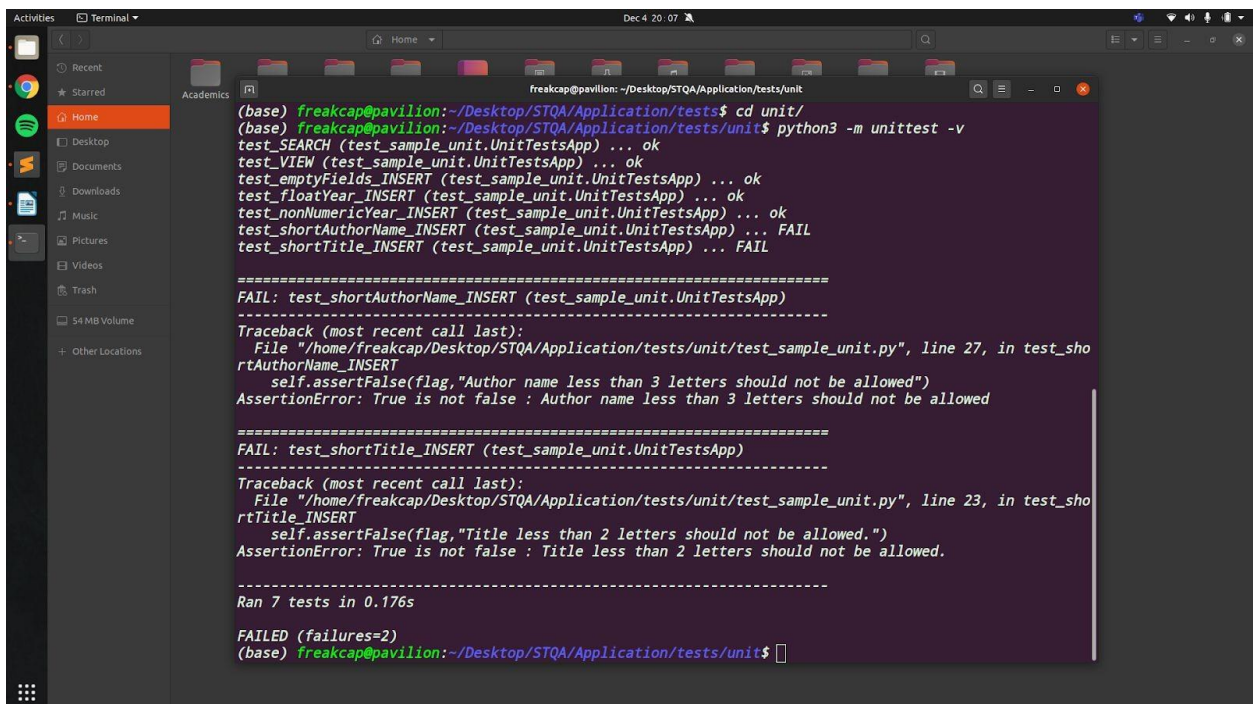
## **Screenshots of application**



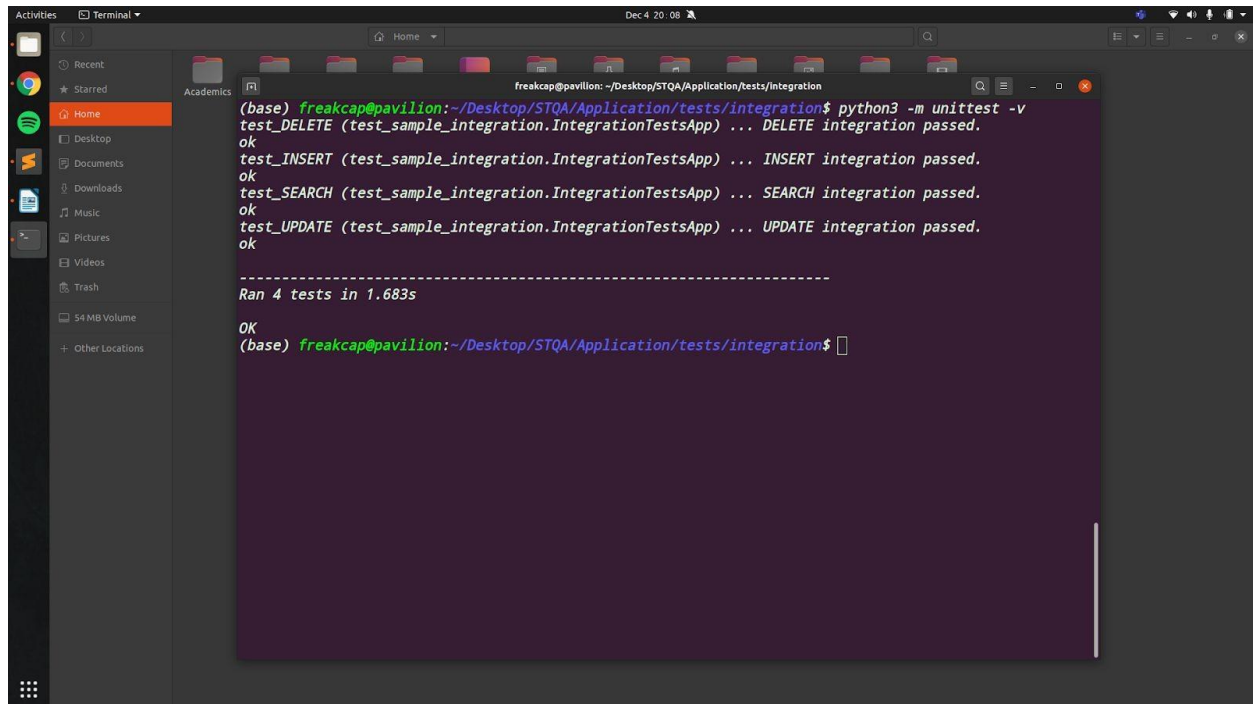
## 1. GUI of the application



## 2. Execution of unit tests



### 3. Execution of integration tests



The screenshot shows a terminal window with a file manager in the background. The terminal output is as follows:

```
(base) freakcap@pavilion:~/Desktop/STQA/Application/tests/integration$ python3 -m unittest -v
test_DELETE (test_sample_integration.IntegrationTestsApp) ... DELETE integration passed.
ok
test_INSERT (test_sample_integration.IntegrationTestsApp) ... INSERT integration passed.
ok
test_SEARCH (test_sample_integration.IntegrationTestsApp) ... SEARCH integration passed.
ok
test_UPDATE (test_sample_integration.IntegrationTestsApp) ... UPDATE integration passed.
ok
-----
Ran 4 tests in 1.683s
OK
(base) freakcap@pavilion:~/Desktop/STQA/Application/tests/integration$
```

## Conclusion :

Hence, we have successfully performed Unit and Integration testing on a Bookstore application performing CRUD operations.

# **Test Plan Report**

## **TABLE OF CONTENTS**

- 1.0 Introduction
- 2.0 Objectives and Tasks
  - 2.1 Objectives
  - 2.2 Tasks
- 3.0 Scope
- 4.0 Testing Strategy
  - 4.1 Unit Testing
  - 4.2 Integration Testing
- 5.0 Hardware Requirements
- 6.0 Software Requirements
- 7.0 Features To Be Tested
- 8.0 Features Not To Be Tested
- 9.0 Deliverables
- 10.0 Tools
- 11.0 Approvals

## 1.0 INTRODUCTION

The Books Management System is a system to record the information of books in a store. The books' information is in the form of following attributes : title, author, year of publication and the ISBN number of the book. Book stores require this kind of systems to keep track of books present in the store and retrieve book information easily. User can insert new books, update existing ones, search for them and delete them as well.

## 2.0 OBJECTIVES AND TASKS

### 2.1 Objectives of Test Plan

Objective of the test plan is to describe scenarios to be tested and covering corner cases of the application as well.

### 2.2 Tasks of Test Plan

Sr. No .	#Business Requirements (BR)	#Functional Requirements (FR)	#Test Scenarios (TS)	#Testing Approaches /Strategies (TA)
01	Operations on existing book Records	Updation, Searching, Deletion	Update book, Search a book, Delete a book, validation on all fields	Functional Testing (Standard Testing) - Unit Testing - Integration Testing
02	Operations to create new book Records	Insertion, View All	Insert a book, View all books, validation on all fields	

### 3.0 SCOPE

#### General

Scope of the Test Plan will be testing all the fundamental validation units and modules , and further test the integration of all the CRUD operations on the application.

### 4.0 TESTING STRATEGY

1. Functional testing
2. Unit testing
3. Integration Testing

Approach	Type of Testing	Manual Testing		Automated Testing on Device	Tools/APIs/Libraries
		Using Device	Using Emulator		
Standard Testing (Functional Testing)	Unit Testing	Yes	No	No	1. Unittest (python unit testing framework)
	Integration Testing	Yes	No	No	

#### 4.1 Unit Testing

MODULE/FUNCTIONALITY NAME:	Insert
UNIT/CLASS:	Database
CREATED BY:	-
DATE OF CREATION:	-
DATE OF REVIEW:	-

TEST CASE ID	TEST CASE	PRECONDITION	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
1	Non numeric year as input	-	insert(title="book",author="author",year="abcd",isbn="isbn0001")	Input should be rejected	Insert return false	Input is rejected	Pass
2	Empty fields in input	-	insert(title="",author="",year="1990",isbn="")	Input should be rejected	Insert returns false	Input is rejected	Pass
3	Floating point value in year field as input	-	insert(title="book",author="author",year="1922.6",isbn="isbn0001")	Input should be rejected	Insert returns false	Input is rejected	Pass
4	Too short title as input	-	insert(title="b",author="author",year="1989",isbn="isbn0001")	Input should be rejected	Insert returns false	Insert return true	Fail
5	Too short author as input	-	insert(title="book",author="a",year="1999",isbn="isbn0001")	Input should be rejected	Insert returns false	Insert returns true	Fail

MODULE/FUNCTIONALITY NAME:	View
UNIT/CLASS:	Database
CREATED BY:	-
DATE OF CREATION:	-
DATE OF REVIEW:	-

TEST CASE ID	TEST CASE	PRECOND ITION	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/ FAIL)
6	View all the records	Atleast one record present in DB	view()	All records should be extracted	Pre condition should hold true	Records are extracte d	Pass

MODULE/FUNCTIONALITY NAME:	Search
UNIT/CLASS:	Database
CREATED BY:	-
DATE OF CREATION:	-
DATE OF REVIEW:	-

TEST CASE ID	TEST CASE	PRECOND ITION	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/ FAIL)
7	No parameter passed to search	-	Search(“”,” ”,””,””)	Query should be rejected	Search returns false	Query is rejected and returns false	Pass

## 4.2 Integration Testing

PROJECT NAME:	Book Management System
MODULE/FUNCTIONALITY:	INSERT
CREATED BY:	-
DATE OF CREATION:	-
DATE OF REVIEW:	-

TEST CASE ID	TEST CASE	PRE-CONDITION	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
8	INSERT a valid book record	-	Title="test", Author="test", year="1990", isbn="ISBN000"	Record is inserted	The record inserted is found using search	Record is inserted	Pass

PROJECT NAME:	Book Management System
MODULE/FUNCTIONALITY:	UPDATE
CREATED BY:	-
DATE OF CREATION:	-
DATE OF REVIEW:	-

TEST CASE ID	TEST CASE	PRE-CONDITION	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
9	UPDATE an existing book record	A valid entry is Present in DB	Title="test", Author="test", year="1990", isbn="ISBN000", updatedisbn = "isbn001"	Record is updated	The existing record is updated	Record is updated	Pass



<b>PROJECT NAME:</b>	Book Management System
<b>MODULE/FUNCTIONALITY:</b>	DELETE
<b>CREATED BY:</b>	-
<b>DATE OF CREATION:</b>	-
<b>DATE OF REVIEW:</b>	-

TEST CASE ID	TEST CASE	PRE-CONDITION	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
10	DELETE an existing book record	A valid entry is Present in DB	Title="test", Author="test", year="1990", isbn="ISBN000"	Record is deleted	The existing record is deleted	Record is deleted	Pass

<b>PROJECT NAME:</b>	Book Management System
<b>MODULE/FUNCTIONALITY:</b>	SEARCH
<b>CREATED BY:</b>	-
<b>DATE OF CREATION:</b>	-
<b>DATE OF REVIEW:</b>	-

TEST CASE ID	TEST CASE	PRE-CONDITION	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
11	SEARCH an existing book record	A valid entry is Present in DB	Title="test", Author="test", year="1990", isbn="ISBN000"	Record is extracted	-	Record is extracted	Pass

## **5.0 HARDWARE REQUIREMENTS**

1. 4GB RAM
2. 100MB HDD
3. i5- 8<sup>th</sup> gen processor

## **6.0 SOFTWARE REQUIREMENTS**

1. Ubuntu 20.04 (focal fossa)
2. Python 3
3. Python tkinter
4. Sqlite DB
5. Python unittest

## **7.0 FEATURES TO BE TESTED**

1. Insertion of records
2. Updation of records
3. Retrieval of records
4. Deletion of records
5. Validation of input

## **8.0 FEATURES NOT TO BE TESTED**

1. GUI of the applications - ( GUI testing is not under the scope of this test plan )

## **9.0 DELIVERABLES**

<b>Deliverable</b>	<b>Date / Milestone</b>
Test Plan	-
Test Result Report	-

## **10.0 TOOLS**

1. Python modules - (unittest)

## **11.0 APPROVALS**

Name (In Capital Letters)	Signature	Date
1.		

# **Test Result Report**

## **TABLE OF CONTENTS**

1.0 Introduction

2.0 Testing Strategy

2.1 Unit Testing

2.2 Integration Testing

3.0 Tools

4.0 Approvals

## 1.0 INTRODUCTION

The Books Management System is a system to record the information of books in a store. The books' information is in the form of following attributes : title, author, year of publication and the ISBN number of the book. Book stores require this kind of systems to keep track of books present in the store and retrieve book information easily. User can insert new books, update existing ones, search for them and delete them as well.

## 2.0 TESTING STRATEGY

1. **Functional testing**
2. **Unit testing**
3. **Integration Testing**

Approach	Type of Testing	Manual Testing		Automated Testing on Device	Tools/APIs/Libraries
		Using Device	Using Emulator		
Standard Testing (Functional Testing)	Unit Testing	No	Yes	No	1. unittest (python unit testing framework)
	Integration Testing	No	Yes	No	

## 2.1 Unit Testing Report

EXECUTION STATUS	COMPLETED/CANCELLED/PENDING
PASSED TESTCASES	7
FAILED TESTCASES	2
PENDING TERSTCASES	0
TEST CASES PLANNED	7

MODULES/ SCENARIOS	DESCRIPTION	% TCs EXECUTED	% TCs PASSED	TCs PENDING	PRIORITY	REMARKS
INSERT	Validation units related to insertion	63	37	00	MODERATE	-
VIEW	Retrieval of records from DB	100	100	00	LOW	-
SEARCH	Searching records in DB	100	100	00	MODERARTE	-

## 2.2 Integration Testing

EXECUTION STATUS	COMPLETED/CANCELLED/PENDING
PASSED TESTCASES	4
FAILED TESTCASES	0
PENDING TERSTCASES	0
TEST CASES PLANNED	4

MODULES/ SCENARIOS	DESCRIPTION	% TCs EXECUTED	% TCs PASSED	TCs PENDING	PRIORITY	REMARKS
INSERT	Insertion of records	100	100	00	HIGH	-
UPDATE	Updation of records	100	100	00	HIGH	-
DELETE	Deletion of records	100	100	00	HIGH	-
SEARCH	Searching of records in DB	100	100	00	HIGH	-

### 3.0 TOOLS

1. Python module - unittest

### 4.0 APPROVALS

Name (In Capital Letters)      Signature      Date

1.