

# Travel in Good Health Management

---

*Software Test Plan*

---

By

S V Dilip Kumar(14CS30030)

M Tharun (14CS10056)

Software Engineering Assignment

## TABLE OF CONTENTS

INTRODUCTION.....	2
Objectives.....	2
Testing Strategy.....	2
Scope.....	3
Purpose.....	3
TESTS.....	3
Unit Testing.....	4
Integration Testing.....	4
Interface Testing.....	4
Security Testing.....	4
Performance testing.....	4
Use Case Testing.....	4
PASS/FAIL CRITERIA.....	5
Suspension Criteria.....	5
Approval Criteria.....	5
ENVIRONMENTAL REQUIREMENTS.....	5
Hardware.....	5
Software.....	6
BUGS REPORTED.....	6
Functional Testing.....	6
Agent Test.....	6
Customer Test.....	7
MAnagement test.....	7

## INTRODUCTION

The Software Test Plan (STP) is designed to prescribe the scope, approach, resources, and schedule of all testing activities. The plan must identify the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan. This Software Testing Plan provides a complete description of testing of all the functions and specifications of the Travel in Good Health Management System (TGHM) developed for providing a bridge between the food suppliers and the customers during a train journey.

## OBJECTIVES

The software test plan is aimed at verifying the functionality and correct working of every aspect and part of the software. The software should not cease to work correctly when it has been exposed to stress and all corner cases should be handled effectively. Testing software is an important part of the development life cycle of a software. It is an expensive activity. Hence, appropriate testing methods are necessary for ensuring the reliability of a software. According to the ANSI/IEEE 1059 standard, the definition of testing is the process of analyzing a software item, to detect the differences between existing and required conditions i.e. defects/errors/bugs and to evaluate the features of the software item.

## TESTING STRATEGY

Testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item.

Specific test plan components include:

- Purpose for this level of test
- Management and technical approach
- Pass / Fail criteria
- Hardware/ software requirements
- Reported bugs
- Functional Testing

## SCOPE

Testing will be performed at several points in the life cycle as the product is constructed. Testing is a very 'dependent' activity. As a result, test planning is a continuing activity performed throughout the system development life cycle. Test plans must be developed for each level of product testing.

## PURPOSE

The purpose of testing is to verify and validate a software and to find the defects present in the software. The purpose of finding those problems is to get them fixed.

- Verification:

It is the checking or the testing of software for consistency and conformance by evaluating the results against pre-specified requirements.

- Validation:

It looks at the correctness of the system, i.e. the process of checking the fact that what has been specified is what the user wanted.

- Defect:

It is a variance between the expected and actual result. The defect's ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

## TESTS

- Features to be tested:

All the use cases are thoroughly tested by white box testing and black box testing. The databases are checked after every use case and if they do not match with the expected outcome, they are reported to be failed.

- Features not to be tested:

- JDBC MySQL connector.

## UNIT TESTING

Each unit has been tested for consistency by giving input manually through the GUI and the corresponding screenshots have been included in the Test Suite Results document.

## INTEGRATION TESTING

All of the above units has been integrated together and tested for consistency. The dataflow from one unit to another has been verified.

## INTERFACE TESTING

The software has been tested on machines with different configurations (for example computers with different operating systems like Windows or Linux) and the functionality of the interface has been verified.

## SECURITY TESTING

Password validation system has been implemented for the management.

## PERFORMANCE TESTING

Each of the use cases are tested with respect to their response times and are implemented in such a way that the response time is as small as possible. To achieve this number of SQL queries that are made while carrying out a particular use case is at most two. It has been made sure that less number of queries has been deployed for the use case, but there is a need to use more queries when dealing with placement of order, verification /authenticity of the food suppliers and agents where data from multiple table are required for the same use. In these cases the query could reach a maximum of 3 queries.

## USE CASE TESTING

The following use cases were checked to cover all Corner Cases as far as possible:

- If the customer registers with an existing user name a message box is displayed appropriately.

- If passwords and usernames do not match appropriate labels are displayed.
- If there are no available food suppliers a blank table and combo box is displayed for the customer.
- If the customer logs in after being rejected a particular apology is displayed along with future course of action.
- If Sno and inappropriate entries are entered in textfields a suitable message box is displayed.
- If an action has been successfully performed then success message is displayed else a connectivity problem message box would be displayed.

### PASS/FAIL CRITERIA

### SUSPENSION CRITERIA

The test is considered suspended if any of the following is encountered:

- The Software crashes.
- It results in wrong output.

### APPROVAL CRITERIA

The test is considered accepted if any of the following is encountered:

- The correct results are obtained
- The software does not take much time to achieve the aforementioned criteria.

### ENVIRONMENTAL REQUIREMENTS

### HARDWARE

- Working network connection
- MySQL should be installed on an appropriate server so that it can handle simultaneous request from various users.(at port : 3306)

## SOFTWARE

- Operating System: Windows/Linux
- Java should be installed (preferably JDK 7)
- JDBC libraries should be included
- MySQL should be installed








## BUGS REPORTED

- Unnecessary buttons were discovered in the Management pane of which were removed during the testing phase
- Auto close of frames were not performed and they were implemented
- Bugs regarding the data base were discovered. Such as the default value of a customer column was kept as NULL which created problems during the running of the program which was later changed to NONE.

## FUNCTIONAL TESTING

Suitable test classes are included in file along with the test plan and test suite for understanding the functionalities of the different classes

## AGENT TEST

-  testStoreInDb()-  
Checks whether an object created could be stored in the database.
-  testGetContact()
-  testGetName()
-  testGetAgentFoodSupplier()
-  testGetTotalDeliveries()
-  testSetAgentNumberOfOrdersAccepted()
-  testGetSuccesfullDeliveries()

All the above get set methods check data retrieval and data injection into the object of the type Agent.

## CUSTOMER TEST

- ✚ testGetName()
- ✚ testSetName()

## MANAGEMENT TEST

- ✚ testGetMenu()
- ✚ testGetOrders()
- ✚ testGetStations\_AgentAuth()
- ✚ testSetMenu()
- ✚ testGetAgents()
- ✚ testAddStation()
- ✚ testRemoveStation()
- ✚ testGetStations\_ManagementAuth()
- ✚ testPlaceOrder()
- ✚ testGetTrains()
- ✚ testAddTrain()
- ✚ testRemoveTrain()
- ✚ testStartTimer() –
  - Timer which keeps of 15 min window in which the agent can accept the order.
- ✚ testModifyOrder()
- ✚ testCancel()
- ✚ testSendRegistrationRequest()
- ✚ testGetRoutes()
- ✚ testAcceptRegistration()
- ✚ testGetStatistics()
- ✚ testGetRegistrants()
- ✚ testAcceptOrder()
- ✚ testRejectOrder()
- ✚ testRejectRegistration()
- ✚ testAddRoute()

And the other test classes which verify the functionality of the smaller classes with less than four functions have been included in the file.