Table of Contents

[VERSION HISTORY Error! Bookmark not defined.](#_Toc523241978)

[1. INTRODUCTION 4](#_Toc523241979)

[1.1 Objectives 4](#_Toc523241980)

[1.2 Scope 4](#_Toc523241981)

[1.2.1 In Scope 4](#_Toc523241982)

[1.2.2 Out of Scope 4](#_Toc523241983)

[1.3 Roles and Responsibilities 5](#_Toc523241984)

[1.3.1 Developer 5](#_Toc523241985)

[1.3.2 Supervisor 5](#_Toc523241986)

[1.3.3 Coordinator 5](#_Toc523241987)

[1.4 Assumptions for Test Execution 5](#_Toc523241988)

[1.5 Constraints for Test Execution 6](#_Toc523241989)

[2. Test Methodology 6](#_Toc523241990)

[2.1 Purpose of TEST 6](#_Toc523241991)

[2.2 Overview 7](#_Toc523241992)

[2.3 Testing Process 7](#_Toc523241993)

[3. TEST STRATEGY 8](#_Toc523241994)

[3.1 Test Objectives 8](#_Toc523241995)

[3.2 Test Assumptions 8](#_Toc523241996)

[3.3 Test Principles 9](#_Toc523241997)

[3.4 Data Approach 9](#_Toc523241998)

[3.5 Scope and Levels of Testing 9](#_Toc523241999)

[3.5.1 Exploratory 9](#_Toc523242000)

[3.5.2 Functional Test 9](#_Toc523242001)

[3.5.3 User Acceptance Test (UAT) 10](#_Toc523242002)

[4. EXECUTION STRATEGY 11](#_Toc523242003)

[4.1 Test Cycles 11](#_Toc523242004)

[4.2 Test Metrics 11](#_Toc523242005)

[4.3 Defect tracking & Reporting 12](#_Toc523242006)

[5. TEST MANAGEMENT PROCESS 13](#_Toc523242007)

[5.1 Test Design Process 13](#_Toc523242008)

[5.2 Test Execution Process 13](#_Toc523242009)

[5.3 Role Expectations 13](#_Toc523242010)

[5.3.1 Project Management 14](#_Toc523242011)

[5.3.2 Test Team-Users 14](#_Toc523242012)

[5.3.3 Test-Lead 14](#_Toc523242013)

[5.4 Item Pass/Fail Criteria 15](#_Toc523242014)

[5.5 Suspension criteria: 15](#_Toc523242015)

[6. Deliverables Matrix 15](#_Toc523242016)

[7. Test Environment 16](#_Toc523242017)

[7.1 Field Comments 16](#_Toc523242018)

[7.2 Hardware 17](#_Toc523242019)

[7.3 Software 17](#_Toc523242020)

[7.4 Bug Severity and Priority Definition 17](#_Toc523242021)

[7.5 Severity List 17](#_Toc523242022)

[7.6 Priority List 18](#_Toc523242023)

[7.7 Bug Reporting 18](#_Toc523242024)

[8. Schedule 18](#_Toc523242025)

[8.1 Schedule 18](#_Toc523242026)

[9. Terms/Acronyms 19](#_Toc523242027)

[10. Approval 19](#_Toc523242028)

[10.1 Approval 19](#_Toc523242029)

[11. References 19](#_Toc523242030)

# INTRODUCTION

This is the test plan on Student Automated Attendance System(SAAS)**.** SAAS is a desktop based application which tends to solve a very problem of tedious manual handling of attendance process, documents and reports generation. This application reduces the hustle by just allowing a user to take or access attendance details.

Test plan will include unit testing (white-box testing, black-box testing); Integration testing, system testing, unit testing etc. to ensure that the system works as in the required program specifications. The decision as to which technique(s) to use for any given unit of code will reside with the developer.

## Objectives

* Identify existing project information
* Identify the approach that should be followed.
* Identify the features that should be tested.
* List the recommended test requirements.
* Recommend and describe the testing strategies to be employed.
* Identify the required resources and provide an estimation of the test efforts.
* Fix the schedule of intended testing activities.
* Identify the risks associated with the test strategy.
* List the deliverable elements of the test activities.

## Scope

Description of the current test plan scope based on role given below.

### In Scope

The Student Automated Attendance System test plan defines the unit, integration, system, regression, and Client Acceptance testing approaches. The test scope includes:

* Testing of all functional, application performance, security and use cases requirements listed in the SRS document.
* Quality requirements and fit metrics of Student Automated Attendance System.
* End-to-end testing and testing of interfaces of all systems that interact with the system.

### Out of Scope

The following are considered out of scope for Student Automated Attendance System Test Plan and testing scope:

* Functional requirements testing for systems outside the system and concerned fingerprint hardware.
* Testing of business concessions, disaster recovery and business continuity plan.

## Roles and Responsibilities

Roles and responsibilities may differ based on the actual time. Below listed functions are for testing phase.

### Developer

A fourth year student wishing to undertake software or solution development activities for fulfillment of COMP 493 Software Project. Responsible to:

(a) Develop the system/application

(b) Develop Use cases and requirements in collaboration with the Adopters

(c) Conduct Unit, system, regression and integration testing

(d) Support user acceptance testing

### Supervisor

A Lecturer selected to oversee formal adoption, testing, validation, and application of products or solutions developed by developer. Responsible to:

(a) Contribute to requirement development.

(b) Contribute to develop and execution of the development test scripts.

(c) Conduct Full User Acceptance, regression, and end-to-end testing; this includes identifying testing scenarios, building the test scripts, executing scripts and reporting test results

### Coordinator

Responsible for coordinating the entire testing process, workflow and quality management with activities and responsibilities to:

(a) Monitor and manage testing integrity and support testing activities.

(b) Coordinate activities across the students.

Add more as appropriate to testing scope

## Assumptions for Test Execution

Below are some minimum assumptions that has be completed with some examples.

User Acceptance testing will be conducted by end-users which will be lecturers, students and the admin.

Test results will be reported on daily basis using bug report mechanism. Failed scripts and defect list from bug report with evidence will be sent to developer directly.

Use cases have been developed by adopters for user acceptance testing. Use cases are approved by test lead.

Test scripts are developed and approved.

Test actors will support and provide appropriate guidance to adopters and developer to conduct testing.

Major dependencies should be reported immediately after the testing kickoff.

## Constraints for Test Execution

Below are some minimum assumptions followed by example constraints.

Adopters should clearly understand on test procedures and recording a defect or enhancement. Testing Process Manager (Developer) will schedule a meeting with supervisor to train and address any testing related issues.

Developer will receive consolidated list of request for test environment set up, user accounts set up, data set (actual and mock data), defect list, etc. through notification after the initial adopter testing kick off.

Developer will support ongoing testing activities based on priorities.

Test scripts must be approved by adopter’s prior test execution.

Test scripts, test environment and dependencies should be addressed during testing kickoff in presence of developer and request list should be listed down.

The Developer cannot execute the User Acceptance and End to End test scripts. After debugging, the developer can conduct their internal test, but no results from that test can be recorded / reported.

Adopters are responsible to identify dependencies between test scripts and submit clear request to set up test environment to the developer.

# Test Methodology

## Purpose of TEST

The Test Plan document documents and tracks the necessary information required to effectively define the approach to be used in the testing of the Student Automated Attendance System project. The Test Plan document was created by the developer of the project. Its intended audience is the project developer, project supervisor, testing users and other concerned parties. Some portions of this document may on occasion be shared with the client/user and other stakeholder whose input/approval into the testing process is needed.Description of the appropriate strategies, process, workflows and methodologies used to plan, organize, execute and manage testing of the software project are also provided.

## Overview

The below list is not intended to limit the extent of the test plan and can be changed as the revision progresses.

The general purpose of the Test Plan is to achieve the following:

* Define testing strategies for each area and sub-area to include all the functional and quality (non-functional) requirements.
* Divide Design specification into testable areas and sub-areas.
* Define bug-tracking procedures.
* Identify testing risks.
* Identify required resources and related information.
* Provide testing Schedule.

## Testing Process

*Figure 1. Test Process approach that will be followed.*

* **Organize Project**: It involves creating a System Test Plan, Schedule & Test approach, and assigning responsibilities.
* **Design System Test**: It involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions, expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.
* **Design Test Procedure**: It includes setting up procedures such as Error Management systems and Status reporting.
* **Build Test Environment**: It includes requesting, building hardware, software and data set-ups.
* **Execute System Tests:** The tests identified in the Design Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary
* **Signoff**: Signoff happens when all pre-defined exit criteria have been achieved

# TEST STRATEGY

### Test Objectives

The objective of the test strategy is to verify that the functionality of the Student Automated Attendance System works according to the specifications provided in the SRS.

The final product of the test is a ready-to use software;

### Test Assumptions

**Key Assumptions**

* Production-like data is required and is available in the system prior to start of Functional Testing

**General**

* Exploratory Testing would be carried out once the build is ready for testing.
* Test case design activities will be performed by the testing designers.
* Test environment and preparation activities will be owned by the developer
* The project Supervisor will review and sign-off all test deliverables
* The developer will manage the testing effort with close coordination with the supervisor and the system users.
* The users of the system have the knowledge of the system prior to testing.
* The system will be treated as a black box. If the Student Automated Attendance System enhances the login capabilities, it will be assumed that the database is working properly.

**Functional Testing**

* During Functional testing, testing team will use preloaded data which is available on the system at the time of execution.
* The Test Team will perform Functional testing on all modules in the Student Attendance system.

### Test Principles

* Testing will be focused on meeting the Student Attendance system objectives outlined in the proposal document - an automated system.
* Testing processes will be well defined, yet flexible, with the ability to implement any changes needed.
* Testing environment and data will emulate a production environment as much as possible.
* Testing will be a repeatable, quantifiable, and measurable activity.
* Testing process for the different user portals will have clearly defined objectives and goals.

### Data Approach

* In functional testing, the portals will contain pre-loaded test data and which is used for testing activities.

### Scope and Levels of Testing

### Exploratory

**PURPOSE**: the purpose of this test is to make sure critical defects are removed before the System is released for use.

**SCOPE**: All modules- Admin, Lecturer and Student modules.

**TESTERS**: Testing team who the users of the system.

**METHOD**: this exploratory testing is carried out in the application without a system documentation to guide the test team.

### Functional Test

**PURPOSE:**  Functional testing will be performed to check the functions of application. The functional testing is carried out by feeding the input and validates the output from the application.

Figure showing testing scenarios.

|  |  |  |  |
| --- | --- | --- | --- |
| User | Scenarios | Complexity | No. of Test Cases |
| Admin | Login  Manage Students  Manage Lecturers  Manage Reports | Easy  Medium  Medium  Hard | 2  4  5  2 |
| Lecturer | Login  Manage Class  Take Attendance  Manage Reports | Easy  Hard  Medium  Medium | 4  4  6  2 |
| Student | Login  View Attendance  Manage Profile  Chat | Easy  Easy  Medium  Hard | 1  4  5  3 |

*Figure 2. Test scenarios by various users.*

**TESTERS**: Testing Team which includes the Users of the system.

**TEST ACCEPTANCE CRITERIA**

1. Approved Software Specification document, Software Design Document with Use case documents must be available prior to start of Test design phase.
2. Test cases approved and signed-off prior to start of Test execution
3. Development completed, unit tested with pass status.
4. Test environment with application installed, configured and ready to use state

**TEST DELIVERABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Deliverable Name** | **Author** | **Reviewer** |
| 1. | Test Plan | Developer | Project Supervisor |
| 2. | Functional Test Cases | Selected Users | System Developer |
|  |  |  |  |

*Figure 3: Testing Deliverables*

### User Acceptance Test (UAT)

**PURPOSE**: This test focuses on validating the business logic of the Student Attendance system. It will allow the end users to complete final review of the system prior to deployment.

**TESTERS**: the UAT is performed by the end users (Admin, Lecturer and Student).

**METHOD**: Since the end users are the most indicated to provide inputs to the system and how the system adapts to them, it may happen that the users do some validation not contained in this Test Plan and write the test cases down. The System developer will review the UAT test cases based on the inputs from End user (Admin, Lecturer and Student).

**TIMING**: Only after this test is completed the product can be released to production.

**TEST DELIVERABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Deliverable Name** | **Author** | **Reviewer** |
| 1. | UAT Test Cases | End Users | The System developer |

*Figure 4: UAT Deliverables.*

# EXECUTION STRATEGY

### Test Cycles

* + There will be 1 cycle for login functionality testing and 5 cycles of testing for all other scenarios. Each cycle will execute all the modules (Shown in figure 1).
  + The objective of the first cycle is to identify any blocking, critical defects, and most of the high defects. It is expected to use some work-around in order to get to all the scripts.
* UAT test will consist of one cycle.

### Test Metrics

The below are some of the metrics to measure the progress and level of success of the test.

|  |  |
| --- | --- |
| **Report** | **Description** |
| Test preparation & Execution Status | Pass or Fail |
| Daily execution  status | To report on Pass, Fail and system defects. |
| Test Process Status report | Reporting |

*Figure 5: Test Metrics.*

### Defect tracking & Reporting

The user testing the system makes a report of the defects, hands them to the Test Lead who then contacts the system developer to fix them. After the defects are fixed, the person testing re-tests the system again and if it is finally corrected the Defect Tracking phase is complete. Otherwise the system Developer is contacted again to fix the defects.

Following flowchart depicts Defect Tracking Process:

**Approved?**

**Start**

**Tester:**

**Report defects**

**Developer:**

**Fixes defects**

**Tester:**

**Retests the product**

**No**

**Stop**

**Close defect**

**Yes**

**Test Lead**

**Validate defects**

*Figure 6: Test Flowchart.*

# TEST MANAGEMENT PROCESS

### Test Design Process

Figure 7: Test Design Process.

• The tester will understand each requirement written in the SRS-System Specification Document and prepare corresponding test case to ensure all requirements are covered.

• Each of the Test cases will undergo review by the System Developer and the review defects are captured and shared to the Test Team-Users.

• During the preparation phase, tester will use the application, use case and functional specification to write step by step test cases.

• Any subsequent changes to the test case if any will be directly update.

### Test Execution Process

Figure 8: Test Execution Process.

As per Process, final sign-off or project completion process will be followed

### Role Expectations

The following list defines in general terms the expectations related to the roles directly involved in the management, planning or execution of the test for the project.

| SN0. | Roles | Name | Contact Info |
| --- | --- | --- | --- |
| 1. | Project Supervisor |
| 2. | Test Lead |
| 3. | Testing Team-Users |
| 4. | System Developer |

Figure 9: Contacts.

### Project Management

• Project Supervisor: reviews the content of the Test Plan, Test Strategy and Test Estimates signs off on it.

### Test Team-Users

• Develop test conditions, test cases, expected results, and execution scripts.

• Perform execution and validation.

• Identify, document and prioritize defects according to the guidance provided by the Test lead.

• Re-test after software modifications have been made according to the schedule.

### Test-Lead

• Acknowledge the completion of a particular module testing process.

• Give the OK to start next level of testing.

• Facilitate defect communications between testing team and the system developer.

• System Developer reviews testing deliverables (test plan, cases, scripts, expected results, etc.) and provide timely feedback.

• Assist in the validation of results (if requested).

• Support the development and testing processes being used to support the project.

• Certify correct components have been delivered to the test environment at the points specified in the testing schedule.

• Define processes/tools to facilitate the initial and ongoing migration of components.

• Implement fixes to defects according to schedule.

### Item Pass/Fail Criteria

The pass/fail criteria are given as below. They have been set with the help of software specification document.

1. If a user fails to login, then that scenario will be considered a fail case.
2. If a user with an account cannot edit his/her account, then that is a fail case.
3. System crash will be considered as fail case.
4. If user clicks on the searches for project and the available project displayed are not what she/she is looking for then that will be a fail state.
5. When the system takes more 5minutes to load search then that will be fail case.
6. If a user clicks to upload a file to the system and the system fails, then that will be a fail case.
7. When user can not generate or view report then that will a fail case.

### Suspension criteria:

The test will be suspended if any of these events occur

1. User cannot access the system.
2. Execution on database cannot be completed.
3. User cannot login.
4. An operating system cannot support the system.
5. The system crashes.

included database and Load testing.

# Deliverables Matrix

Below is the list of artifacts that are process driven and were produced during the testing lifecycle. Certain deliverables should be delivered as part of test validation.

This matrix should be updated routinely.

|  |
| --- |
| **Deliverable** |
| **Documents** |
| Test Approach |
| 🡪 Test Plan |
| 🡪 Test Schedule |
| 🡪 Test Specifications |
| **Test Case / Bug Write-Ups** |
| Test Cases / Results |
| Test Coverage Reports |
| **Reports** |
| Test results report |
| Test Final Report - Sign-Off |

All **High priority** defects should be addressed within 1 day of the request and resolved/closed within 2 days of the initial request

All **Medium priority** defects should be addressed within 2 days of the request and resolved/closed within 4 days of the initial request

All **Low priority** defects should be resolved/closed no later than 5 days of the initial request.

# Test Environment

### Field Comments

Hardware – The testing environment hardware was laptop PC.

Product – Product was Student Automated Attendance System.

Operating System – Windows 2010 pro.

Version - Application version 001.

Severity – The application was open to any decrier.

Resolution - Only developer will update based on the defect.

Module - For an application with multiple modules, select appropriate module for the defect reported.

URL – [Developer@gmail.com](mailto:Developer@gmail.com).

Assigned to - To be updated by Developer.

Priority – High.

Summary - Summary of the defect, bug or issue

Detailed Description. - Details of the defect, bug or issue.

Submit - Submit the bug reporter.

### Hardware

Hardware requirements to test the application was identified as a constraint. The only hardware used was a normal laptop with 4GB RAM and 500GB ROM although it should be at least 2GB RAM and 250GB ROM. Testing was also done using a desktop to ensure accuracy was effected. Testing also have had access control range from the minimum to the recommended client hardware configurations listed in the project’s Software Requirements Specification and Design Specification documents.

### Software

In addition to the application and any other customer specified software, the following list of software should be considered a minimum:

* Windows 2010.
* XAMPP Database.
* NetBeans IDE.

### Bug Severity and Priority Definition

Bug Severity and Priority fields are both very important for categorizing bugs and prioritizing if and when the bugs will be fixed. The bug Severity and Priority levels will be defined as outlined in the following tables below. Testing will assign a severity level to all bugs. The Test Lead will be responsible to see that a correct severity level is assigned to each bug.

### Severity List

The tester entering a bug into SAAS is also responsible for entering the bug Severity.

|  |  |  |
| --- | --- | --- |
| **Severity ID** | **Severity Level** | **Severity Description** |
| 1 | Critical | The module/product crashes or the bug causes non-recoverable conditions. System crashes, database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Severe 1 bug. |
| 2 | High | Major system component unusable due to failure or incorrect functionality. Severity 2 bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Severe 2 bugs can have a work around, but the work around is inconvenient or difficult. |
| 3 | Medium | Incorrect functionality of component or process. There is a simple work around for the bug if it is Severity 3. |
| 4 | Minor | Documentation errors or signed off Severity 3 bugs. |

### Priority List

|  |  |  |
| --- | --- | --- |
| **Priority ID** | **Priority Level** | **Priority Description** |
| 5 | Must Fix | This bug must be fixed immediately; the product cannot ship with this bug. |
| 4 | Should Fix | These are important problems that should be fixed as soon as possible. It would be an embarrassment to the developer if this bug shipped. |
| 3 | Fix When Have Time | The problem should be fixed within the time available. If the bug does not delay system process, then fix it. |
| 2 | Low Priority | It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed. |
| 1 | Trivial | Enhancements/ Good to have features incorporated- just are out of the current scope. |

### Bug Reporting

Developer recognizes that the bug reporting process is a critical communication tool within the testing process. Without effective communication of bug information and other issues, the development and release process will be negatively impacted.

The Test will be responsible for managing the bug reporting process. Testing’s standard bug reporting tools and processes will be used. The best to mention are the java option panes and development will enter their data into the error database following the field entry definitions.

# Schedule

### Schedule

The schedule for testing the system will be as follows

|  |  |  |
| --- | --- | --- |
| **TEST PHASE** | **RESPONSIBLE PERSON** | **TIME(weeks)** |
| Unit Testing | The Project Developer | 2 |
| Component Testing | The Project Developer | 3 |
| Integration Testing | The Project Supervisor | 1 |
| Use Case Validation | The Project Developer | 2 |
| User Interface Testing | The Project Supervisor | 2 |
| Load Testing | The Project Developer | 1 |
| Performance Testing | The Project Developer | 1 |
| Release to Production | Senate Member | 1 |

# Terms/Acronyms

The following terms were used.

| **TERM/ACRONYM** | **DEFINITION** |
| --- | --- |
| SAAS | Student Attendance Management System |
| API | Application Program Interface |
| CAT | Client Acceptance Testing |
| End-to End Testing | Tests user scenarios and various path conditions by verifying that the system runs and performs tasks accurately with the same set of data from beginning to end, as intended. |
| N/A | Not Applicable |
| QA | Quality Assurance |
| TBD | To Be Determined |

# Approval

### Approval

The test will be approved once it meets all quality required.

# References

1, T. P. (2018, August Friday). *Canvas Unstructure.* Retrieved from Test Plan and Report 1: https://canvas.instructure.com/courses/1289269/assignments/8025888

WestFallTeam. (2010). Writing Testable Requirements. *Writing Testable Requirements*, 1-24.