Integration Test Plan

Project Name:	
SyncGallery+	
Project Manager:	**Revision:**
Will Goreshi This test plan was done by: Hithagna Dinesh (SQA Engineer from the Testing Team)	2.0

Table of Contents

Reference Material

1. IEEE Std 829-1998, IEEE Test Plan, IEEE 1998

2. Software Quality Assurance/Test Plan (SQATP)
   Project – SyncGallery +
   (From Phase A)

1. Objectives

The main objective of this integration testing plan is to make sure that the interaction of the different components used to create the SyncGallery+ application produces results that satisfy functional requirements.

Other objectives:

- Cause failures involving the interactions of the integrated software components when running on a single platform.
- Report these failures to the software development team so that the underlying defects can be identified and fixed.
- Help the software development team to stabilize the software so that it can be successfully distributed prior to system testing.
- Minimize the number of low-level defects that will prevent effective system and launch testing.

2. Testing Responsibility

SQA Engineers (testing team):

- Hithagna Dinesh: Test cases, Test procedures.
- Sarah Yang: Test cases, Test procedures.
- Will Goreshi: Document Results.

3. Requirements

- Units which have passed their test cases within the different components and ready to be tested with other components.
- It is important the right versions of components are tested.
- Code that is being tested is completely commented by the programmers

4. Defect Tracking

Integration Testing will lose its edge if the defects are not tracked correctly. Each defect should be documented and tracked. Information should be captured as to how the defect was fixed. This is valuable information. It can help in future integration and deployment processes.

5. Assumptions

During the coding phase, lots of assumptions are made on how the programmer will receive data from different components and how data will be passed to different components. Purpose of integration testing is also to make sure that these assumptions are valid. There could be many reasons for integration to go wrong, it could be because of

- Interface Misuse

- Interface Misunderstanding

Therefore, interface use and interface understanding will be tested in each round of the integration testing. This is to ensure that the programmers do not make any component calls that call another component which results in an error in the use of its interface. This will probably occur by calling/passing parameters in the wrong sequence. The integration on aviary and drop box should not have this error in order for it to integrate properly with the natural code. Also, integration testing should ensure that calling components do not

6. Test Procedures

6.1 Test Approach

A bottom-up testing approach will be sufficient to do at this stage due to the fact that the SyncGallery+ application software is near completion. The components will be separated into the level of importunacy and the least important modules will be worked on first, then slowly you would work your way up by integrating components at each level before moving upwards. This method will be used since the prototype is close to completion.

. 6.2 Test Environment

The will be a built in library for our test files and test cases with a set API to test from.

6.3 How will the test be completed/strategy?

The tester will follow the following:

Step 1: Create a test plan for each integration case.
Step 2: Create Test Cases and Test Data
Step 3: If applicable create scripts to run test cases
Step 4: Once the components have been integrated execute the test cases
Step 5: Fix the bugs if any and re test the code
Step 6: Repeat the test cycle until the components have been successfully integrated

6.4 What platform will be used?

The platforms that will be used are both Hardware and software packages which include the android ICS and the phone baseline release candidate along with the latest candidate from development team.

6.5 What interfaces will be tested?

- Aviary
- Drop Box

- Natural code done by the programmers

6.6 How many 'tests' make an acceptable level of test?

Three integration test cases will be efficient considering that 3 major integrated components are being tested.

This is how the unit tests for the integration testing will be recorded.

Table 3-1 - Test Cases *(*** Example 1 ***)*

Program Unit Name	**Testing Order**	**Special Tests**
Program Unit 1		
Program Unit 2		
Program Unit 3		

7. Exit Criteria

- A test suite of test cases exists for each interface between software components.
- All software integration test suites successfully execute (i.e., the tests completely execute and the actual test results match the expected test results).

8. Testing Tools

JUnit: Built in testing took for android apps in eclipse.

9. Testing Facilities

- Equipment – desks, books, computers, network
- Location – conducive to testing, easily accessible for the team, possibly consider an off-site testing facility for customers or users.
- Process simulations – If there are any process changes that have been accomplished during the development of the system be sure to incorporate those changes into the Acceptance testing.

10. Risk

- Units which are required for the integration testing fail their test cases.
- Programmers have not completed coding the units required for the integration testing.
- Testers are unable to execute test cases due to bugs in the test case code.