Software Quality Assurance/Test Plan (SQATP)

Project – SyncGallery +

Revision History

Date	Revision	Description	Author
29-Jan-12	1.0	Document Created	W. Ghorishi
07-Mar-12	1.1	Document Updated	W. Ghorishi

Acronyms

Acronym	Definition
SQA	Software Quality Assurance
SQATP	Software Qaulity Assurance/Test Plan
SOW	Statement of Work
ECR	Engineering Change Request
EPR	Engineering Problem Report
STP	Software Test Plan
SRS	Software Requirements Specification
SDD	Software Design Documentation
SPMP	Software Project Management Plan

Reference Material

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

Team SyncGallery+ , Software Project Management Plan – Project SynGallery +, 2012

Team SyncGallery+ , Functionality & Feature Specification Document – Project SynGallery +, 2012

Team SyncGallery+ , Software Design Documentation – Project SynGallery +, 2012

Team SyncGallery+ , Unit Test Plan & Test Cases – Project SynGallery +, 2012

Team SyncGallery+ , Integration Test Plan & Test Cases – Project SynGallery +, 2012

Team SyncGallery+ , System Test Plan & Test Cases – Project SynGallery +, 2012

Table of Contents

1 Introduction

This document describes the Software Quality Assurance (SQA) program and test plan for the SyncGallery+ application, conducted in Toronto, Ontario. The objective of this SQA program will be to establish basic requirements for the design, development, test, verification, release and maintenance of software.

As applicable, other SyncGallery+ software development team members (at external facilities from Toronto, Ontario) who will develop software that will be integrated into Handheld software group category I or category II software (as defined in paragraph 2) will submit to this SQA Plan.

Software and firmware are both referred to as software for the purpose of this document

2 Scope

This plan specifies the generic SQA activities for the SyncGallery+ development project.

The plan covers activities for core and test software development, with generic activities that are always implemented for customer-specific features development.

The appendices to this plan can address an instantiation of this SQA plan on annual basis, or for a specific customer. The instantiation will address the following items:

- Applicability of this SQA Plan including additional SQA requirements mandated by customer Statement of Work;
- Schedule for conducting SQA activity,
- SQA Organization structure and resources,
- Risks and Mitigation Plans for achieving stated SQA requirements
- Skills/Training
- Human Factors
- Elapsed time

Note: For details on schedule, including the feasibility of implementing the generic SQA requirements specified in this plan, please refer to the Quality Manager's Roadmap.

The software categories identified and referenced in this plan are as follows:

- I Category I includes
  - Core Driver Software: this includes application functionality including public and published interfaces, which are designed to work with the current iteration of the SynchGallery+ application as well as any future extensions to it, while maintaining context with respect to the platform and OS requirements, specified in the SRS & SPMP;
  - Platform Software: Implementation of required features for a particular customer product(s); future extensions of the application
- II Category II includes
  - Software used to test core functionality as well as any public/published interfaces for qualification/acceptance and release, and
  - Software to test implementation of required features for a particular customer product and,
  - Support software used in the design, and development of deliverable software.

3 SQA Organization

Within the SyncGallery+ software development team, multiple people will concurrently develop software to be delivered to customers.

The ‘Core’ software team develops the generic framework required by the application including the integration of the Dropbox cloud interface as well as the Aviary image processing open source components.

“Platform’ software teams develop, support, and customize solutions to deliver the individual feature requirements, as specified in the Software Requirements Specification (SRS). The Platform software teams act as advocates in getting new functionality and support from the core software development team. It should be noted that given the small size of the development organization, there will be shift of resources from core to platform team, during different phases of the project.

Resources from the SQA team are assigned by the Quality Manager to ensure the quality of development products of each of the software development teams.

The SQA staff technically assures the software standards, plans, and processes and evaluates the activities of the core and platform software development and release management groups. SQA is also responsible for software acceptance testing and for system software testing including the development of test tools, test cases, test plans and test reports. The primary internal and external contact for SQA is the Quality Assurance Manager, William Ghorishi. The Quality Assurance Manager is responsible for resource allocation and for the technical content of the SQA tasks, including the Test Plan. The technical content will also include the development of SQA methods and processes as required.

Supporting the QA Manager, Sarah Yang will be overseeing the Test Case Development function for the SyncGallery+ project while Hitha Dinesh will be spearheading the Verification and Validation mandate in this organization.

Table 1: SQA Activities

		Software Categories	
	SOFTWARE DEVELOPMENT LIFECYCLE INDEPENDENT SQA ACTIVITIES	I	II
a	SQA Audits	X	
b	Progress Reporting	X	x
c	Outside Developed Software	X	
d	Third Party Furnished Equipment/SW	X	
e	Training	X	X
f	Checklists	X	
g	Reusable Software		X
	SOFTWARE DEVELOPMENT LIFECYCLE DEPENDENT SQA ACTIVITIES	I	II
a	Planning Evaluation	X	
b	Requirements Evaluation	X	X
c	Design Evaluation	X	
d	Milestone Reviews	X	
e	Software Design and Code Reviews	X	
f	Configuration assessment of software tools	X	X
g	Software Process Evaluation	X	X
h	Software Coding Unit and Functional Testing reviews	X	
i	Software Development Library Control	X	X
j	System Software Testing (Use Cases)	X	X
k	Software Acceptance	X	
l	Software Problem Reporting	X	X
m	Software Release	X	X
n	Configuration Management	X	X
o	SQA Records	X	X

4 Software Quality Assurance Activities

The SQA tasks have been classified as software lifecycle independent and software lifecycle dependent activities in order to provide visibility and assurance that the objectives of each phase of the software development life cycle have been met at the appropriate time. For software lifecycle dependent activities, each activity described hereafter specifies the phase where it will be performed and the document(s) that need to be prepared and/or available from the core or platform software development group(s).

4.1 Software Development Lifecycle Independent Activities

The following SQA activities are software development lifecycle independent SQA Activities and are applicable as shown in **Table 1**:

4.1.1 SQA Audits

The SQA Engineer will audit

- Core software procedures,
- Software Development Plan for each project associated a major new core development or with a major development for a customer.

As an example, core software procedures may include the following:

- Document Control Procedure
- Change Control Procedure
- Defect Classification and Handling Procedure
- Configuration, Build and Release Management Procedure
- Release Development Procedure
- Corrective / Preventive Action Request Procedure
- Project initiation
- Project Estimation Procedure

- Project Tracking Procedure

Audits will be sufficiently detailed to allow evaluation of both the progress and effectiveness of development and SQA tasks and the need for adjustments or changes.

4.1.2 Software Process Evaluation

Standards for software processes and products will be reviewed and approved by SQA at every issue, for contract compliance if required, and for technical adequacy.

Following SQA approval of a software development standard or process, any change to that standard or process will require a reevaluation by SQA.

The SQA Engineer will evaluate the software development process to identify and/or control actual and potential problems which could introduce latent failures into software.

4.1.3 Progress Reporting

The SQA group will prepare two progress reports of SQA activities. The first report will be part of the overall software department status reporting. The second, for inclusion into project status reports, will be submitted through the project manager to the customer. The frequency and content of customer specific progress reports will be as agreed upon in the customer's Statement of Work (SOW).

4.1.4 SyncGallery+ Remote Teams

A SQA Engineer may fly to other development locations to ensure the requirements specified in this document are upheld. The SQA Engineer will have the right to verify these developers' software standards, procedures manuals, and activities to ensure compliance with the project requirements.

4.1.5 Third Party Furnished Equipment & Software

A SQA Engineer will formally and promptly notify the Project Manager upon receipt of such deliverables, or while testing or otherwise operating the deliverable, it is determined that its characteristics are not consistent with the requirements of the project.

4.1.6 Configuration Assessment of Software Tools

A SQA Engineer will assure that software production tools, such as compilers and code checkers, are maintained to an approved configuration and that proper operation of these tools following system update is demonstrated before resuming production operations. In-house developed non-deliverable support software which interacts with deliverable software will be controlled in accordance with Table 1: SQA Activities for category II software.

4.1.7 Training

SQA personnel performing software testing, inspections, or analysis will have the proper training and qualifications.

4.1.8 Checklists

A SQA Engineer will prepare checklists to review and evaluate documents, plans or procedures, as appropriate. The checklists will be used to identify and track discrepancies.

4.1.9 Document Standardization Assurance

Software development and SQA approved standards lay down forms for recording data and templates for documents. Examples of templates that may be generated include:

- SPMP (Software Project Management Plan)
- SRS (Software Requirements Specification)
- SDD (Software Design Documentation)
- STP (Software Test Plan)
- STD (Software Test Descriptions)
- Release Notes
- Version Description Document (VDD)
- Waiver / Deviation Form
- Release Approval Form

Part of the review procedure (or audit when applicable) is to compare the review material with the standard forms and templates. SQA will check the document for integrity, clarity and adherence to template. When defects are found, they will be documented and provided to the document owner. Once the SQA comments have been addressed, the document can be approved and released.

4.1.10 Software Development Plans Assurance

Software Development Plans will be reviewed (and approved) for contract compliance if required, feasibility and consistency with other documents, including schedules. The review process will start with informal liaison during the development of the plan. SQA will provide input and will also participate in the internal and external review meetings. SQA signature on software development plan provides assurance to the customer with respect to an independent evaluation of the compliance to the applicable project requirements and standards.

4.1.11 Evaluation of Development Documents

SQA will assure a variety of documents and software media as part of its duties. SQA assurance of the contents of material submitted for review or approval varies from reading and checking traceability tables included in the document, as is the case for development plans, procedures, and ECRs that supports new feature requests, to ensuring that an appropriate review by groups other thA SQA has taken place. If SQA is not satisfied that an adequate technical review has occurred, it will withhold approval of the document until the proper review has been performed and issues are addressed. SQA will consider the category of software to which the document pertains when determining the amount of effort to be expended when reviewing a document.

4.1.12 Evaluation of Risk Management

The software group and SQA have the responsibility for reporting systematic risks to the Project Manager and for assuring that mitigation or contingency plans to reduce those risks are developed. SQA will survey the risks, plans and actions connected with software and raise notices through the organization reporting channels.

4.1.13 SQA Role in Deviations and Waivers

When the development group plans a course of action that is inconsistent with project standards and policies, they are required to raise a deviation request to the customer Configuration Control Board (CCB), with technical support and project justification for their proposed actions. SQA will review software related deviation requests and provide comments to the platform software CCB. This board consists of the Project Manager and technical leads as appropriate.

In cases of non-compliances to project requirements, standards, policies or to a parent document, the software development group may request a waiver to use the software as is. SQA will review the proposed waiver and will provide comments to platform software CCB regarding the disposition.

4.1.14 Metrics

SQA will ensure that quality goals are established and that metrics are generated to assess achievement of these quality goals. The following is a list of metrics that may be generated:

- Test Status (Q)
- Defect Arrivals and Closure (Q)
- Escaped Defects (Q)
- Open Aging Defects (Q)
- Code Inspection Quality (Q)

4.2 Software Development Lifecycle Dependent SQA Activities

The following SQA activities are software development lifecycle dependent SQA Activities and are applicable as shown in **Table 1**:

4.2.1 Development Planning Evaluation

The software development plan will be evaluated to ensure feasibility, consistency with system development schedules, resource allocation, that activities and products for each phase have been defined, that the conventions, standards and procedures have been defined and that the plan is consistent with contractual requirements (i.e. Customer Statement of Work).

4.2.2 Requirements Evaluation

A SQA Engineer will evaluate the software requirements specification (SRS) (including interface requirements or use cases, if applicable) to ensure traceability to parent specifications and design documents, and for clarity and testability.

4.2.3 Design Evaluation and Software Design Reviews

The core development software design document (SDD) and the software test plan/descriptions (STP/STD) will be assessed at the architectural and detailed levels for adherence to approved software standards and procedures and for traceability to the software requirements.

A SQA Engineer will participate in planned core software design reviews to ensure completeness and accuracy of the material presented and to verify that the design satisfies functional requirements, operational standards and performance specifications.

4.2.4 Milestone Reviews.

A SQA Engineer will support the Customer's milestone reviews to reviews activities, accomplishments, and related technical documents. SQA Engineering participation in these reviews will ensure that SQA requirements are adequately considered in decisions affecting software design, production, testing, and operation, and that results from SQA technical activities are used to support the applicable milestone reviews.

4.2.5 Software Coding, Unit and Functional Testing Reviews

SQA will participate in selected code walkthroughs where activities will include a review of the software code to ensure compliance with software requirements and SynGallery+ Software Coding Standard.

Selected listings for software components will be checked for coverage of functional requirements, structure, naming conventions and readability. The criteria of selecting certain listings for SQA review will depend on the development phase (i.e. new functionality versus modification to existing code and on module risk of implementation).

Daily builds of checked-in code will be performed to ensure code integrity and consistency.

The software development team will produce the software test descriptions. SQA will review and assess test descriptions, monitors unit and functional test activities to ensure that test objectives are achieved; assesses test results to ensure that requirements are adequately demonstrated; and assures that problems are documented using the problem reporting system (TRAC). SQA will assure that the problems are closed after a peer review of whether the problem is a result of a coding error or a test case error and assures the fix up accordingly.

SQA will also be responsible for the following activities:

- Verification of configuration control of the integrated items
- Verification of the maintenance of the software development folders (software depots),
- Reviewing updates of the unit and functional test cases
-

4.2.6 Software Development Library Control

A SQA Engineer, through periodic checks, will ensure that procedures and controls for the proper handling of software code and related data in their various forms and versions (as defined in a Configuration Management Plan) are adhered to. The objective of these controls will be to ensure that different computer project versions are accurately identified (and documented, if required), that only authorized modifications are made, and that the software submitted for testing is the correct version. A library index containing the following data elements for software components will be maintained and be accessible: project name, title and identification number, revision number, entry date, description, and subsystem/system name.

4.2.7 Software Acceptance Testing

SQA will write a formal Acceptance Test Plan and, prior to a formal core driver software release, will test the software against the requirements (including use cases where applicable) in order to formally demonstrate compliance with requirements. .

The SQA Engineer will assess test results to ensure that requirements are adequately demonstrated; and will ensure that discrepancies are reported in TRAC problem reporting module. SQA will hold a Test Results Review Board (TRRB) where the Acceptor will be notified of the test results.

SQA will assure that the code is only released by Release Management upon receipt of indication of Acceptance from the Acceptor. Outstanding discrepancies will be documented in that particular SW release “Release Note”.

4.2.8 System Software Testing

SQA will write a formal System Software Test Plan, and prior to a customer release, will test the system software against the requirements (including use cases where applicable) in order to find errors and latent problems.

The SQA Engineer will assess test results to ensure that requirements are adequately demonstrated; and will ensure that discrepancies are reported in TRAC problem reporting module When discrepancies are found, but a release is nevertheless required by the customer, SQA will assure that the customer is notified through the Project Manager of the test results (through a TRRB) and that code is only released by Release Management upon receipt of agreement for release from the customer. In that case, SQA will ensure that discrepancies are documented in that particular platform customer release “Release Note”.

4.2.9 Post Implementation Review (Post Mortem)

On periodic basis, SQA will ensure that a review is held at the conclusion of a core or platform release or conclusion of a project to assess the development activities implemented and to provide recommendations for appropriate action.

4.2.10 Software Problem Reporting

The SQA Engineer will ensure that the TRAC EPR module is used for problem reporting and tracking. SQA will also ensure the timely and accurate identification and resolution of software problems discovered during SQA testing, those reported by customers, and those found during the development process. Through review of EPR, SQA will ensure the completeness and clarity of problem reports raised.

Weekly problem review boards (PRB) will be held to evaluate the impact of each problem on the current stage of development, to review progress of work on previously discovered problems, and to review problems prioritizations. SQA will also ensure that each fixed problem is a source of new test cases to be incorporated into the SQA testing process and that a change list number is identified for the code fix.

4.2.11 Software Release

SQA will ensure that the release process is identified in the CM Plan and, through periodic checks, that it is followed. A SQA Engineer will ensure that deliverable software conforms to software release requirements as defined in the CM Plan.

The release does not include only binary files. For some customers, software development group provides additional files to facilitate development. Different customers receive different sets of components, according to the program requirements and agreed upon NDA. In this case, SQA will ensure that the CM team creates the final package for distribution to each customer. Prior to announcing the release, the SQA team tests packaging and verifies the complete process of installation of the release package.

At the conclusion of acceptance tests, SQA ensures that a Release Note and, when required by customer, a Version Description Document (VDD) or Configuration Specification are accurately generated that:

- Provides precise information of the new release,
- Describes the difference between the current release and the previous release

- Contains a detailed list of fixed and outstanding problems and new features,
- Outlines specifically how to install the new release.

Note: the Configuration Specification and VDD offer more details than the Release Note.

4.2.12 Configuration Management

SyncGallery+ will use the TRAC tool Engineering Change Request (ECR) process to manage feature requests (requirements) related to core and platform software development. In addition, SyncGallery+ uses SVN Software Configuration Management System as its version control software (VCS) tool. The tool generates a change list number that uniquely identifies changes made to the code.

Configuration Management policies and practices of data products will be defined in a CM Plan.

SQA will ensure the use of the TRAC tool to manage feature/requirements requests and their development and will assure the requirements management and tracking process through participation in design reviews and feature prioritization meetings.

SQA will review the change records in the code database to assure that development is in accordance to software requirements specification and quality requirements

Software configuration management (SCM) will be achieved through Library Control and the CCB, where the SQA Engineer will be a member. The SQA Engineer will periodically conduct checks to ensure that the identification, status accounting, change control and maintenance of the software library are as required.

4.2.13 SQA Records Storage and Submission, when required, to the Customer.

The SQA Engineer will ensure the retention and availability of the following records:

- Audit reports
- Metrics
- Daily build results and test results
- Meeting Minutes

- Problem Reports
- Sub-Test Plans

4.2.14 SQA Status Reporting

SQA Status Report will cover, as minimum, the following areas:

- Status of SQA project and internal activities, problems and accomplishments;
- A summary table or updates, as applicable of:
    - Test result
    - Metrics;
    - Other SyncGallery+ development team status

5 High Level Test Strategy

5.1 Functional Requirements

The following highlights the techniques that will be applied to execute different levels of testing. Please refer to the plan/test-case documents for each of the following phases of testing as referenced in this document.

- Unit Testing
    - Unit test cases will be implemented and coded as APIs to enable generating of all possible input scenarios (including corner cases) and validate them against positive and negative test cases
    - The unit test API calls will be embedded in the application in a non-intrusive manner to executed and validate the code
- Integration Testing

- Integration test cases will be similar to how unit test cases are executed with the exception that these test cases will focus on the APIs delivering an integrated functionality (interface APIs between DropBox & Gallery or Gallery and Aviary)
- The integration test API calls will be embedded in the application in a non-intrusive manner to executed and validate the code

- System Testing
    - System testing will be performed manually and will focus on UI driven scenarios such as capturing an image, modifying it with a image filter (ie grayscale) and uploading the modified image to cloud to make it available to other users of the system
    - Ideally, given an extension to the schedule or

5.2 Non-Functional Requirements

The functional requirements that will be tested and measured for the SyncGallery+ project include:

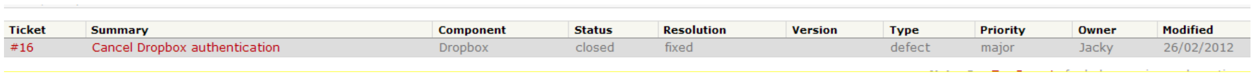
- Correctness and reliability
- Extensibility and maintainability
- Responsiveness and intuitiveness

5.2.1 Correctness and reliability

Given the scale of the project, the short development cycle as well as limited number of defects found, extrapolating a trend on the defects (poorly design/implemented modules, poorly defined requirements, etc.), realizing a pattern (release schedule predictions) or generating other meaningful information based on the bug details will be statistically inaccurate. Having said that, the following is the current list of defects and relevant information as recorded in Trac, the defect tracking tool used for SyncGallery+.

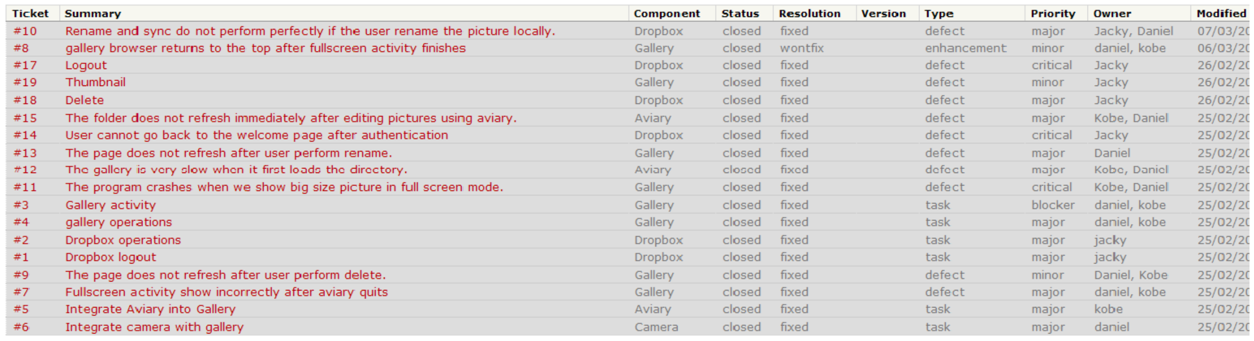
The following are the list of defects and their status, based on releases, as defined in the SPMP:

Alpha:



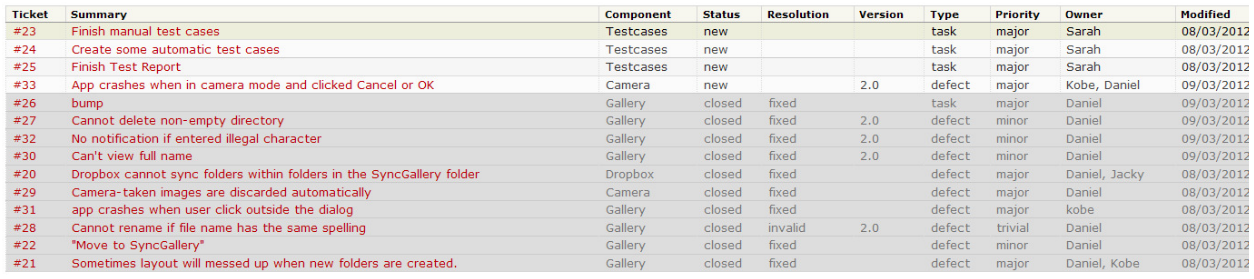
Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#16	Cancel Dropbox authentication	Dropbox	closed	fixed		defect	major	Jacky	26/02/2012

Beta:



Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#10	Rename and sync do not perform perfectly if the user rename the picture locally.	Dropbox	closed	fixed		defect	major	Jacky, Daniel	07/03/20
#8	gallery browser returns to the top after fullscreen activity finishes	Gallery	closed	wontfix		enhancement	minor	daniel, kobe	06/03/20
#17	Logout	Dropbox	closed	fixed		defect	critical	Jacky	26/02/20
#19	Thumbnail	Gallery	closed	fixed		defect	minor	Jacky	26/02/20
#18	Delete	Dropbox	closed	fixed		defect	major	Jacky	26/02/20
#15	The folder does not refresh immediately after editing pictures using aviary.	Aviary	closed	fixed		defect	major	Kobe, Daniel	25/02/20
#14	User cannot go back to the welcome page after authentication	Dropbox	closed	fixed		defect	critical	Jacky	25/02/20
#13	The page does not refresh after user perform rename.	Gallery	closed	fixed		defect	major	Daniel	25/02/20
#12	The gallery is very slow when it first loads the directory.	Aviary	closed	fixed		defect	major	Kobe, Daniel	25/02/20
#11	The program crashes when we show big size picture in full screen mode.	Gallery	closed	fixed		defect	critical	Kobe, Daniel	25/02/20
#3	Gallery activity	Gallery	closed	fixed		task	blocker	daniel, kobe	25/02/20
#4	gallery operations	Gallery	closed	fixed		task	major	daniel, kobe	25/02/20
#2	Dropbox operations	Dropbox	closed	fixed		task	major	jacky	25/02/20
#1	Dropbox logout	Dropbox	closed	fixed		task	major	jacky	25/02/20
#9	The page does not refresh after user perform delete.	Gallery	closed	fixed		defect	minor	Daniel, Kobe	25/02/20
#7	Fullscreen activity show incorrectly after aviary quits	Gallery	closed	fixed		defect	major	daniel, kobe	25/02/20
#5	Integrate Aviary into Gallery	Aviary	closed	fixed		task	major	kobe	25/02/20
#6	Integrate camera with gallery	Camera	closed	fixed		task	major	daniel	25/02/20

Release Candidate:



Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#23	Finish manual test cases	Testcases	new			task	major	Sarah	08/03/2012
#24	Create some automatic test cases	Testcases	new			task	major	Sarah	08/03/2012
#25	Finish Test Report	Testcases	new			task	major	Sarah	08/03/2012
#33	App crashes when in camera mode and clicked Cancel or OK	Camera	new		2.0	defect	major	Kobe, Daniel	09/03/2012
#26	bump	Gallery	closed	fixed		task	major	Daniel	09/03/2012
#27	Cannot delete non-empty directory	Gallery	closed	fixed	2.0	defect	minor	Daniel	09/03/2012
#32	No notification if entered illegal character	Gallery	closed	fixed	2.0	defect	minor	Daniel	09/03/2012
#30	Can't view full name	Gallery	closed	fixed	2.0	defect	minor	Daniel	09/03/2012
#20	Dropbox cannot sync folders within folders in the SyncGallery folder	Dropbox	closed	fixed		defect	major	Daniel, Jacky	08/03/2012
#29	Camera-taken images are discarded automatically	Camera	closed	fixed		defect	major	Daniel	08/03/2012
#31	app crashes when user click outside the dialog	Gallery	closed	fixed		defect	major	kobe	08/03/2012
#28	Cannot rename if file name has the same spelling	Gallery	closed	invalid	2.0	defect	trivial	Daniel	08/03/2012
#22	"Move to SyncGallery"	Gallery	closed	fixed		defect	minor	Daniel	08/03/2012
#21	Sometimes layout will messed up when new folders are created.	Gallery	closed	fixed		defect	major	Daniel, Kobe	08/03/2012

A few data points worth noting are:

- Total # of defects/failures: 22
- Total # of faults: 0 (Understandable given the heavy involvement the team had in requirements phase)
- Total # of enhancements: 1
- Total # of tasks: 10
- Total # of rejected defects/failures: 1

Other notable observations are:

- Defect count increases towards Beta release and subsequently decreases towards the RC (typical SW release trend)
- The number of faults in the system is zero as the development team was heavily involved with the requirements definition phase for the project.

5.2.2 Maintainability & Extensibility

Evaluating the details for each of the above defects and correlating them to the changes made in the source repository for each defect (the mechanics around this process should be improved; current examination was done using manual techniques), the data shows that the defects reported were mostly related to the following integration points:

- Gallery integration with Dropbox
- Aviary integration with Gallery
- UI and associated workflows

Hence from a maintainability and extensibility perspective, the functional blocks used in SyncGallery+ including Dropbox and Aviary are highly adaptable and extendable. Dropbox API sequences can be incorporated into business logic as required. Furthermore, the simplicity associated with the integration framework used in SyncGallery+ will allow maintainability of the application.

Future evaluations of maintainability and extensibility of the SyncGallery+ project can leverage Doxygen, but given the small scope and short duration of this project, this technique was deemed not feasible at this point in time.

5.2.3 Responsiveness & Intuitiveness

One of the most critical yet often neglected non-functional requirements for a software application is how easily a novice user can operate it and how responsive the application is to different sequences invoked by the user. To measure these attributes for the SyncGallery+ project, the following measurements were taken.

Number of user steps per major use-case:

- Take a picture (4 steps):

  Enter the application→ press “Gallery” → press ”Menu” → press “Camera” (It will call the built-in camera).

- Applying a simple photo-edit (5 steps):

  Enter the application→ press “Gallery” → Go to the directory of the photo → Press “Menu” → Choose “Edit”

- Synchronizing pictures (6 steps):

  Enter the application→ press “Connect” → login / register → Click “Allow” → Press “Menu” → Choose “Sync”

- Share a picture (5 steps):

  Enter the application→ press “Gallery” → Long-click the picture → choose share → select the method of sharing

- Other folder/images operations (4-steps):

  Enter the application→ press “Gallery” → Long-click the picture → choose operation

- Search (6 steps):

  Enter the application→ press “Gallery” → press ”Menu” → press “Search” → Type the keyword → press the search button

- Logout (4 steps):

  Enter the application→ press “Connect” → press ”Menu” → press “Logout”

Based on the above results, the application is highly intuitive, in comparison to some other popular mobile apps available (Facebook, etc.)

6 High Level Schedule

Following the agile development methodology adopted by the SyncGallery+ team, test-case development and execution will be done in parallel to feature development. Hence, please refer to the milestones and feature readiness schedule as provided in the SPMP.

If a feature is shown as ready, the required test cases have been fully developed and executed against the feature in order to declare it as complete.

Release level testing is also done, leveraging the test cases provided under the context of unit, integration and system level testing. Hence, the release test cycle can be fairly short as all of the test cases have been executed against the features developed as well as part of the periodic regression test cycle.

7 Test Cases

Please refer to the Unit, Integration and System Test Plans for detailed documentation of test-cases executed under each phase.