# AngSalitaNgDiyos.com

# Liturgical Site

Software Quality Assurance Plan

In Partial Fulfillment of the Requirements

in Software Quality Assurance

By:

Dimapilis, Joshua C.

Elizondo, Kimberly Mae B.

Urquiza, Trixia Marie A.

IT - 111

February 2015

1. **Software Project Management Plan**

This Software Quality Assurance Plan document, was developed specifically for the project *AngSalitaNgDiyos.com Liturgical Site.* During the project’s *Planning stage,* the proponents arrived with the following description of the ventures at hand. The purpose of this representation is to *accurately reflect* the project’s *scope* and *structure.* The following will assess the planned activities to be fulfilled throughout the cycles of the project.

1. **Project Structure**

The following sections of the *Project Management Plan* will be discussed in length in the succeeding segments:

* Vision and Scope
* Feasibility and Risk Analysis
* Management Approach
* Technical Approach

In the event of *end user reviews*, these first two chapters will initially be recommended for the reviewer’s benefit.

1. **Project Content**
   1. **Vision and Scope**

The succeeding sections describes the conditions driving the development of the calendar tool. This section introduces the application as intended and describes the scope and limitations of the development effort.

* + 1. **Vision**

In the advent of the technology age, more and more industries are beginning to grasp the benefits of having an *online presence.* This is why even the *religious* sector of our country aims to enhance their capability of *catering* the needs of their intended spiritual audience, online.

*AngSalitaNgDiyos.com*, is an online Catholic Liturgical site that accommodates the online audience and their needs, spiritually and information - wise.

In line with this, the proponents of this project have tasked the IT – 111 students to create a calendar structure that can display the *Lectionary Cycles* and *Liturgical Feasts* within a certain year. It is known that every year, the *liturgical* calendar changes in line with certain technicalities defined by Lectionary Cycles (3 cycles for Sunday Lectures, 2 cycles for Weekday Lectures). The proposed calendar structure will identify specifically when these dates will be and arrange them accordingly.

The primary goal of the *Liturgical Site’*s calendar structure is to provide an informative, detailed and accurate representation of all the lectionary cycles and religious feasts within a specific year. This will designate the said dates and represent them in an online version of the Calendar.

* + 1. **Scope**

The project aims to provide a tool that helps online users to identify the designated dates of *religious lections and feasts* within a specified year.

* Create a database that includes all the *religious events* (e.g. lections, feasts and celebrations) within the year
* Create a tool that utilizes the created database to insert the specified *events* in line with specific requirements that are specified by the Catholic church
* Provide a calendar (that uses the database and the tool’s synergy) to display links to the necessary *pages* and/or *audio* files, representing lections and mass readings for each date
* Encourage the online community to visit the *AngSalitaNgDiyos.com* site for informative and spiritual purposes

The functions of the system include the following:

* The main target market of this tool will mainly be defined as *users*. These users are those who access the calendar via the site, *AngSalitaNgDiyos.com*
* The calendar tool will automatically display the dates within the month that it is being accessed
* The calendar will display 4 types of *events*: (1) **Sunday readings**, (2) **Weekday readings**, (3) **Moveable feasts**, (4) **Solemnities**,(5) **Memorials** and (6) **Special Feasts**
* Each type of item has its own definite *business requirement*
  + **Sunday readings** have 3 yearly cycles: Year A, Year B and Year C. Each cycle has a designated set of Sunday readings which are variably different from each cycle. Year determinant is the remainder of the sum of all the digits within the year, divided by 3. The succeeding table illustrates the determinant.

|  |  |
| --- | --- |
| Remainder | Year |
| 0 | Year A |
| 1 | Year B |
| 2 | Year C |

Table 1.1 **Determinant of year cycle for Sunday readings**

Example -For the Year 2013:

**Sum of all digits in the year**

= 2+0+1+3= 6  
**Remainder of sum divided by 3**

= 6 / 3 = 2 remainder 0Remainder is 0.Therefore 2013 is considered within the Year A cycle.

* + For **Weekday readings,** there are two sets of readings. Year 1 &Year 2. The succeeding table describes the determinant.

|  |  |
| --- | --- |
| Year Type | Year |
| Odd | Year 1 |
| Even | Year 2 |

Table 1.2 **Determinant of year cycle for Weekday readings**

Example:

Year 2013 is an odd year. Therefore, it is within the Year 1 cycle.

* + In the case of **Movable feasts,** the dates are set based on other movable feasts / solemnities within the year. (Some examples include *Easter Sunday, Palm Sunday* and *Pentecost Sunday*)
  + For **Solemnities** and **Memorials,** they have their own set of readings, which replace Sunday and/or weekday readings excepts for specific Sundays (i.e. Sundays in Advent, Lent and weekdays of Holy Week / Easter Octave).Solemnities and memorials are big – time feasts. (e.g. *Presentation of the Lord,* and *Annunciation of the Lord*)
  + For **Special Feasts,** these dates are static and are not day – sensitive.
* Event items such as *Movable feasts, Solemnities* and *Memorials* will be displayed as text, which will serve as markers for the specific events they represent
* These items that will be displayed in the calendar, will be in the form of *links* that will redirect to specific *pages* / *audio files*

The tool is envisioned to possess the following versions of the application framework:

* Web – version
* Mobile – version

The timeframe for the system development process is ten weeks; for the finished product, the scheduled system evaluation is March 31, 2015.

* 1. **Feasibility and Risk Analysis**
     1. **Feasibility Standards**

This sectionaddresses the issuesof *application complexity* as well as the anticipated *risks* in *schedule* and *operation* procedures*.* The following factors have been defined in order to verify the project’s feasibility. The table below illustrates these points and the team’s proposition for the development and quality assurance stages of the project.

|  |  |  |
| --- | --- | --- |
| Standard | Risks / Issues | Proposition |
| Application Complexity | * The various requirements included in the calendar warrants well – defined conditions and proper synergy between the database and the tool. * The integration of the tool to the site itself should also be considered of vital importance | * Proper standards in code development must first be defined, upon establishment of these defined standards, following it is key * The usage of the tools provided by our *adviser*, and a proper understanding of the MVC Framework is essential * Development procedures must be in line with the professional opinions of our *mentors* and *adviser* * For System Integration, it must be done in line with the system |
| Schedule Constraints | * The allotted timeframe for the project development and quality assurance phase is a maximum of 10 weeks. | * Development – wise, the tool’s functions must be assessed if the 10–week development process can accommodate the necessary changes and coding efforts by the team * In the Quality Assurance phase, the tool should be assessed by certain metrics and success factors to be defined in the latter part of this section * A schedule for both *SOFTDEV* and *QUALITY* has been given, and for the success of the development and testing phases, the team’s processes must be in line with this defined schedule |
| Client, User and Organizational Risks | * The risk of obtaining an error in logic in one year will mean an inheritance of the succeeding years’ errors * The risk of having to engage in maintenance and support for the application, after development | * Proper development of the product must be implemented. The client must verify the correctness of the data in the database of the proposed system, before the implementation phase * Plans for maintenance and support must be in place, even after implementation. This plan must be proposed to the client, along with the system |
| Operational Feasibility | * Risk of web - hosting and storing massive amounts of data in the database * Risk of user apathy and unresponsiveness from the target market | * Hosting the site, with large amounts of data may be mitigated by specifying early on with the client, the expected and the actual size of the data to be stored and used for the project * Through proper dissemination in local churches, and on the web – users may be properly oriented with the goal of the tool and the site itself |

Table 1.3 **Feasibility Standards**

* + 1. **Quality Metrics**

The project may be deemed as successful if the following short – term metrics are satisfied:

* Continuity of development phase – consistent delivery of individually required features during each iteration / cycles, by meeting the scheduled evaluation of March 31, 2015
* Explicit affirmation from client – confirmation of met requirements / expectations by the users
* Convenience, Speed and Reliability – if the proposed tool was able to affect the users positively; if the users find the tool informative, and if the users feedback about the tool’s response includes it being *rapid*, *reliable* and *accurate*
  + 1. **Success Factors**

The tool’s implementation can be considered successful if it meets the following criteria:

* Assurance that the system conforms to the mentioned business requirements and client standards; receiving a passing rate in the Quality Assurance Testing Phase
* Successful integration to the implemented site, which must yield improvement in the user experience
* A positive response by the spiritual audience online; either by their valued response or by their patronization of the system / high utilization
  1. **Management Approach**
     1. **Development**

The development process to be used is the *Agile Methodology.* There will be intensive development, and succeeding iterations (0,1, and 2). Each iteration involves functional integration and detailed change requests, adjustments and tracking – in accordance to client decisions. Bugs and issues may be found during each iteration, when quality assurance testing is done, fixing these bugs will be prioritized after each iteration.

  
Image 1.1 **Agile Methodology, source:** [**Code2u.net**](http://www.code2u.net/)

* + 1. **Communication**

Meetings will be done at least thrice a week. Updates, for daily scrum and weekly status reports are located in the [repository](https://code.google.com/p/apc-softdev-it111-05/w/list). For each iteration, the team will consider each sets of 3 weeks as a sprint. There will be 3 sprints in this term. The sprints will be aligned with each iteration that has been scheduled by the team’s *SOFTDEV* adviser.

  
Image 1.2 **Sprint, source:** [**Kaeru.se**](http://www.kaeru.se/scrum.png)

* + 1. **Quality Assurance**

From the above-mentioned methodology, the Quality Assurance Testing Phase will be following the same process of *Developing, Base-lining, and Testing*.

The processes of which were mentioned above happens each iteration / sprint. The steps inside each phase, involve the following:

* Research and Analysis
* Prototyping and Designing
* Testing and Planning

Various static tests and dynamic tests have been duly scheduled in line with the iterations. Naturally, the static testing schedules have been given priority over the succeeding dynamic tests.

The following image illustrates the proponents’ intended QAT Phase of the Project (which involves Testing, Quality Control and Quality Assurance):



Image 1.2 **Testing, Quality Control and Quality Assurance,  
 source:** [**SystemsAppsControls.com**](http://systemsappscontrols.com/images/diagram.png)

* + 1. **Roles and Responsibilities**

The following responsibilities have been designated for each team member to ensure the project’s success. The table below illustrates the individual roles of each member.

|  |  |  |
| --- | --- | --- |
| Name | Roles | Responsibilities |
| Joshua C. Dimapilis | Project Manager / Developer | * Overseeing of the Project Status and Progression * Management and leadership of the Project team * Planning and Evaluation of Development and Quality Assurance |
| Kimberly Mae B. Elizondo | Quality Assurance Tester / Developer | * Quality Assurance Consulting * Monitoring of schedule, iterations and sprints * Business Requirements Analysis |
| Trixia Marie A. Urquiza | Quality Assurance Tester / Developer | * Database Design and Management Consulting * Approval of Change Requests and adjustments * Management of   documentation and scrum |

Table 1.4 **Teams’ Roles and Responsibilities**

* 1. **Technical Approach**
     1. **Technologies for Development**

The following tools are to be used for development:

|  |  |
| --- | --- |
| Generic Tools | Specific Tools |
| Programming Languages | PHP, HTML5, CSS3 |
| Database Server | MySQL |
| Web Server | Apache Server |
| Coding Tools | Yii PHP Framework, Sublime Text, fullcalendar.io |
| Mobile Phone Testing | Devices running JellyBean or newer versions of Android |
| Documentation Tools | Microsoft Office, Microsoft PowerPoint, MySQL Workbench |
| Repository | [*code.google*](https://code.google.com/p/apc-softdev-it111-05/) |

Table 1.5 **Tools Used for the Project**

* + 1. ****Use Cases**

Image 1.3 **Use Case Diagram**

* 1. **Glossary of Terms**

Below is a glossary of project – specific terms:

* Calendar
* Events
* Lectionary Cycles
* Memorials
* Movable Liturgical Feasts
* Reading Set
* Religious Events
* Solemnities
* Special Feasts
* Sunday readings
* Weekday readings

1. **Requirements Document**

*The Requirements class of deliverables are produced during the Requirements stage and updated if necessary during the Design, Development, and Integration & Test stages. The purpose of the Requirements class is to accurately define the scope, structure, and high-level functionality of the database application under design.*

1. **Requirements Structure**

The Requirements class of deliverables is composed of three related documents:

• The Logical Database Description

• The Requirements Document

• The Requirements Traceability Matrix

1. **Requirements Content**

* 1. **Logical Database Description (LDD)**



Image 1.4 **Logical Database Description**

**Entity:** **USER**

**Description:**

A **user** is a person who uses the system and access the calendar.

**Relationship:**

A user with an admin role can create, read, update and delete calendar events.

A user with a member role can view calendar events.

**Actor Interaction:**

|  |  |
| --- | --- |
| ROLE | ACTOR |
| Create | Admin |
| Read | Admin, Member |
| Update | Admin |
| Delete | Admin |

**Entity:** **YEAR**

**Description:**

A **year** is an entity that can hold zero or multiple events.

**Relationship:**

A year contains multiple dates, events, a Sunday reading cycle and a weekday reading cycle.

**Actor Interaction:**

|  |  |
| --- | --- |
| ROLE | ACTOR |
| Create | Admin |
| Read | Admin, Member |
| Update | Admin |
| Delete | Admin |

**Entity:** **DATE**

**Description:**

A **date** is defined by a month and a numerical date, and is an entity that can hold zero or multiple events.

**Relationship:**

A day can have zero or multiple events.

**Actor Interaction:**

|  |  |
| --- | --- |
| ROLE | ACTOR |
| Create | Admin |
| Read | Admin, Member |
| Update | Admin |
| Delete | Admin |

**Entity: EVENT**

**Description:**

An **event** is an occurrence that warrants a presence in the calendar. Events are relatively of great importance, and are based on the business requirements provided by the client. The event table has four *lookup* tables: Movable Feasts, Special Feasts, Memorials and Solemnities.

**Relationship:**

An event can either be classified as a movable feast, special feast, memorial, or solemnity.

Several or zero events can occur in a single day.

**Actor Interaction:**

|  |  |
| --- | --- |
| ROLE | ACTOR |
| Create | Admin |
| Read | Admin, Member |
| Update | Admin |
| Delete | Admin |

**Entity:** **YEARLY READING SET**

**Description:**

A **yearly reading set** is a compilation of readings that are separated by cycle types and numbers. (i.e. For *Sunday Readings*: Cycle A, B and C, for *Weekday Readings*: Cycle 1 and 2).

**Relationship:**

A yearly reading set contains many weekly readings.

A yearly reading set contains many Sunday readings.

**Actor Interaction:**

|  |  |
| --- | --- |
| ROLE | ACTOR |
| Create | Admin |
| Read | Admin, Member |
| Update | Admin |
| Delete | Admin |

**Entity: WEEKDAY READINGS**

**Description:**

A **weekday reading** refers to a reading that is used by lecturers and priests during weekdays.

**Relationship:**

Each weekday should have one weekday reading.

**Actor Interaction:**

|  |  |
| --- | --- |
| ROLE | ACTOR |
| Create | Admin |
| Read | Admin, Member |
| Update | Admin |
| Delete | Admin |

**Entity: SUNDAY READINGS**

**Description:**

A **Sunday reading** refers to a reading that is only used during Sundays.

**Relationship:**

Each Sunday should have one Sunday reading.

**Actor Interaction:**

|  |  |
| --- | --- |
| ROLE | ACTOR |
| Create | Admin |
| Read | Admin, Member |
| Update | Admin |
| Delete | Admin |

* 1. **Software Requirements Document (SRD)**
* **System Requirements**

a. The system must provide a login page so that the registered users can access the site.

b. The system must contain a calendar filled with generated readings.

c. The system must allow the unregistered users to sign up for membership.

d. The system must allow registered users to listen to audio files included in the calendar.

* **User Requirements**

a. The user must register first with his/her username and password.

b. The user must login to access the site.

c. The user can view the calendar.

d. The user can perform CRUD within the calendar, as long as he has admin roles.

e. The user can listen to audio files provided in the calendar.

f. The user can click links.

g. The user can view the calendar on a monthly basis.

h. The user can navigate to different months of the year.

* 1. **Requirements Traceability Matrix (RTM)**

*The RTM makes use of the analysis listings in the SRD and its parent SPMP or Component Iteration Plan (CIP) document. The purpose of the RTM is to show that each requirement is related to a specific goal in the SPMP or CIP, that all goals in the project plan have at least one associated requirement, and that no requirements in the SRD are related to non-existent goals.*

1. **Design Document**

*The Design class of deliverables are produced during the Design stage and updated if necessary during the Development and Integration & Test stages. The purpose of the Design class is to accurately define the scope, structure, and high-level functionality of the database application under design.*

1. **Design Structure**

The Design class of deliverables is composed of three related documents:

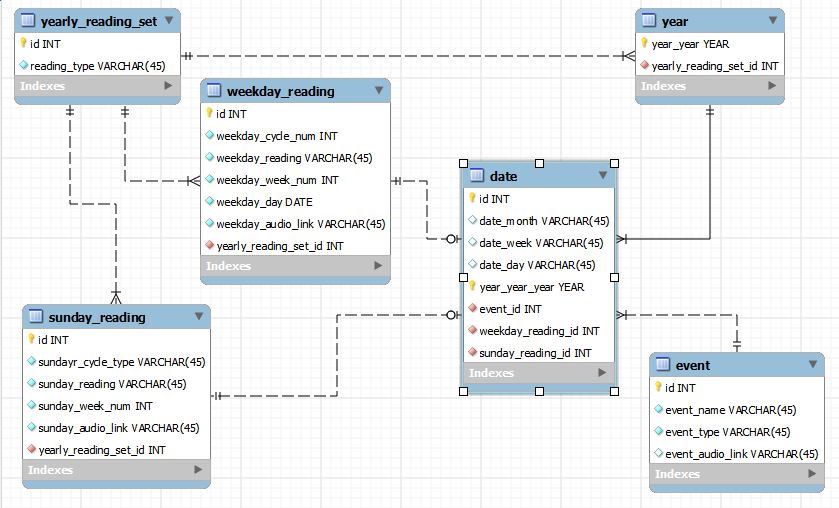
• The Physical Database Description

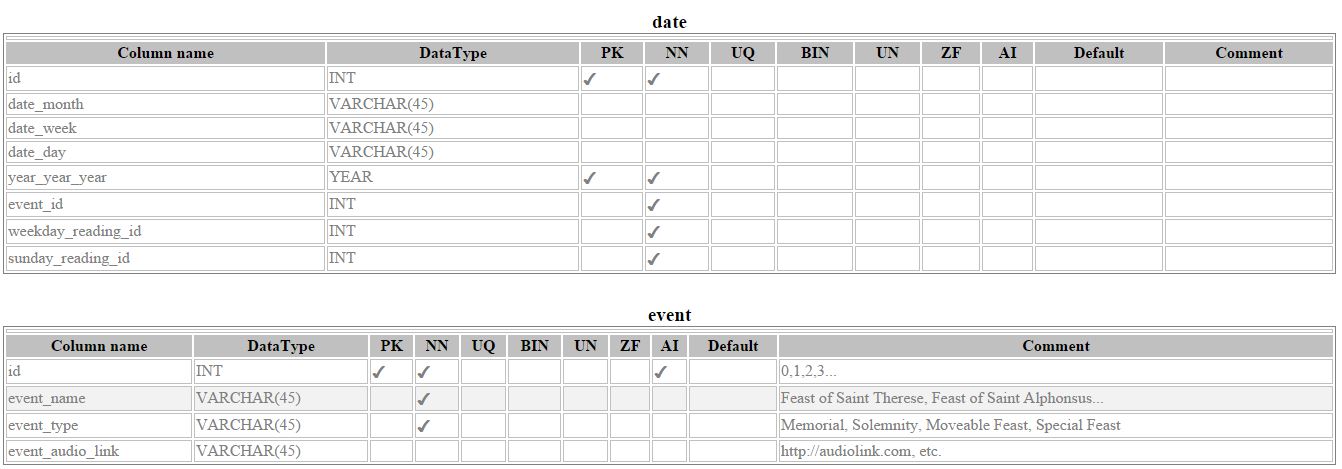
• The Software Design Document

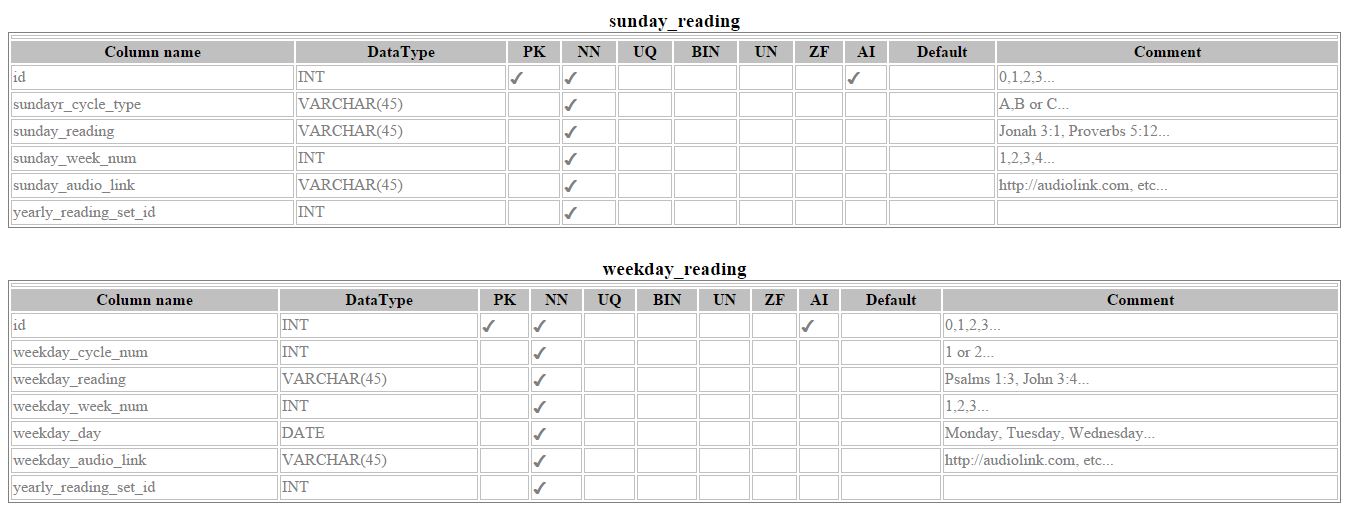
• The Requirements Traceability Report

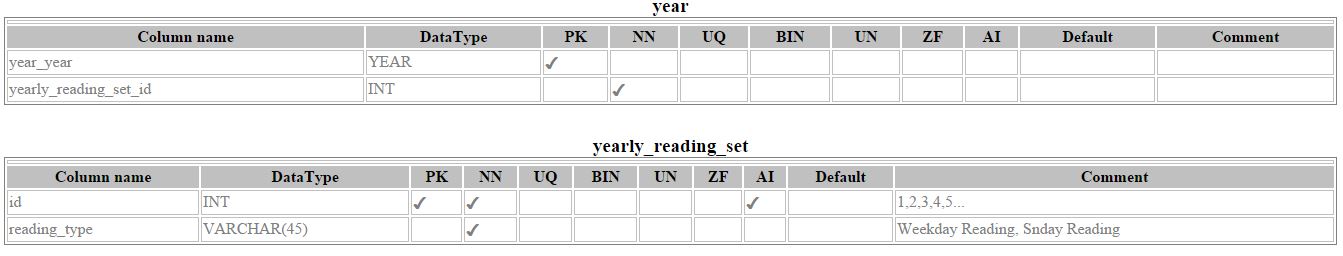
1. **Design Content**
   1. **Physical Database Description (PDD)**

Image 1.5 **Entity Relationship Diagram**

****

** DATA DICTIONARY**

**

**

*The physical database description defines the basic structure of the application at a conceptual level. The PDD focuses on providing a detailed description of the database structure to be implemented for the application.*

*The PDD consists of an introduction, an Entity Relationship Diagram (ERD) and a series of table and field descriptions that define the relationships between the entities, field characteristics, and business rules.*

*The PDD is included by reference in the Design Document.*

* 1. ****Software Design Document (SDD)**

**

Image(s) 1.6 **Swim Lane Diagram**

* 1. **Requirements Traceability Matrix (RTM)**

*The RTM makes use of the analysis listings in the SDD and its parent SRD. The purpose of the RTM is to show that each design element is related to a specific requirement in the SRD, that all goals in the project plan have at least one associated requirement, and that no requirements in the SRD are related to non-existent goals.*

1. **Online Help**
2. **Implementation Map**
3. **Test Plan**

This Software Test Plan was defined for the purpose of documenting the test procedures, test cases, and test steps required to validate the development effort.

1. **Background & Introduction**

Projects done using the Agile Method of development, are necessarily given a requirement of passing a certain level of confidence that the system is working at par with the industry’s requirements. To be able to satisfy this requirement, testing must be done. Testing chosen specifically for this project involves various stages (i.e. Static, Dynamic Quality Assurance Testing, etc.) It is included in this project’s success factors that the project’s passing rate should be in line with the standards defined in this test plan.

Scheduled *Static* and *Dynamic Tests* have been given to the proponents of this project. Therefore, during the said intended schedules of testing, the team’s prototype for the current iteration will be assessed by an external Quality Assurance Team using generically independent standards to rate the deemed project.

In conclusion of the each test phase, evaluation by the external Quality Assurance Team will be duly noted and assessed. For static test results, correction to the specified documentation and files will be implemented as soon as the reception of errors / comments. For dynamic test results, a record of issues and a log of bugs must be specified to identify the specified change requests and fixes to be implemented by the team for the project. Succeeding Quality Assurance tests will be subject to the results of the previous static / dynamic test, in line with previous updates and corrections done by the team.

1. **Assumptions**

Before the basis of any assumption, the following dependencies are core concepts that may serve as the scaffolding of the project’s test phase:

* The system must have sufficed the following basic deliverables, some of which, have been declared in the project scope:
  + Basic document deliverables (i.e. SQAP, Test Cases, etc.) to be evaluated **(Static testing)**
  + A database that includes all the *religious events* (e.g. lections, feasts and celebrations) within the year **(Static testing)**
  + Create a tool for value-inclusion in the database **(Dynamic testing)**
  + Provide a calendar that represents lections and mass readings for each date **(Dynamic testing)**
* Developer and designer involvement in the project have been clarified
  + The roles and responsibilities of each member of the team have been defined and clarified
* The Gathering Requirements phase has been accomplished successfully, and the current phase includes designing and development

In testing the system, the following assumptions are considered and are expected:

* System requirements have been set, and conditions dependent on the system have been provided by the proponents
  + The testing plan, test cases and other necessary requirements for testing to proceed have been provided by the proponents
* An external Quality Assurance Team has been selected to review the selected deliverables
* A certain standard has been set to properly evaluate the said deliverables; preferably these standards are set in a scale that can be quantified through values, and duly evaluated through description

Assuming that all the dependencies and assumptions are satisfied, specified and scheduled testing phases may proceed. It is important to note that these testing phases are specifically contingent to the schedules set for the project’s evaluation.

1. **Test Items / Programs**

The proponents have prepared the following to properly establish the items that must be tested during the phases of Quality Assurance Testing:

* 1. **Documentation for Static Testing**
     1. Project Requirements Definition
     2. Project Logical Design
     3. Project Physical Design
     4. Database Design
     5. User Interface
     6. Software Program Logic / Code
     7. Software Error Handling
     8. Test Plan, Test Cases
     9. Overall Quality of Documentation
     10. Documentation Completeness
  2. **Calendar display**
     1. Calendar Interface
     2. Features
  3. **CRUD for data storage**
     1. Create capability
     2. Read capability
     3. Update capability
     4. Delete capability

1. **Features To Be Tested / Features Not To Be Tested**

The following features of the system must be tested and duly evaluated:

* 1. **Security of User Registration and Login**
  2. **Correctness and legitimacy of event display**
  3. **Completeness of events being displayed**
  4. **Readability of calendar events**
  5. **Usability of calendar links**
  6. **Dynamism of Administrator Privileges and CRUD**

1. **Approach**

The agile approach for development warrants a brief definition of the Quality Assurance Testing Phases of this project. For the currently defined phases, the following table illustrates the necessary information for the approach in each QAT phase:

|  |  |  |
| --- | --- | --- |
| Name | Approach | Methodologies Involved |
| Static Testing | * Documentation – oriented * Specification and Requirements Scrutiny | * Identification of requirements, targets and methods * Analysis of completeness of information necessary to proceed to development phase * Evaluation of documentation and conformity to standards in the industry |
| Dynamic Testing | * Project and prototype – oriented * Testing for Cases / Scenarios | * Identification of features involved in project * Verification of the project’s achievement of the required functionality through evaluation *via* the *selected standards* and quality assurance *metrics* |

Table X.X **Approaches for the Project’s Quality Assurance Testing Phases**

1. **Pass / Fail Criteria (Itemized List of expected output and tolerances)**

The succeeding table summarizes the specific criteria that may serve as the establishment of the necessary standards to properly evaluate and analyze the application’s verification of the required functionalities:

|  |  |  |
| --- | --- | --- |
| Feature | Margin of Validity | Margin of Fault |
| Security of User Registration and Login | * User is **registered with a viable username and password** of his/her choice * Username and password **works effectively** | * User **cannot be** **registered with a viable** username and password * Username and password **does not work effectively** |
| Correctness and legitimacy of event display | * The events being displayed are **in line** with the client’s requirements * The events being displayed are **identical** to the data presently stored in the database | * The events being displayed are **not in line** with the client’s requirements, and are variably different * The events being displayed are **not** **identical** to the data presently stored in the database |
| Completeness of events being displayed | * The data present in the calendar are **complete** and **conforms** to the populated data | * The data present in the calendar are **incomplete** and **does not conform** to the populated data |
| Readability of calendar events | * The events displayed in the calendar are **readable** and are **recognizable** * The data **brings clarity** to the prospect users | * The events displayed in the calendar are **not readable** and are **unrecognizable** * The data causes the prospect users to be **misled** |
| Usability of calendar links | * The links **properly redirect** the user to the necessary audio files / text files / websites | * The links are **intertwined** and **does not follow** the intended file/page to be redirected to. |
| Cogency of Administrator Privileges and CRUD | * Administrator privileges are **well-defined** * CRUD **performs** required functionality | * Administrator privileges and user privileges are **not defined** * CRUD **does not** **perform** required functionality |

Table X.X **Pass / Fail Criteria**

1. **Test Deliverables**
2. **Testing Tasks / Setup**
3. **Environmental needs**
4. **Responsibilities**
5. **Staffing and Training**
6. **Schedule**
7. **Resources**
8. **Risks and Contingencies**
9. **Approvals**
10. **Deployment Plan**
11. **Acceptance Plan**
12. **Installation & Acceptance**