**1. Test Plan Identifier**

**File Name:** 1.0\_VASOP\_tplan\_v1.0.doc

**Location:Bangalore**

**2. Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **REV** | **Author** | **Edits** | **Date** |
| 1.0 |  | Initial Draft |  |
|  |  |  |  |
|  |  |  |  |

**3. Intorduction**

VAS (Vidyala Automation Systems) is software product developed by INFIX SOLUTIONS for addressing various software needs of the Hospitals.

This test plan will outline the features to be tested as part of outpatient details in the 1.0 release of VAS.

4. **Features to be Tested**

Medium priority features are features that have been stable and High Priority features are features that have been recently changed.

|  |  |  |
| --- | --- | --- |
| **Feature To be Tested** | **Priority** | **Comments** |
| Registration | High |  |
| Renewal | High |  |
| Lab details |  |  |
| X-ray details |  |  |
| Scan details |  |  |
| Abdominal Report |  |  |
| Obsteric Scan Reports |  |  |

1. **Features NOT to be Tested**

Help Contents will not be tested.

Installation and presentation will not be tested.

Support of the other architectures also will not be tested.

1. **Test Strategy**

The test strategy for VASOP1.0 will include manual testing with automated testing where time/need permits. Automated testing will be used to test the basic Interface objects, keyboard and mouse navigation or any redundant tasks best translated to automation. Automated testing will be limited to a Win-32 platform because of Automation tool limitations.

* Sanity test will include an very short battery of tests to ensure the VASOP1.0 runs/launches and can connect to the back end database. Test execution will happen as individual component testing. These tests will be fully automated.
* Acceptance tests will include a short battery of tests that will only touch major features to ensure VASOP1.0 will operate smoothly at a basic level. These tests will be executed on Windows 98, Windows NT 4, Windows 2000, workstations and servers. The goal will be to automate all of these tests.
* Full Functional tests will include a comprehensive test of all features listed in this document. These tests will be limited to the Windows 2000 workstation client. These test will include automation with time allowing.

**Functionality Testing**

**Registration**

Verify New Registration

Verify Modification of the existing record

Verify Deletion of the existing record.

Verify Searching of existing of records.

Verify Refresh of the Data Fields.

Verify Closing of the registration window.

**Renewal**

Verify Renewal of existing records.

Verify Searching of existing of records.

Verify Refresh of the Data Fields.

Verify Closing of the registration window.

**Lab details**

**X-ray details**

**Scan details**

**Abdominal Report**

**Obsteric Scan Reports**

**Usability Testing**

The usability testing will be accomplished by verifying the information in each window are accurate. Menus, icons and toolbar functionality will be tested as applicable to the navigation and results panes. Multi Window Overlapping will be tested because product supports opening of multiple documents.

**Robustness/Reliability Testing**

Test the GUI for correct keyboard and mouse navigation, windows, menus, buttons, etc. These tests will include Keyboard only navigation of VASOP1.0, Mouse only navigation. Correctness and existence of warning/error dialog boxes. The correct/expected functionality of buttons, icons and menus.

**Stress Testing**

Not applicable

**7. Test Deliverables**

Test plan 1.0\_VASOP\_tplan.doc

Test Specification 1.0\_VASOP\_tspec.doc

Test Automation Plan TBD

Test Checklist TBD

Test Logs TBD

Test tools Scripts, Segue SILK Test Tool

**8. Testing Tasks & Resource Needs**

Use of MS ACCESS database.

Write Test Specification.

Write Test Automation plan.

Write Test Checklist.

Setup workspace.

Develop Automation script.

Minimum requirements for platforms:

Intel: Intel ~400mz, with 96mb\* of Ram for NT workstation client.

Intel ~400mz, with 64mb \*of Ram for 98 workstation client.

(\* Automation needs an additional 128MBs to function)

**9.Schedule**

Test Documentation

| Document | Development Time | | Sign-off Date | Dependencies |
| --- | --- | --- | --- | --- |
|  | Person-Days | Completion Date |  |  |
| Test Plan | 4 days |  |  | Functional Spec. Review |
| Test Specifications | 7 days |  |  | Testplan, Functional Spec, Review |
| Test Logs | TBD |  |  |  |
| Test Summary Report | TBD |  |  |  |
| Checklist | TBD |  |  |  |

Test Development

| Deliverable | Development Time | | Dependencies |
| --- | --- | --- | --- |
|  | Person-Days | Completion Date |  |
| Functional Test Suite | 5 |  |  |
| Automation build |  |  |  |
|  |  |  |  |

Test Execution

| Test Activity | | Execution Time (Person-Days) | | | Dependencies |
| --- | --- | --- | --- | --- | --- |
|  | | Alpha | Beta | FCS |  |
| Acceptance | | 3 Days | 3 Days | 4 Days |  |
| Functional | Functionality | 5 | 4 |  |  |
|  | Usability | All High to Medium testcaces |  |  |
|  | Robustness |  |
|  | Stress |  |
|  | Multi Inst. |  |
| Regression | |  |  |  |  |
| Documentation Review | | 2? | 2? | 2? |  |
| Total Test Cycle | | 8 | 8 | 6 |  |

**10. Issues/Risks and Contingencies**

| RISK | CONTINGENCY | COMMENTS |
| --- | --- | --- |
| Any Automation will need to be created in parallel of testing. | Manual testing | Goal is to add Automation where possible.  Initial setup might be able to be worked on in an earlier stage if time permitting. |
| VB Accessibility Utilities and specs to verify compliance to Accessibility are not available. | Simplistic tests? | Accessibility guidelines are not specified in spec. |
|  |  |  |
|  |  |  |

**11.Approvals**

| POSITION  \* Approval Required | NAME | SIGNATURE | REVIEWED | APPROVED |
| --- | --- | --- | --- | --- |
| Tech Lead |  |  |  |  |
| STAFF ENG, SW |  |  |  |  |
| Project Manager |  |  |  |  |
| Clint Approval |  |  |  |  |

12. **References**

VAS1.0 User Interface Specification

VAS1.0 Functional Specification

VAS1.0 Design Specification

**Daily Status Report**

**The information that needs to be a part of an individual’s “Daily status report” is:**

1. What did you do today?
2. What are you planning to do tomorrow?
3. Did you face any issues during your day? If yes, how did you resolve them or are they still open?
4. Do you need any inputs for tomorrow? If yes, from whom and what are they?

The recipient of this email/report is generally the manager, also the team members can be CC’ed in some cases – this depends on the communication protocol the team follows.

**Test Reports**

Now, it’s time to get specific and learn all about the reports that Testing/QA teams send.

**Testing teams send out various reports at different phases in the STLC.**

* Test plan status
* Test documentation status
* Test execution status(defect status)

[**Test Plan**](http://www.softwaretestinghelp.com/test-plan-sample-softwaretesting-and-quality-assurance-templates/): It is enough to communicate with the rest of the project teams, when a test plan is created or when a major change is made to it.

[**Test documentation**](http://www.softwaretestinghelp.com/why-documentation-is-important-in-software-testing/) – Let all the teams know when the designing of the tests, data gathering and other activities have begun and also when they are finished. This report will not only let them know about the progress of the task but also signal the teams that need to review and provide signoff on the artifacts, that they are up next.

**Test execution**– Execution is the phase of a project when the testing team is the primary focus – positively and negatively – we are both the heroes and the villains.

A typical day during a test cycle is not done, unless the daily status report is sent out. In some teams, they could agree on a weekly report, but having it sent daily is the norm.

It is also not uncommon to have a status meeting everyday (or week) to present the QA team’s status to the concerned parties.

**Hence, the mode of a status report can be:**

1. Email/document
2. Meeting/presentation
3. Both – daily email and weekly meeting or so.

**Test Execution Status Report**

**Daily/Weekly Test Execution Report:**

**What is it?** Generally, this is a communication sent out to establish transparency to the QA team’s activities of the day during the test cycle – includes both defect information and test case run information.

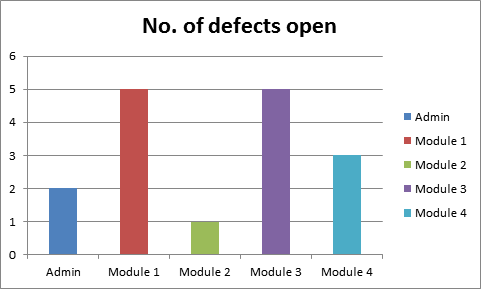
**Who should it go to?**  – Normally, Development team, Environment support team, Business analyst and the project team are the recipients/meeting participants. The Test plan is the best place for you to find this information.

**What does a test execution status report contain? – 10 points**

1. Number of test cases planned for that day
2. Number of test cases executed – that day
3. Number of test cases executed overall
4. Number of defects encountered that day/and their respective states
5. Number of defect encountered so far/and their respective states
6. Number of critical defects- still open
7. Environment downtimes – if any
8. Showstoppers – if any
9. Attachment of the test execution sheet / Link to the [test management tool](http://www.softwaretestinghelp.com/category/test-management-tools/) where the test cases are placed
10. Attachment to the bug report/link to the defect/test management tool used for incident management

– Shows the overall status with a color indicator. Eg: Green – on time, Orange- Slightly behind but can absorb the delay, Red- Delayed.  
– Include some simple metrics like: Pass % of test cases so far, defect density, % of severe defects; by doing this you are not just giving numbers, you are actually providing a glimpse of the quality of the product that you are testing.  
– If a significant phase is complete- highlight that.  
– If there is a critical defect that is going to block all/a part of future execution- highlight that.  
– If using a presentation, make sure to include some graphs to make a better impact.

For example, the below graph is a representation of **the number of open defects, module-wise**:

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2013/09/Open-Defects-Report.png)

**Apart from these, you can also optionally include:**  
– What are the activities planned next?  
– Do you need any inputs from any of the other teams and if so, what?

**Lastly, few pointers to help the process along:**

1. Be concise at the same time complete
2. Make sure the results you report are accurate
3. Use bulleted points to make the report very readable
4. Double check to include the right date, subject, to list and attachments.
5. If the report is too big and has too many factors to report: place it in a common location as a file and send a link in the email instead of the file itself.(Be sure the recipients have access permissions to this location and the file)
6. If it is a status meeting – Be prepared for the presentation, arrive on time and most importantly, maintain an even tone (don’t be too proud of the defects – they are in general “bad news”).

**Sample Status Report**

**QA Testing Status Report:**

Following these guidelines, we arrived at the below status report.

For the convenience of our readers we have included 3 sheets conveying different levels of information that they can convey.

**Sheet 1** – is like a summary of the overall status of the project.  
**Sheet 2** – is more about the individual detail of the test cases status.  
**Sheet 3** – is a sample bug report.

We are testers – in other words bug finders. Defect/Bug/Issue/Fault/Failure/Incident – whatever we choose to call – our primary job description revolves around finding, recording, reporting, managing and tracking these.

While there is no harm in using an excel sheet to record/track and emails to report/alert/communicate – as the magnitude of the projects, the number of test cycles, the count of the people involved grows – it becomes absolutely important that **we need a much stronger mechanism that will make the management of these issues simpler and consistent** so we can concentrate harder on actually [finding more issues in the AUT](http://www.softwaretestinghelp.com/how-to-find-a-bug-in-application-tips-and-tricks/) than managing the ones already found.

To enable the same, the QA market has seen the emergence of various bug tracking systems or defect management tools over the years.

As is the general rule, all the tools that belong to a certain ‘genre’ consist of certain common/similar features that we can bank on.

**For a bug tracking software it is essential to have:**

1. [**Reporting facility**](http://www.softwaretestinghelp.com/how-to-write-good-bug-report/) – complete with fields that will let you provide information about the bug, environment, module, severity, screenshots etc.
2. **Assigning** – What good is a bug when all you can do is find it and keep it to yourself, right?
3. Progressing through the [life cycle stages](http://www.softwaretestinghelp.com/bug-life-cycle/) – Workflow
4. History/work log/comments
5. **Reports** – graphs or charts
6. **Storage and retrieval** – Every entity in a testing process needs to be uniquely identifiable, the same rule applies to bugs too. So, a bug tracking tool must provide a way to have an ID, which can be used to store, retrieve (search) and organize bug information.

The above mentioned are the features of essence – which means these are absolutely necessary for any system that claims to be a bug tracking system. Apart from that, there might be additional features of convenience – like watching, saving searches etc., and some of assurance – like voting, showing the bug info in a live stream and so on.

While features of convenience and assurance are ‘nice to have’ it is the features of essence that become the game-changers during evaluation and making a choice as to what tool to use.  Then, there is economics to consider too.

We know that the tools available in the market are innumerable – with some of them being a perfect fit for you and the others that just won’t cut it. The remainder of this article is primarily going to focus on some of the crème de le crème of the bug tracking tools available and introduce you to them briefly.

**15 Most Popular Bug Tracking Software**

*Here goes:*

**1. Bugzilla:**



**Details:** Bugzilla has been a leading bug tracking tools widely used by many organizations for quite some time now. It is very simple to use, web based interface. It has all the features of essence, convenience and assurance. It is completely open sourced and is free to use.

**2.** [**JIRA**](http://www.softwaretestinghelp.com/atlassian-jira-tutorial-1/)**:**



**Details:** Atlassian JIRA, primarily an incident management tool is also commonly used for bug-tracking. It provides the complete set of recording, reporting, workflow and other convenience related features. It is a tool that integrates directly with the code development environments thus making it a perfect fit for developers as well.  Also, due to its capability to track any and all kinds of issues, it is not necessarily concentrated to only software development industry and renders itself quite efficiently to help desks, leave management systems etc. It supports agile projects also. It is a commercial licensed product with many add-ins that support extensibility.

**3. Mantis:**



**Details:** I have one thing to say about this tool – do not be deceived by its simple exterior. I mean, in terms of simplicity and ease of use, this tool wins the crown. It has every feature you can hope for and then some. To catch up with the changing times, Mantis not only comes as a web application, but also has its own mobile version. It is implemented in PHP and is free for use. If you would like it to be hosted, they do charge a price, but quite affordable, I must say.

**4. Trac:**



**Details:** Trac also is not necessarily a specialized bug tracking system and is an issue tracking system. It is written using Python and is web based. When you integrate Trac with a SCM system, you can use it to browse through the code, view changes, view history etc. The issues/incidents in Trac are referred to as ‘tickets’ and the ticket management system can be used for defect management as well, if you wish to do so.

**5. Redmine:**

Redmine issue tracking

**Details:** Redmine is an open source issue tracking system that integrates with SCM (source code management systems) too. Even though it is not a ‘bug tracking’ tool it involves working with issues, where issues can be features, tasks, bugs/defects etc. It is a web application that works across many platforms, but will need Ruby to be available. For more info, check out: [Redmine details.](http://www.redmine.org/)

**6.** [**HP ALM/Quality Center**](http://www.softwaretestinghelp.com/learn-hp-quality-center-qc-in-4-days/)**:**



**Details:** Well, no list of bug tracking tools will be complete without the HP QC, would it? HP ALM is an end-to-end test management solution with a robust integrated bug tracking mechanism within it. HP ALM’s bug tracking mechanism is easy, efficient and everything you can ask for. It supports Agile projects too. It is one of the pricey tools available in the market, which continues to be a prime source of criticism along with the fact that it is not very ‘friendly’ with all the web browsers.

**7. FogBugz:**



**Details:** FogBugz is also a web based bug tracking system that refers to defects as ‘cases’. It allows you to create, list, assign and work on cases created. Also the project information can be created in terms of milestones so that the progress of the cases can be evaluated against the milestones. Very simple to use and has all the features of essence for sure. Additionally, with FogBugz you can create wikis to be made available for the general public. It is a commercial product but very reasonably priced.

**8. IBM Rational ClearQuest:**

IBM Rational ClearQuest logo

**Details:** Clear Quest is a client-server based web application that supports defect management process. It provides integration with various automation tools which can be considered an additional feature. Other than that, it has an end-to-end, customizable defect tracking systems. It is a commercial product and can seem a little costly. You can try it free for 30 days.

**9. Lighthouse:**

Lighthouse defect tracking logo

**Details:** Light house is an issue tracker that is web based and is also compatible with your mobile devices. It is simple and organized. All the issues are referred to as tickets in here too. There is an activity stream, mile stones etc. Another nice feature is that, lighthouse lets you store project document online in its interface itself.

**10. Zoho bug tracker:**



**Details:** Zoho Bug Tracker is one of the modules in the task management software Zoho Project. It is an online tool that will let you create Projects, milestone, tasks, bugs, reports, documents and so on. The bug tracker module by itself has all the features of essence that you generally look for. The product is commercial but not very expensive.

### 11. The Bug Genie:

The Bug Genie logo

**Details:** Though the name sounds like it must be a bug-tracking tool – that is not all Bug Genie is. It is a complete Project management and issue tracking tool – which involves defect management to be one of its aspects along with integration with many SCM systems, Project creation and handling features, issue tracking mechanism, integrated wiki and easy to use web interface. Supports Agile projects also .

### 12. BugHost:

BugHost logo

**Details:** A web-based defect tracking system that is very simple and has all the features that you will need to manage issues for your project effectively. It also has a nifty little service WebHost that you can use for the users (the end customers) to create an issue directly into your project. Though commercial, it is very affordable.

### 13. DevTrack:

Devtrack logo

**Details:** Devtrack cannot be categorized as your average defect tracker although it does function well if that is what you have in mind. It can be obtained as a stand-alone component or it comes along with Agile Studio, DevTest studio or the DevSuite. As the name implies it is a comprehensive solution to implementation tracking. Supports both agile and waterfall projects. It is a commercial product. A free trial is available too.

### 14. BugNET:



**Details:** BugNET belongs to the “issue management” group of tools – quite a good one at that. The issues could be features, tasks or defects. It has all the features of creating projects, managing them, creating issues against them and tracking them to completion, search, reports, wiki pages, etc. There is a pro version for this tool that is licensed and commercial, but the regular version is free to use.

### 15. eTraxis:



**Details:** eTraxis is also another tracking tool, that can be used to track bugs but again, that is not all. You can choose to track basically anything. So, the target audience is not confined to software systems. The best feature of this tool is the flexibility it provides with regards to the creation of custom workflows- in other words, you can choose to define the rules that need to be followed in the process of tracking and progress-ing a certain aspect through its lifecycle stages. These custom workflows are referred to as templates and they can be very handy.

The above 15 are the most widely used bug tracking tools, although you might have noticed that bug-tracking by itself is not the target of many tools. This is because of the fact that, defect management rarely makes sense when it is an isolated activity from the entire project related aspects. Therefore, bug management becomes a part of these tools but is not all there is to them.