Telerik Academy Learning System

Test plan

Project URL:

***http://stage.telerikacademy.com/***

*Document Revision History*

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Comment |
| 23.11.2015 | 1.0 | I. Havalyova | Initial version. |
| 24.11.2015 | 1.1 | D. Dechev | Added stress testing |
| 25.11.2015 | 1.2 | D. Panayotov | Added performance Testing part. |
| 25.11.2015 | 1.3 | Y. Yosifov | Formatting changes. Added Functional Testing part. |
| 26.11.2015 | 1.4 | I. Havalyova | Added Load testing. |
| 26.11.2015 | 1.5 | I. Zhelev | Added Regression testing. |
| 27.11.2015 | 1.6 | Y. Yosifov | Text cleaning. |
| 27.11.2015 | 1.65 | D. Dechev | Week 1 final draft |
| 03.12.2015 | 1.7 | D. Dechev | Revision after product Owner feedback |
| 03.12.2015 | 1.75 | I. Havalyova | Formatting |

Table of Contents

1. Introduction4
2. Testing strategy4

Objective4

*Test Assumptions* *4*

1. Test criteria5

Entry criteria 5

*Exit criteria 5*

1. Scope5

*Functions to be tested*6

Admin part 6

User part 6

1. *Risk Analysis****6***
2. *Training needs****6***
3. *Test strategy****7***

*Unit testing7*

*Integration testing7*

*System testing7*

*Acceptance testing8*

*Functional testing8*

*Performance testing8*

*Load testing9*

*Stress testing9*

*Regression testing9*

1. *Schedule****10***
2. *Defect tracking****10***
3. *Deliverables****10***
4. *Communication Approach****11***
5. *Environmental needs****12***

*Hardware12*

*Software12*

Tools*13*

1. **Introduction**

Telerik Academy Learning System is student system developed for the purpose of aiding training at the Telerik Software Academy. It is built and maintained by the Academy of Telerik centralized information system, which stored information about training courses, participants in the Telerik Academy, their results in the courses (homework, projects, assessments and achievements), schedules classrooms, courses and workshops, and other information related to teaching and activities in the Academy. This plan is part of a master plan (insert reference here) aiming to assure the quality of the system in question. The current document is focused on the functionalities in article 4 of the current document. This document is focused on system testing of the system, because the system is currently at that level of development.

1. **Test Strategy**
   1. **Objective**

Objective of this test plan is to ensure that the functionalities of the Telerik Academy Student System listed in article 4 of this document are tested. The top priority functionalities will be “Login” and “Registration”. Medium priority will be the “Courses enrolment”, “Access to lectures/presentations/home works” and “Changing the form of training attendance/online”. Low priority are “Friends” and “Messages”. Users must be able to register and login with a user and not to be able to login or change another user account. Users must have the opportunity to enroll to a course and access all the courses resources and eventually switch from onsite to online education. Users must be able to invite other users to be friends and if the friendship is accepted they must be able to exchange messages.

* 1. **Assumptions**

The web application to be tested is located at <http://stage.telerikacademy.com/> and is in complete code freeze, ready for testing. The testing is going to be performed by five independent teams of quality assurance trainees. Each of the teams will have responsibility to test a set of functionalities. The cost of testing will be the cost of the trainer staff and the infrastructure to perform the training. The tools used will be either open source or Telerik products.

1. **Test criteria**
   1. **Entry criteria**

The testing will begin when there is testing environment available at <http://stage.telerikacademy.com/> . The code must be unit tested by the development team. Hardware and software with the requested tools must be provided to the testing team.

* 1. **Exit criteria**

The objectives of this plan will be fulfilled when:

* 100% of functional tests of any priority pass
* At least 90% of the high priority non-functional tests pass
* At least 75% of the medium priority non-functional tests pass
* At least 50% of the low priority non-functional tests pass
* No blocking bugs are unresolved

1. **Scope**

|  |  |
| --- | --- |
| Functions to be tested by the team | |
| Users | Admin |
| Changing the form of training - attendance / online | Front |
| Access to lectures / presentations / home works | Front |
| Courses enrollment | Front |
| Registration | Front |
| Log in | Front |
| Friends | Front |
| Messages | Front |

1. **Risk Analysis**

The following risks have been identified and the appropriate efforts to mitigate the impact on the project are identified. The impact is a relative – High, Medium and Low.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Risk | Impact | Probability | Risk Owner | Mitigation plan |
| 1 | Ability of the team members to perform the Quality Assurance tasks. | High | Low | Product Owner | Weekly training, Monthly control exams |
| 2 | Insufficient team organization | Medium | Medium | Team Leader | Weekly review of team results by Product Owner |
| 3 | Binary files merge conflicts | Medium | Medium | Team Leader | Work on binary files will be organized by the team leader |

Major project risk is the lack of organization and lack of competence from the test team members. As trainees in the Telerik QA Academy the team has little to no previous experience in the area. The team will be self-organized so time pressure and lack of organizational skills will be another risk. In order to mitigate these risks weekly reviews of each team’s progress will be performed and guidance will be provided by the trainers.

1. **Training needs**

The test team has limited experience in software testing. Through the time period this plan covers there will be three, four hour long, training sessions each week. The purpose of the training is to familiarize the trainees with the test process and the tools that are used in software quality assurance. On monthly basis there will be exams measuring the progress of each individual team member.

1. **Test strategy** 
   1. **Unit testing**

Unit testing is already performed by the development team.

* 1. **Integration testing**

Integration testing is already performed by the development team.

* 1. **System testing**

System testing is the main focus of this plan. Black box functional and non-functional testing will be performed

* 1. **Acceptance testing**

Will be performed when system testing is done and product requirements are clarified with the stakeholders.

* 1. **Functional testing**

All of the following stuff should be tested and verified:

* **Links**
* **Forms**
* **Cookies, sessions**
* **Passwords, Permissions, Roles**
* **User interface**

For detailed information see the Appendix 1.

* 1. **Performance testing**

The top priority functionalities to be tester for performance issues and bottlenecks are “Access to lectures/presentations/home works” and “Messages”. The functionalities that invoke the database for access to its content as the messaging system and the database of all the lectures, presentations and home works must be optimized to not slow down the system, especially in cases where multiple user are accessing the at the same time. The “Friends” management system is not as performance dependent as the top priority ones but in a future state it may also generate a large amount of data to be transferred if the user list manages to grow large so it should also be specifically tested to ensure adequate future behavior.

* 1. **Load testing**

Telerik Learning System will be tested for determining the maximum number of concurrent users that the application server can support under a given configuration. We will watch and measure the response time of application and determine whether it is in acceptable limits. We will analyze and present the load test results at the conclusion of the performance test cycle.

***The areas of testing are:***

* Access to lectures / presentations / home works with login
* Courses enrollment with login

Access to lectures / presentations / home works and Courses enrollment will be tested with many regular users (development team and the product owner will be involved in providing a plausible estimation of the load) and results will be analyzed. The main purpose of this type of test is to ensure the application can sustain acceptable levels of performance over an extended period of time without exhibiting degradation, such as might be caused by a memory leak or server crash.

* 1. **Stress testing**

Stress testing of all functionalities will be performed. The behavior of the application will be tested with the database and/or web server offline. The user should be redirected to page with appropriate messages.

* 1. **Regression testing**

Тhe purpose of doing regression test is to ensure that all currently implemented functionalities are still fully functional and no new bug are introduced, weather a new functionality is added or an old issue is fixed. This test will be done again after another bug have been found and fixed. This will be done by reusing the automated test suite written in order to ensure the quality of the currently implemented functionalities.

1. **Schedule**

There is time allocated for the completion of this test plan in the master plan document (reference here). The work will be completed in three months.

|  |  |  |
| --- | --- | --- |
| Test Phase | Time | Owner |
| Test plan creation | 1 week | Test team |
| Test case specifications | 1 week | Test team |
| Test case automation with Telerik Test Studio | 3 days | Test Team |
| Test case automation with Telerik testing framework | 4 days | Test Team |
| Test case design with test design techniques | 5 days | Test Team |
| Test automation with Sikuli | 4 days | Test Team |
| Test design for functional mobile testing | 3 days | Test Team |
| Web service test design and implementation | 5 days | Test Team |
| Security test design and implementation | 4 days | Test team |

The team will use the scrum software development model. There will be one week-long iterations and test design implementation and execution will be performed in each iteration. Each iteration schedule will be discussed and accepted by the team on the weekly sprint planning meeting each Monday at 20:10.

1. **Deliverables**

|  |  |  |
| --- | --- | --- |
| Deliverable | For | Date |
| Test Plan | Product owner | 27.11.2015 |
| Revised test plan | Product owner | 04.12.2015 |
|  |  |  |

1. **Defect tracking**

Telerik Team Pulse will be used as a bug tracking system.

A severity classification is based on the degree of the error impact on the system. The severity classification is as follows:

|  |  |
| --- | --- |
| Severity code | Description |
| * + 1. Critical | The defect causes a failure of the complete software system. The test process cannot proceed further. |
| * + 1. High | The defect does not cause a failure of the system, but causes the system to produce incorrect results when valid information is used as input in test cases. |
| * + 1. Medium | The defect does not cause a failure of the system. The test case or procedure can be completed and produces correct output when valid information is input, but produces incorrect output when invalid information is input. |
| * + 1. Low | The defect does not impact the functionality and usability of the system. All written test cases passed and produce expected results, but there could be minor revisions and small changes. |

A priority classification of a software error is based on the importance of resolving the error. The priority classification is as follows:

|  |  |
| --- | --- |
| Priority code | Description |
| Immediate | The bug should be resolved immediately in the current release. |
| High | The bug should be resolved in the current release after all critical bugs are resolved. |
| Medium | The bug should be repaired after all more important bugs (with immediate and high priority code) are resolved in the current or in the next release. |
| Low | The bug should be fixed only after all other defects are fixed. There is no future time limits. |

1. **Communication Approach**

The team will use the scrum development process. Weekly planning meetings will be held at 20:00 on Monday. The standup daily scrum meetings will be held on Facebook at 22:00 every week day except Monday and Friday. There is a Github organization and Facebook group set up in order to facilitate communication and collaboration in the team.

* 1. **Team members**

|  |  |  |
| --- | --- | --- |
| Name | User | Role |
| Asya Georgieva | N/A | Product Owner |
| Decho Dechev | decho.D | QA Test Team |
| Dimitar Panayotov | The.Bager | QA Test Team |
| Ilvie Havalyova | havaliova | QA Test Team |
| Yane Yosifov | Curiosity | QA Test Team |
| Ivan Zhelev | Ivan.jelev.16 | QA Test Team |

1. **Environmental needs**

* ***Hardware***
* **CPU**: Intel Xeon CPU @ 3.10 GHz (Quad Core)
* **RAM**: 8GB
* **Storage**: 100GB
* ***Software***
* **OS**: Windows Server 2012 R2
* **Web Server**: MS IIS 8.5
* **SQL Server**: MS SQL Server 2012
* **Tools**
* **Defect tracking system**: Telerik Team Pulse
* **Source control system**: Git
* **Test case management system:** Excel,[Visual Studio Team Services](https://teamlich.visualstudio.com/)
* **Testing tools:**
  + Telerik Test Studio
  + Telerik Testing Framework
  + Visual Studio 2013 Ultimate, Visual Studio 2015

**Appendix 1**

**Functional testing**

***Links***

* *Test all internal links.*
* *Test links used to send message to admin or other users.*
* *Check for orphan pages.*
* *Check for broken links.*
* *Check links for indication of a link action that can be performed or was performed (i.e. color different from normal text).*

***Forms***

* *Check validations on each field.*
* *Check for the default values of fields.*
* *Confirm inputs and boundary values for all fields.*
* *Confirm options to create forms if any, delete, view or modify the forms.*
* *Check different user types and roles.*
* *Data*
* *Check data accuracy and integrity*
* *Check data input validation*

***Cookies and sessions***

* *Test enabling or disabling of cookies in different browsers’ options.*
* *Test for cookie encryption before writing to user machine.*
* *Check for login sessions and user statistics after session end.*
* *Confirm effect on application security by deleting the cookies.*
* *Check if the session is recorded during correct time.*
* *Test for application robustness for rejecting cookies.*
* *Check if the session cookies or persistent cookies are used appropriately.*

***Passwords, Permissions and Roles***

* *Password should be displayed as asterisk or other masking.*
* *Password should be encrypted in source code.*
* *Change password should be take effect after login in the same computer.*
* *Password should not be able to be copied/pasted.*
* *Password security rule check (password strength): length, letter & number combination etc.*
* *Each user should have correct privileges as per their role to execute tasks and access information.*

***User Interface***

* *All components should be displayed in the correct location even changes in page sizes.*
* *Check for overlap or truncation for each component.*
* *Fonts and colors should be clear and easy to discern.*
* *Check for garbage characters.*
* *Check different resolutions to ensure that all application components display correctly.*
* *Check changing window sizes and ensure all application components display correctly.*
* *Check supported browser/OS to ensure that all application components display correctly.*