Non Functional Requirements

Project Underground – Web Applications and APIs

|  |  |
| --- | --- |
| Author | David Gladstone |
| Status | Draft |
| Date | 9 February 2017 |

Table of Contents

[1 Change log 3](#_Toc474750562)

[2 Non-functional requirements 4](#_Toc474750563)

[2.1 Security 4](#_Toc474750564)

[2.2 Deployment 4](#_Toc474750565)

[2.3 Logging and diagnostics 4](#_Toc474750566)

[2.4 Quality and test automation 5](#_Toc474750567)

[2.5 Frameworks 6](#_Toc474750568)

[2.6 Responsive design 6](#_Toc474750569)

[2.7 Accessibility 6](#_Toc474750570)

[2.8 Performance 6](#_Toc474750571)

# Change log

|  |  |
| --- | --- |
| Date | Purpose |
|  |  |

# Non-functional requirements

This section summarises non-functional requirements for the implementation of the solution.

## Security

Access to all systems and interfaces require authentication of the user (or application) and all operations must be authorised. For example, a provider portal user must authenticate (prove identity) to login and must have the necessary authorisation to access data or interact with the core systems.

Client applications are responsible for ensuring that users are correctly authenticated and authorised to access the application.

APIs are responsible for ensuring that actions are only performed with correct authorisation and only authorised data are released.

### Use of TLS

All HTTP connections must be secured with TLS. Clients must always validate server certificates.

### Audit events

Certain activities must be recorded to an audit log. These activities to be discussed with nib but may include:

* Changes to providers
* Billing entities
* Procedures
* Logins

## Deployment

### Automation

All build and deployment steps should be automated.

### Outages

System deployments should be able to be performed during business hours without impacting users.

## Logging and diagnostics

The system must incorporate sufficient diagnostic logging capabilities to enable diagnosis of failures.

### Logging levels

For logs are consistently useful and manageable the following log level classifications should be used:

|  |  |
| --- | --- |
| Level | Use cases |
| Trace | * “I'm about to do something that has a low probability of failure.” * “I’ve done something trivial” * “Here’s more details about an action logged at a less verbose log level” - *including potentially sensitive data*. |
| Debug | * “I'm about to do something.” * “Here's summary details about an action logged at a less verbose log level.” - *including potentially sensitive data*. |
| Info | * Significant “I'm about to do something” messages that have a relatively high probability of failure. * Genuinely useful, *non-sensitive data*. |
| Warn | * Warning messages, *non-sensitive data*. |
| Error | * “Something went wrong, but it's not the end of the world.” |
| Fatal | * “Stop the world and let me get off.” |

### Sensitive data

Sensitive data should never appear in anything but Debug and Trace level messages. Under normal circumstances a production system will not be logging this level of detail and it allows for tight control over what is shipped to cloud-hosted log aggregation services[[1]](#footnote-1).

Sensitive data includes:

* Information that can identify an individual (names, email addresses, policy numbers, etc.)
* Passwords or other credentials
* Any payment related information (bank account numbers, credit card details)

### Log retention

Local log files should be removed automatically after a configurable number of days.

## Quality and test automation

### Integration testing

All user stories (i.e. Portal registration, lodging claims, etc) must be covered by automated integration tests.

* There should be at least one end to end test for each user story happy path
* There should be no more than one or two variations of each end to end test (i.e. no more than three end to end automated tests per user story).

### DOM event testing

Web applications must have DOM event tests against each page.

* These tests should be used to cover error conditions, exceptions, validation scenarios, etc. i.e. all cases (inputs/output variations) not covered by the end to end integration tests.
* Any server side interaction (e.g. ajax posts) needs to be stubbed

### Unit tests

All code (including C# and JavaScript) must have unit test suites that are adequate in terms of coverage and readability. In particular, the unit tests for each class must be written so that, as a suite, they clearly and concisely describe the *behaviour* of the class they are being written for. This can be accomplished using standard unit test best-practices (e.g. testing in isolation, use of BDD style naming conventions for tests, use of single Asserts for each test, etc)

### API testing

API client methods must have automated test scripts. This must be done in collaboration with the API providers (i.e. nib) using appropriate tooling (e.g. Swagger + Swagger Request Validator, PACT, etc).

### Manual Testing

Solution must be manually tested with test cases and results documented.

## Frameworks

All frameworks used to be approved by nib. These frameworks to be discussed and determined in collaboration with nib during the discovery phase. However, the following are currently in use in existing nib solutions:-

* ASP.NET MVC, .NET Framework 4.5.2, Ninject 3.2, Javascript, Knockout, .less used for the web application.
* Selenium WebDriver and SpecFlow used for the integration tests
* Jasmine-jquery used for the DOM event tests
* NUnit used for the unit tests

## Responsive design

All web pages must be responsive with design for tablet size as well as desktop.

## Accessibility

All web pages must be designed with accessibility in mind i.e. buttons, input fields and other gui elements must have shortcut keys associated with them, and be able to be accessed using the tab keys, arrow keys, etc.

## Performance

The following performance requirements must be met by the solution. Please note: these requirements exclude the calls made by the provider portal to the nib api and are based loosely on the current nib customer portal.

### Response times

98% or greater of all response times to all pages must be less than 8 seconds when measured over a two hour period.

### Load

Deployed on standard nib infrastructure, the provider portal must be able to support up to 500 active users a day and 600 sessions a day.

### Scalability

The provider portal must be able to support up to 50000 registered users.

1. Nib use SumoLogic as a cloud-hosted log aggregation service. [↑](#footnote-ref-1)