CS425 – Software Engineering

**Software Requirements Specification**

Aha

Version – 0.1

**25 July 2018**

TABLE OF CONTENTS

[1 OVERVIEW 3](#_Toc520293768)

[1.1 Purpose 3](#_Toc520293769)

[1.2 Business Objectives 3](#_Toc520293770)

[1.3 Scope 3](#_Toc520293771)

[1.4 Out of Scope 3](#_Toc520293772)

[1.5 Glossary 3](#_Toc520293773)

[1.6 Assumptions 3](#_Toc520293774)

[1.7 Constraints 3](#_Toc520293775)

[1.8 Dependencies 3](#_Toc520293776)

[2 USER REQUIREMENTS 4](#_Toc520293777)

[2.1 Actors 4](#_Toc520293778)

[2.2 Use Case Model 4](#_Toc520293779)

[2.3 Use Case Catalogue 6](#_Toc520293780)

[3 DATA DICTIONARY 13](#_Toc520293781)

[4 COMMON BEHAVIOURS 14](#_Toc520293782)

[5 NON-FUNCTIONAL REQUIREMENTS 16](#_Toc520293783)

[5.1 Definition of non-functional requirements 16](#_Toc520293784)

[5.2 User Interface 17](#_Toc520293785)

[5.3 External Interfaces 17](#_Toc520293786)

[5.4 Usability 18](#_Toc520293787)

[5.5 Performance 19](#_Toc520293788)

[5.6 Security 20](#_Toc520293789)

[5.7 Supportability 20](#_Toc520293790)

[5.8 Reliability 21](#_Toc520293791)

# OVERVIEW

This document will specify New Car Rental (NCR) system. In here include first the purpose of the system, its Scope and Objectives. Then detail of the system will be described through use case diagram and detail of each action.

## Purpose

The SRS should fully describe the external behaviour of the system. It also describes non-functional requirements, design constraints, and other factors necessary to provide a complete and comprehensive description of the requirements for the software.

## Business Objectives

The business objectives are to create New Car Rental system. The purposes of the system are:

1. Do not need to spend much money to invest in buying new cars at beginning.
2. Utilize unused or less-used cars to generate income for anyone those who are interested.
3. Create a new investment channel for anyone those who want to.
4. The car rental prices are much cheaper than the traditional car rental ones.

## Scope

* This project’s scope is to cover all use cases described on section [2.3 – Use Case Catalogue](#_Use_Case_Catalogue) of this document.

## Out of Scope

* N/A

## Glossary

|  |  |
| --- | --- |
| **Term** | **Description** |
| NCR | New Car Rental |
|  |  |

## Assumptions

|  |  |
| --- | --- |
| **Ref** | **Assumption** |
|  |  |
|  |  |
|  |  |

## Constraints

|  |  |
| --- | --- |
| **Ref** | **Constraint** |
|  |  |
|  |  |
|  |  |

## Dependencies

|  |  |
| --- | --- |
| **Ref** | **Dependency** |
|  |  |
|  |  |
|  |  |

# USER REQUIREMENTS

## Actors

|  |  |
| --- | --- |
| **Actor** | **Description** |
| System Admin | We do not need to spend much money to invest and buy cars at first. |
| Car Owner | Utilize unused or less-used cars to generate income for anyone who has those ones. |
| Customer | Create a new investment channel for anyone who wants to. |
|  |  |

## Use Case Model

### Use Case Model for System Admin



### Use Case Model Car Owner



### Use Case Model Customer



## Use Case Catalogue

### Car Owner Application Analysis

|  |  |  |
| --- | --- | --- |
| **Use Case Number**: 1 | | |
| **Name** CarOwner Application Analysis | | |
| **Brief description** This use case allows the Admin to manage Car Owner Applications | | |
| **Actors** Admin | | |
|  | | |
| **Preconditions** | | |
| The Admin must be logged in. | | |
| **Flows of Events:** | | |
| 1. Basic Flows | | |
| **1.1.0 Review and approve Profiles of Car Owner** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The admin selects to view pending Applications of Car Owner | The system returns a list of all pending Car Owner Applications |
| 2 | The admin selects the application that he wants to review | The system gets and displays details of selected Car Owner Application |
| 3 | The admin approves and saves the application |  |
| **Postconditions** | | |
| The Car Owner Application is updated and saved in DB | | |
| **Business Rules** | | |
|  | | |

### Car Offer Analysis

|  |  |  |
| --- | --- | --- |
| **Use Case Number**: 2 | | |
| **Name** CarOffer Analysis | | |
| **Brief description** This use case allows Admin to manage Car Offers | | |
| **Actors** Admin | | |
|  | | |
| **Preconditions** | | |
| The Admin must be logged in. | | |
| **Flows of Events:** | | |
| 1. Basic Flows | | |
| **1.1.0 Review and approve Profiles of Car** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The admin selects to view pending Car Offers | The system returns a list of all pending Car Offers |
| 2 | The admin selects the Car Offer that he wants to review | The system gets and displays details of selected Car Offer |
| 3 | The admin approves and saves the Car Offer | The system changes the state of the Car Offer and saves it in the system.  The system returns a success or fail of this action |
| **Postconditions** | | |
| The Car Offer is updated and saved in DB and the car is ready for rental | | |
| **Business Rules** | | |
| The car is up to 5 years old from the date is approved | | |
|  | | |

### Car Owner Application Process

|  |  |  |
| --- | --- | --- |
| **Use Case Number**: 3 | | |
| **Name** Car Owner Application Process | | |
| **Brief description** This use case allows Car Owner to apply for it, and edit its profile information | | |
| **Actors** CarOwner | | |
|  | | |
| **Preconditions** | | |
| The Car Owner must be logged in. | | |
| **Flows of Events:** | | |
| 1. Basic Flows | | |
| **1.1.0 Add a Car Owner profile** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The Car Owner calls Apply as Car Owner command | The system displays a Car Owner Profile form with fields: first name, last name, DOB, email, phone, address and bank account. |
| 2 | The Car Owner fills out and submit the form | The system validates required fields, verifies there is no other profile the same requested one, saves the profile with pending status in DB.  The system returns a success or fail. |
| **Postconditions** | | |
| The Car Owner Application is persistent in DB | | |
| **Business Rules** | | |
| There is no duplication of Car Owner profile: no duplication of email, bank account. | | |
|  | | |
| **1.1.1 Edit an existing Car Owner profile** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The Car Owner gets their profile | The system displays Car Owner Profile form a full info of first name, last name, DOB, email, phone, address and bank account. |
| 2 | The Car Owner selects to update fields they want to change, and submit the changes to the system | The system validates updated fields, saves the profile with pending status in DB.  The system returns a success or fail. |
|  |  |  |
| **Postconditions** | | |
| The system will save changes of Car Owner profile in DB | | |
| **Business Rules** | | |
| There is no duplication of Car Owner profile after changed: no duplication of email, bank account. | | |
| The Car Owner status will be changed to pending and waiting for Admin to approve. | | |

### Car Offer Registration

|  |  |  |
| --- | --- | --- |
| **Use Case Number**: 4 | | |
| **Name** Car Offer Registration | | |
| **Brief description** This use case allows Car Owner to add, edit and delete their cars | | |
| **Actors** CarOwner | | |
|  | | |
| **Preconditions** | | |
| The Car Owner must be logged in. | | |
| **Flows of Events:** | | |
| 1. Basic Flows | | |
| **1.1.0 Add a Car profile** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The Car Owner selects to register a new Car Offer | The system displays a Car Offer form with fields: car name, model, manufacturer, year of manufacturing, rental price. |
| 2 | The Car Owner fills out and submit the form | The system validates required fields, saves the car profile with pending status in DB.  The system returns a success or fail. |
| **Postconditions** | | |
| The Car Offers with its rental price is saved in DB | | |
| **Business Rules** | | |
| One Car Owner can add one or more Car Offers (same model, same manufacturer, etc) | | |
|  | | |
| **1.1.1 Edit an existing Car profile** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The Car Owner gets a list of their Car Offers | The system displays Car Offer list. |
| 2 | The Car Owner select a Car Offer they want to update | The system displays Car Offer form with a full info |
| 3 | The Car Owner selects to update fields they want to change, and submit the changes to the system | The system validates updated fields, saves the profile with pending status in DB.  The system returns a success or fail. |
|  |  |  |
| **Postconditions** | | |
| The system will save changes of Car Offer in DB | | |
| **Business Rules** | | |
| The Car Offer status will be changed to pending and waiting for Admin to approve. | | |
|  | | |
| **1.1.2 Delete an existing Car profile** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The Car Owner gets a list of their Car Offers | The system displays the Car Offer list |
| 2 | The Car Owner select a Car Offer that he wants to delete | The system asks for a confirmation to delete it. |
| 3 | The Car Owner confirms to delete | The system deletes the profile from DB.  The system returns a success or fail. |
|  |  |  |
| **Postconditions** | | |
| Car offer is not existing in DB | | |
| **Business Rules** | | |
|  | | |
|  | | |

### Search for a car offers

|  |  |  |
| --- | --- | --- |
| **Use Case Number**: 5 | | |
| **Name** Search for a car offers | | |
| **Brief description** Customer can filter the set of available cars. | | |
| **Actors** Customer | | |
|  | | |
| **Preconditions** | | |
| The system must have cars in the DB. | | |
| **Flows of Events:** | | |
| 1. Basic Flows | | |
| **1.1.0 Browse car offers** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The customer opens the “home” page. | The System shows the list of cars available. |
| **Postconditions** | | |
| The system will display the list of cars available. | | |
| **Business Rules** | | |
|  | | |

### Filter the car offers

|  |  |  |
| --- | --- | --- |
| **Use Case Number**: 6 | | |
| **Name** Filter the car offers | | |
| **Brief description** Customer can browse available cars. | | |
| **Actors** Customer | | |
|  | | |
| **Preconditions** | | |
| The system must have cars in the DB. | | |
| **Flows of Events:** | | |
| **Flows of Events:** | | |
| 1. Basic Flows | | |
| **1.1.0 Browse car offers** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The customer opens the “home” page. | The System shows the list of cars available. |
| 2 | Select one value for the filters (brand, model, year, …) and clicks “filter”. | The system shows a filtered set of car offers. |
|  |  |  |
| **Postconditions** | | |
| The system apply the filter and return a subset of car offers. | | |
| **Business Rules** | | |
|  | | |
|  | | |

### Select a car offer

|  |  |  |
| --- | --- | --- |
| **Use Case Number**: 7 | | |
| **Name** Select a car offer | | |
| **Brief description** Customer can select a car offer and book it. | | |
| **Actors** Customer | | |
|  | | |
| **Preconditions** | | |
| The system must have cars in the DB, the customer must to be logged in. | | |
| **Flows of Events:** | | |
| 1. Basic Flows | | |
| **1.1.0 Browse car offers** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | User click on one displayed car. | The system opens a detail page. |
| 2 | The customer input the necessary data to book the car offer and clicks “confirm” button. | The system shows a payment page. |
| 3 | The customer inputs the payment data and clicks “Pay and Book”. | The system shows the confirmation page. |
|  |  |  |
| **Postconditions** | | |
| The system process the payment, book the car offer for the customer and shows the confirmation. | | |
| **Business Rules** | | |
|  | | |
|  | | |

# DATA DICTIONARY

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | **Data Field** | **Description** | **Is**  **Required** | **Data Type** | **Format** |
| User | Username | Retrieve NT Account | Yes | Following TMS’s format |  |
| Password | Retrieve NT Account | Yes | Following TMS’s format |  |
| ARMS Role | IT Specialist, | Yes | Alphabet |  |
| Project Member | Staff Code | Retrieve from TMS | Yes | Following TMS’s format |  |
| First Name | Retrieve from TMS | Yes | Following TMS’s format |  |
| Last Name | Retrieve from TMS | Yes | Following TMS’s format |  |
| Position | Retrieve list from TMS | Yes | Following TMS’s format |  |
| Project Role | Retrieve list from TMS | Yes |  |  |
| Office Location | Retrieve list from TMS | Yes | Following TMS’s format |  |
| Join Date | Input manually | Yes | DD/MM/YYYY | Date |
| Release Date | Input manually | No | DD/MM/YYYY | Date |
| Project | Project Code | Retrieve from AGS | Yes | Following AGS’s format |  |
| ProjectName | Retrieve from AGS | Yes | Following AGS’s format |  |
| Starting Date | Retrieve from AGS | Yes | Following AGS’s format | Date |
| Closing Date | Retrieve from AGS | No | Following AGS’s format | Date |
| Tool | ToolID | Automatically added | Yes | Alphanumeric | 5 |
| ToolName | Input manually | Yes | Alphanumeric  + White Character | Min: 2  Max: 50 |
| Default Profile | Profile ID | Input manually | Yes | Alphanumeric | Min: 2  Max 25 |
| Project Role | Input manually | Yes | Alphanumeric + White Character | Min: 2  Max: 25 |

# COMMON BEHAVIOURS

|  |  |  |
| --- | --- | --- |
| **Element** | **Description** | **GUI** |
| Textbox  (for insert value) | Min: 2  Max: 25 |  |
| Grid | 25 records for 1 page  Paging: Previous … (page) … Next |  |
| Grid  (show Staff info) | Order by Staff Code, ascending |  |
| Grid  (show Project info) | Order by Project Code, ascending |  |
| Grid  (show Other Tools) | Following by Tools’ order  Display active tools only |  |
| Grid  (no search result) |  | After executing a search, if there is no result shown, a message “no result” or similar should be displayed somewhere on grid to notify. |
| Pop-up  Message Confirmation | Is warning dialog  Is confirmation dialog  Close if “ESC” pressed |  |
| Message Acknowledgement  (Existed User) |  | Message:  “This staff has been registered as an ARMS’s user. Please select different staff.” |
| Message Acknowledgement  (Existed Staff  to Project) |  | Message:  “This staff has been assigned into this project. Please choose and assign another staff to this project.” |
| Message Acknowledgement  (Existed Project  in ARMS) |  | Message:  “This project has been stored in ARMS database. Please select different project.” |
| Message  (Successfully) | Notify on top-right of the screen | Update … Successfully  Add … Successfully  Unassign … Successfully |
| Message  (Fail) |  | Update … Fail  Add … Fail  Unassign … Fail |
| Required Field | Show notification “Required Field” underneath |  |
| Validation | Validation on-field |  |
| System Error | Could not submit changes according to system’s interruption/termination | Notification message on same page, red colour:  “System error. Your changes could not be changed. Please try again.” |
| Activate/Deactivate | What is shown on GUI should be affected; otherwise, not be affected |  |
| Combo box  (Project Role) | Order by names of project roles  Ascending |  |
| Combo box  (Project Status) | Order by statuses of projects  Ascending |  |
| Combo box  (Position) | Order by names of positions  Ascending |  |
| Combo box  (Office Location, ODC Location) | Order by names of locations  Ascending |  |
|  |  |  |

# NON-FUNCTIONAL REQUIREMENTS

## Definition of non-functional requirements

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that specify specific behaviour or functions captured in use cases, sequence diagrams, state charts etc. Typical non-functional requirements are reliability, scalability, and usability. Non-functional requirements are often called the ‘qualities of a system; other terms for non-functional requirements are "constraints", "quality attributes" and "quality of service requirements". They usually cannot be implemented in a single module of a program; they are hard to model and are usually stated informally

The following stated non-functional requirements are applied for this project.

## User Interface

| Non-Functional Requirements relating to User Interface | |
| --- | --- |
| No. | Requirement |
|  | All GUI’s will have a standard look and feel of the target Operating System (i.e. they will have a consistent look & feel) |
|  | All ‘grids’ (lists) of data shown on screen will not have the functionality to be sorted and filtered by clicking on the appropriate column of that grid. |
|  | All deletions to have a confirm (Y/N) for the delete. For web application, confirmation dialog shall be the browser default one. |
|  | When navigating away from added/updated data, the system will prompt a confirm (Y/N) to save the data. For web application, confirmation dialog shall be the browser default one. |
|  | All error messages should try to give suggestions of what the error is and if possible solution(s) to rectify the problem.’ |
|  | All errors should be captured as unique and for identification a separate file should be maintained. |
|  | Vertical scrolling is allowable (but paging should be used for search results etc.)  Horizontal scrolling should be avoided. |
|  | All controls or group of controls will have a text label |
|  | Tab order on the GUI should flow from left to right, top to bottom by default but can be changed by localization if required (Please note that localization is not a standard non-functional requirement). |
|  | Use confirmations only when:   * There is a clear reason not to proceed and a reasonable chance that sometimes users won't. * The action has significant consequences or cannot be easily undone. * The action has consequences that users might not be aware of. * Proceeding with the action requires users to make a choice that doesn't have a suitable default. * Given the current context, users are likely to have performed an action in error. |
|  | For desktop applications and websites: default support will be to a resolution of 1024x768 pixels. |
|  | Where it is possible to calculate then Progress Bars / waiting indicators (depending on the system) should be used for lengthy processes to give the user an idea of what has completed – if not then the standard waiting indicator in the shape of a circling ring will be shown. |
|  | Any drop down lists will be in alphabetical or numerical order (if not stated otherwise within the functional requirements). |
|  | Where a style guide is provided for the system then the UI will follow those guidelines.  Where screen mock-ups, wireframes are provided for individual screens, these will be used in conjunction with the style guide.  Where nothing is available then this document will be used. |

## External Interfaces

| Non-Functional Requirements relating to Programming External Interfaces | |
| --- | --- |
| No. | Requirement |
|  | The standard for all interfaces going out (from) the system will be via XML or JSON. The programming interface and all data items will be documented. |
|  | For programming interfaces coming in to the system, the standard process will be to parse the data, report any errors at this point, then process the data (and report on any errors at this point, unless there’s explicit requirement that the system does not need to check because of performance reasons.  Each programming interface (including data items) is expected to be documented to a level that design can commence. |

## 

## Usability

| Non-Functional Requirements relating to Usability | |
| --- | --- |
| No. | Requirement |
|  | The GUI will contain ‘plain English’ (therefore spell checked) and not ‘jargon’ that only certain staff can understand |
|  | No auto-completion of fields will be developed. |
|  | The system will try to minimize the number of clicks per GUI/transaction and increase the input capabilities per window. |
|  | When errors occur at any part of the system then a specific diagnostic capture of current status to aid investigation shall be created. |
|  | Where 2 users are updating the same record at the same time, then the user who clicks save last will have their data saved to the database.  Where the client requests to lock the last save, then additional effort will need to be added. |

### Data Protection Act

| Non-Functional Requirements relating to Data Protection Act | |
| --- | --- |
| No. | Requirement |
|  | The system will need to comply with the Data Protection Act.  This act gives individuals the right to know what information is held about them. It provides a framework to ensure that personal information is handled properly; the Act has 2 main areas:   1. Anyone who processes personal information must comply with eight principles, which make sure that personal information is:    * Fairly and lawfully processed    * Processed for limited purposes    * Adequate, relevant and not excessive    * Accurate and up to date    * Not kept for longer than is necessary    * Processed in line with your rights    * Secure    * Not transferred to other countries without adequate protection 2. Individuals have the rights to find out what personal information is held on computer and most paper records.   This section is added as a reminder to the client about their responsibilities for the Data Protection Act; There will not specifically write any code for the Data Protection Act requirements |

## Performance

| Non-Functional Requirements relating to Performance | |
| --- | --- |
| No. | Requirement |
|  | All normal data input, data output and content presenting application screens need to be ready to be interacted with within 3 seconds during normal load and normal client/server connection with following definition   * Normal data input screen: screens with under 30 fields, interact directly with back end without complex data calculation, not interact with external systems and optionally save data directly into database, normal data input screen will not have big contents such as big image, video, audio files (less than 200KB in total size) * Normal data output screen: query directly from database without complex lookup, not go thru complex data processing or external server interaction, to display less than 50 rows and 10 columns and each row less than 100 character, without big contents (maximum content related size less than 200KB in total) * Normal load: 30 concurrent user for non-load balanced normal server * Normal server: Intel Core i5, 4GB RAM, 500GB hard disk. * Normal client/server connection: 500KB/s   By default we will not do performance testing.  For other cases (complex processing, seeking, calculation, and big report) there will be specific agreement after doing real benchmarking, although we will seek to optimize the code. |
|  | No single process shall consume inappropriate CPU usage. (E.g.: a common request leads 100% CPU workload) |

## 

## Security

| Non-Functional Requirements relating to Security | |
| --- | --- |
| No. | Requirement |
|  | Before accessing the system users will authenticate themselves with a username and password. |
|  | If either the username or password is incorrect, then the system will respond with ’The username and/or the password was incorrect, please re-try’. (i.e. the response will not tell specifically whether the username was wrong or the password was wrong). |
|  | Access to screens/data shall be controlled by role based security |
|  | At a minimum all sensitive data (e.g. passwords, all personal data, ID’s, any financial/accounting data, credit/debit card information, banking information etc.) communicated shall be encrypted via 1024bit SSL. |
|  | All sensitive data stored will be encrypted. (Sensitive data will be detailed in the functional requirements). |
|  | All interfaces containing sensitive data will be encrypted to 1024bit and utilize certificate based authentication. |
|  | Digital Certificates should be used in transactions for authentication purposes |
|  | Passwords shall never be displayed |
|  | As a principal the system shall be designed with the OWASP top 10 Web Application Security Risks in mind; but if full compliance to these is required then this must be specifically requested by the client (which may increase the effort and estimate). |
| 1. . | Error messages shall not disclose the internal working of the system |

## Supportability

| Non-Functional Requirements relating to Supportability | |
| --- | --- |
| No. | Requirement |
|  | For a Web interface (if applicable), it will be able to be accessed via standard PC/Laptops running the latest MS Windows operating system (and the previous version) and will be supported to run on the following browsers:   * Internet Explorer (latest official version as the project start date) * Google Chrome (latest official version as the project start date) * Firefox (latest official version as the project start date)   Note that supporting more than the 3 browsers stated will require additional effort and cost. (which would be added to the estimate) |
|  | For a Web interface (if applicable) and if we are required to support mobile browsers, by default we will support the following mobile browsers:   * Safari Mobile for iPad (latest official iOS version at the project start date) * Android stock browser for Tablet 10 inches with Android OS (latest official version at the project start date) * Internet Explorer for Windows RT (latest official version at project start date)   Note that supporting mobile browsers is to provide a touch friendly UI only and a consistent view across desktop and mobile browsers, not to provide different screen layout or behavior except differences that are inherited from the OS(s) (such as behavior for form elements, date time pickers, dropdowns etc. |
|  | There is no requirement to be able port the software onto additional Operating Systems or hardware platforms beyond an agreed single platform. |
|  | All application code shall be documented so that it can easily be maintained and enhanced. (This may be by in-line code comments and/or separate documentation to the code). |
|  | All code (Java, .NET & PHP) will be written to the Nash Tech coding convention and standards (for which copies are available). |
|  | The system will try to ‘re-use’ existing services if possible. So as new Services are commissioned and introduced they will benefit from re-use of standard operations and architectural patterns utilized by the Solution. This reduces development and deployment time as well as significantly reducing risk. |
|  | All interfaces shall have a supplied test harness to load and simulate all successful (just within boundary values) as well as failure events.  Interfaces supplied by the client they are required to be provided with a test harness. |
|  | If there are client dependencies then the application shall test / diagnose whether these are in place when it first executes and provide appropriate identification to the end users as well as creating an event in the server side logs for IT investigation. E.g. a smart client app within the system detects the .net framework is the wrong version, or browser detects an incompatible browser version of browser setting. |
|  | The system shall be installed or upgraded via automated installation scripts or programs to avoid potential for human error for local only. These scripts shall contain error trapping to aid installation diagnostics.  There will be no automated update / upgrade for applications thru network.  There will be no automated scripts for production deployment completely. |
|  | There shall be no version correction checking across servers |
|  | Client installations shall be centrally manageable as far as possible. |
|  | The system shall be able to be uninstalled completely. |
|  | Where ever possible the systems shall be centrally configurable (e.g. to avoid editing in files on multiple application servers) |
|  | The system will log every key function call.(NB: There are free open source pieces of software available for this e.g. Log4J) |

## Reliability

| Non-Functional Requirements relating to Reliability | |
| --- | --- |
| No. | Requirement |
|  | Client internal systems software should be designed to be available 12 hours x 7 days per week \* |
|  | Web systems software shall be designed to have an annual availability from unplanned outages of 99.9% \* |
|  | Pure batch systems software should be designed to have a successful run rate (i.e. full completion to end) of 99.9% \* |