**Software Engineering**

**Software Requirements Specification**

**(SRS) Document**

**LINQ**

**20th April 2020**

**Ver. 1.01**

|  |
| --- |
| **Revisions** |

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.00  1.01 | James Apps  James Apps | First overall draft  Requested web interface (NodeJS), instead of desktop client (Java); minor changes and emphasis to the suite | 29th March 2020  20th April 2020 |

|  |
| --- |
| **Table of Contents** |

[1. Introduction 3](#_Toc36067728)

[2. General Description 4](#_Toc36067729)

[3. System Requirements 5](#_Toc36067730)

[4.External Interface Requirements 6](#_Toc36067731)

[4.1 User Interfaces 6](#_Toc36067732)

[4.2 Hardware Interfaces 6](#_Toc36067733)

[4.3 Communications Interfaces 6](#_Toc36067734)

[4.4 Software Interfaces 6](#_Toc36067735)

[5. Non-Functional Requirements 6](#_Toc36067736)

1. Introduction

* 1. **Purpose**

The goal of this project is to provide an efficient and responsive student record management system for schools and colleges. The main objectives are:

1. Reduce the time required to process students’ raw results to yield statistical insights and trends across a given time period.
2. Provide a centralised database of student data which can be independently accessed qualified members of staff, as well as parents and students.
   1. **Intended audience**

This SRS is intended to outline and describe the uses of the student record management suite, known as LINQ, to teachers, school and IT administrators, and software developers.

* 1. **Scope**

Schools assess pupils and then gather results to homework, coursework and tests to analyse and improve student performance. Data needs to be recorded with regularity, conformity and accuracy. Much of the data is *formative*, which aims at quickly identifying areas of improvement for teachers and students. Other data is *summative*, which is a measure and summary of student performance and attainment at a given stage of a course (usually at the end of a term, semester or academic year).

The *raw data* is almost always converted into *processed data* so that comparisons and trends can be drawn. At various stages of the academic year letter grades and/or overall scores, which are used for internal assessment and external examination predictions, are calculated from *cumulated* processed data. The conversion often presents an added burden to teachers, who may be trying to follow school assessment policies consistently across large datasets and to tight deadlines. The data is in part made available to not only fellow teachers but also parents, board of governors, inspection bodies and other colleges/universities. The data also proves helpful during end-of-year school reviews.

Formative results pertain to assignment types, or *components*. A component is a student assessment or task which contributes to the overall analysis of a student’s academic performance for the year. There are different types of components: homework, tests (short, focused exams), coursework, student effort and mock exams (long, comprehensive exams). Some schools calculate an overall score and letter grade which is derived from weighted contributions of each component. Scores can be expressed as a percent, a uniform mark scale (UMS) or a percentage uniform mark (PUM).

LINQ focuses on the recording, processing and publication of students’ in-school formative results. It unifies the policies related to the data required of all teachers. LINQ grants school leaders control of processing policies, processing the data automatically, freeing up subject teacher workloads. LINQ grants relevant cleared staff access to the data at any time without requiring intervention from teachers who recorded the data. The output can be viewed directly from LINQ, printed or take the form of other file formats, and be displayed in web browsers on Windows, Linux and Mac machines as well as Android and IOS based tablets and mobile phones. The data stored on LINQ is confidential and so encryption, authentication and authorisation procedures will be implemented.

## 2. General Description

**2.2 Product features**

LINQ is suite of applications.

Subject teachers can have their own local copy of LINQ, herein referred to as **LINQ-Android**. Teachers can connect to, view and update records on a central copy of LINQ, known as **LINQ-server**. Authenticated users (parents) can be granted read-only access to student’s data using **LINQ-web**. More specific and up-to-date details of these applications can be found in their respective SRS documents. Any general references to “LINQ” should be taken as a reference to all components of the LINQ-suite, each of which contribute to the given functionality in some way.

Student data is recorded via LINQ-client only. User credentials, authorisation, general troubleshooting and result processing and publication policies are handled by LINQ-server.

Below are the major system features regarding data uploading and processing to LINQ-server database. Note, points (c), (g)-(j) are set by school leaders only.

1. Alphanumerical raw data input via camera/scanner (optically recognisable template for student assignments provided) with LINQ-Android or manually, via the LINQ-server user interface
2. Processing of numerical data (statistical mean, median and standard deviation)
3. Customisable statistical formula editor to enable school staff to process data in forms not provided by LINQ
4. Trend plotting of (b) and (c) presenting, for example, student mean compared to the class mean and student mean compared to accumulated mean.
5. Graphical plotting of trends of (c)
6. Letter grade allocation (based on supplied thresholds) for a given assignment (a)
7. Individual students’ accumulative average of a given type of assessment (a) and/or (b), *i.e.* average of all homework, coursework, test or mock exam scores over a given time period
8. Letter grade allocation of the accumulative average(s) from (f)
9. Overall end-of-term or end-of-semester percentage based on weighted contributions of each type of assessment from (f)
10. Letter grade allocation of overall end-of-term or end-of-semester percentage of (h), based on agreed thresholds

Below are the major system capabilities of the web-based portal, LINQ-web, which is handled by LINQ-server.

1. Web-based chat client which relates school-parent messages with relevant assignment data
2. Web-based portal UI which closely matches LINQ’s UI (including tabulation and trend plotting)
3. Authentication protocols to safeguard sensitive information

LINQ can also output selected records into the PDF file format. A print facility is also available. LINQ can import/export student data from properly formatted XLSX files.

The LINQ server database integrates all records on a centralised MySQL database. Subject teachers who teach the same student are granted access to other subject teachers’ data stored on the server database. Subject teachers can download an Excel XLSX file with information on their own students (names, contact details, raw scores, assignment details and thresholds, and cumulative averages).

* 1. **User class and characteristics**

The largest group of users will be subject teachers, each of whom log on LINQ-server to handle the data entry, plot trends and communicate with parents. Access to LINQ requires authentication. Teachers can print results and generate PDF readouts. Teachers can also export XLSX files. Finally, teachers can access the data recorded by other teachers, if they all teach the same student and access any processed data, set by the school leaders.

Students and parents will not be granted access to LINQ and instead access finalised results (raw only) after authentication, either from a printed report, an email message or via LINQ-web. They will also be able to communicate with the subject teacher regarding specific assignments, via a chat-based system as part of LINQ.

School leaders (Principals and their deputies, and Heads of Department) will decide how data is processed and set policies on LINQ-server. School leaders will have (if they do not already) their own LINQ-server login details with which to set result processing policies on LINQ-server. School leaders can build their own plots on their own LINQ-client database and store plotted data and trends on LINQ-server. Different plots will be labelled accordingly to distinguish between school leader plots and subject teacher plots. Furthermore, school leaders can decide how to process summative results, if required, to compute overall averages for a term, semester or academic year, as well as decide what other information (in addition to raw results) to prepare for parents, colleges/universities and government inspectors.

* 1. **Operating environment**

LINQ-server is designed to run on Windows, Linux and Macintosh machines. The computer specification will generally depend on the size of the school.

* 1. **Constraints**

None

* 1. **Assumptions and dependencies**

None.

## 3. System Requirements

**3.1 Functional requirements**

All the requirements within the system or sub-system in order to determine the output that the software is expected to give in relation to the given input. These consist of the design requirements, graphics requirements, operating system requirements and constraints if any.

**LINQ-server**

* Any individual should be required to login with a username and password. The option of a single, faster PIN system (after the login and password details have been confirmed) to log in with is available.
* Repeated login attempts should signal a temporary lockout. This should be resolvable locally (not remotely) by one appointed school administrator. LINQ is dependent on MySQL running in the background. Login credentials to MySQL are handled independently to LINQ and the responsibility of the school administrators.
* Ensure that only LINQ administrators are permitted to login and work locally from LINQ-server.
* LINQ administrators are granted read-write access to all data in LINQ whereas teachers are granted access to academic data which they are directly responsible for.
* Allow for the initialisation of new LINQ user accounts. This should include a minimum of name, department, email address, username and password.
* Allow for the initialisation of new LINQ-server administration accounts, requiring authorisation from all current LINQ-server administrators. This should include a minimum of name, department, email address, username and password.
* Allow for the initialisation of new user accounts (username and password) of the web-portal.
* Allow for the initialisation and configuration of all student details. The table should include a minimum of name, form, form tutor’s name, form tutor’s email address, form tutor’s LINQ\_id, Year group, Head of Year’s email address, email addresses of parents and/or guardians and subjects studied. More descriptive fields (columns) can be added in the future if required.
* Allow for the generation of SQL tables by student and subject teacher. The former allows form tutors or (other) subject teachers to gather data for an individual student and the latter allows subject teachers to record their own students’ data.
* Allow for a school leader to initialise and configure the fields of SQL tables for the purposes of:
  + processing raw data, calculating scores as a percentage (scores marked NULL are ignored in all subsequent calculations to prevent penalising students)
  + generating cumulative results and calculating the latest averages
  + mapping of cumulative averages into letter grades
  + processing of finalised cumulative results into end-of-term/semester/year numerical scores and associated letter grades
  + Collate and summarise year-wide data by subject
* Allow for the backup of SQL tables when required
* Allow for the configuration of automated emails to relevant users (parents, teachers *etc*.)
* Allow for the notification of error- and periodic maintenance-related notifications for LINQ users
* Allow LINQ administrators to send messages to LINQ-client users as required
* Allow LINQ administrators to add/remove students/teachers, while retaining anonymised academic data for future review purposes.

**LINQ-web**

* Present all functions on a responsive user interface, available to users with 1080p monitors down to mobile phone screens
* Secure (https) login setup and management screen
* Require user interface which can refresh/reload the latest student data while logged in
* Allow for printing and PDF file download of available student data (as determined by LINQ-server)
* Automatically logout the user after a given time limit, period of inactivity or shutdown of a web browser tab or window

**Non-administrator users of LINQ**

* Ensure that only one LINQ login connects to LINQ-server at any given time
* Allow for the creation, deletion, and update of their designated subject on LINQ-server tables
* Allow for the retrieval of student data from SQL tables
* Allow for the configuration of new homework assignment, coursework, test, and mock exam fields when required
* Allow for the input of raw student results (with possible values > 0, 0 or NULL)
* Allow for instant conversion of raw to processed data
* Allow for the configuration and conversion of processed results into letter grades (these would be determined by class or set)
* Allow for comparisons to the class average to be deduced, monitoring the numerical raw or processed data since the beginning of a semester or term. This will take the form of
  + colour coded spreadsheet format and/or
  + graphical plot
* Allow for letter grade and on-going comparisons to be uploaded to LINQ-server

**LINQ-Android**

* LINQ-Android serves as an Android version of LINQ but with specific focus to:
  + Scanning of scores by camera (require a template which LINQ-Android recognises)
  + Synchronisation with LINQ-server
* A near-identical UI to LINQ-client is expected

## 4.External Interface Requirements

4.1 User Interfaces

*Please see application specific SRS documents*

4.2 Hardware Interfaces

LINQ is designed to run on Windows, Linux and Macintosh machines.

Details of network capabilities TBD.

4.3 Communications Interfaces

Details of network protocols and backend web interfaces TBD.

4.4 Software Interfaces

LINQ-server is designed to run alongside MySQL server version 8 and NodeJS version 13. LINQ-Android is expected to run on top of the Java 11 VM.

## 5. Non-Functional Requirements

**5.1 Performance requirements**

Except for the handling and processing of large datasets (anything higher than school class level size) all responses to data entry and data changes should be instant (< 1 s).

Staff should be able to download, and process class-sized datasets in < 2 s. School wide processing of data should be completed within 600 s.

**5.2 Safety requirements**

List out any safeguards that need to be incorporated as a measure against any possible harm the use of the software application may cause.

*None*

**5.3 Security requirements**

Privacy and data protection regulations that need to be adhered to while designing of the product

*All schools and colleges are expected to prepare SSL based connections to the server and configure MySQL securely (we recommend addressing many of the default settings, minimising the number of administrative users, and blocking all unnecessary ports with a firewall. LINQ-server can also be installed on cloud-based systems. The cloud-based system adopted is the responsibility of the customer.*

**5.4 Software quality attributes**

Detailing on the additional qualities that need to be incorporated within the software like maintainability, adaptability, flexibility, usability, reliability, portability etc.

*TBD*

**5.5 Other requirements**

These may include the legal requirements, resource utilizations, future updates etc.