Requirement Specification

Group 2

Group Members:

Nico Taljaard (10153285)

Abraham Daniel Pretorius (12022404)

Mathys Ellis (12019837)

Mbulungo Musetsho (10176382)

Verushka Moodley (29117454)

Eduan Bekker (12214834)

Johan Esterhuyse (10043285)

Version 3.0

Change Log

10/02/2014	Version 1.0	Document Created	Nico Taljaard
15/02/2014	Version 1.0	System Description	Mathys Ellis
17/02/2014	Version 1.1	Edited Formatting	Nico Taljaard
17/02/2014	Version 1.1	Technical Specification	Mbulungo Musetsho
17/02/2014	Version 1.1	Created Non-functional Requirements	Eduan Bekker
18/02/2014	Version 1.2	External Interface Requirements	Abraham Daniel Pretorius
18/02/2014	Version 1.2	Functional Requirement	Nico Taljaard
18/02/2014	Version 1.2	Updated Non-functional Requirements	Eduan Bekker
18/02/2014	Version 1.2	Updated External Interface Requirements	Abraham Daniel Pretorius
18/02/2014	Version 1.2	Updated System Description	Mathys Ellis
20/02/2014	Version 1.3	Introduction	Verushka Moodley
20/02/2014	Version 1.3	Fixed Compile Errors	Nico Taljaard
20/02/2014	Version 1.2	Updated External Interface Requirements	Abraham Daniel Pretorius
21/02/2014	Version 1.2	Fixed Spelling	Nico Taljaard
21/02/2014	Version 2.0	Change formatting	Nico Taljaard
23/02/2014	Version 2.0	Checked spelling and grammar	Mathys Ellis
24/02/2014	Version 3.0	Changed to new layout specification	Nico Taljaard
24/02/2014	Version 3.0	Required Functionality	Nico Taljaard
24/02/2014	Version 3.0	Quality requirements	Eduan Bekker
24/02/2014	Version 3.0	Integration Requirements: Protocols, API specifications,	Abraham Daniel Pretorius
26/02/2014	Version 3.0	Integration Requirements: Integration channel to be used:, Protocols, API specifications, Quality requirements for Integration,	Abraham Daniel Pretorius
27/02/2014	Version 3.0	Added use case prioritization and added possible glossary terms with out defi	Mathys Ellis

Contents

1 Introduction

This document is the software requirements specification for a computerised marking system to be used by the University of Pretoria. It begins by addressing the purpose and vision of the project then continues with a general discussion on the background of the project. Thereafter, the architectural requirements are identified with a main focus on the access channel requirements, quality requirements, integration requirements, and architectural constraints. The function requirements follow by discussing the application functionality required by the users of the system. In this section of the document, many diagrams are used to depict the flow of communication and interaction between the users and the system. It will also illustrate the processes that the system will need to complete and the states that the system will undergo. The next topic will communicate the open issues on some of the requirements, and the document concludes with a glossary of possibly unfamiliar terms used in this document.

2 Vision

The purpose of this document is to communicate the requirements and proposed solution to the client, Mr Jan Kroeze, who requires a computer system for marking purposes. This document will outline the scope of the project and thus serve as a formal agreement and contract between the developers and the client. It will also serve as a reference and eliminate any confusion that may occur in the later stages of development.

3 Background

The University of Pretoria currently has a manual paper based marking system which has many flaws (losing marks is one of the major issues). Our solution aims to assist them by providing a computerised system. The required computer system should aid the teaching assistants and lecturers in the university in recording the marks of assessments for students, and the maintenance of marks thereafter. It should also allow students to keep up to date with their marks. The solution will consist of an application that students, teaching assistants and lecturers will use in the following way:

- Teaching assistants will receive mark sheets of registered students and will be allowed to assign practical marks to the students via the app. Lecturers will also be granted these permissions.
- Students will be able to view a list of their marks via the application

The solution will also consist of a web interface that will allow the following:

- Lecturers can maintain and integrate marks
- Lecturers can request reports with a certain criteria. A graph of some nature or statistics will then be presented
- An audit log will be updated automatically

The solution has the following restrictions:

- Users must be registered- a username and password is required for access to the system
- Marking sheets may only be locked/unlocked by lecturers that are assigned to the module. Teaching assistants will be allowed to update marks according to the status of the mark sheet. Students may only view their mark when the mark sheet is locked.
- No one may be allowed to edit the audit log. Only lecturers have permission to view the audit log.

4 Architecture requirements

4.1 Access channel requirements

- The system must support the following platforms:
  - Android (API 15)
  - Web browser (with HTML 5 compatibility)
- The system must strictly operate over HTTPS
- The following technologies and languages must be used for implementation purposes:
  - Python with Django (server-side programming)
  - Java (Android Module Development)
  - MySQL (Database)
- SOAP interface must be utilized for this system
- LDAP (account management)

4.2 Quality requirements

- Authentication
  - All users have to login before they are able to access the system
- Audit-ability
  - All the actions of all the users will be added to the audit trail log
  - The events consider as audit trails will include:
    * The login and logout of users
    * The modifications of any data
- Scalability
  - The system should be able to handle multiple running practicals simultaneously
  - The performance of the system should not be dependent on the number of active users
- Availability **There will be 3 levels of privileges: students, markers and lecturers**
  - Students will only be able to view marks
  - Uploading of practical marks by markers should only be available when the practical starts until a time specified by the lecturers
  - Editing marks will only be available for lecturers
  - Audit trails can only be viewed by lecturers
  - Audit trails cannot be modified by anyone

4.3 Integration requirements

Integration channel to be used:
(Priority: High, Requirement: ARQIRQ1)

- SOAP (Simple Object Access Protocol) will be used as an interface between different mediums and platforms to interact with the system.

Protocols
(Priority: High, Requirement: ARQIRQ2)

1. SOAP (Simple Object Access Protocol) will be used to transfer structured information over the network (namely the Internet).

API specifications
(Priority: High, Requirement: ARQIRQ3)

The system has to interface with the following external mediums:

1. Internal API for Android systems. Android API version 15 (Ice Cream Sandwich).
2. External API for data transfer between interconnecting systems. This API uses SOAP transfer the data.

All API's should represent their data from SOAP to:

1. XML for interfacing with web servers using WSDL.
2. Output marks to .CSV format.

Quality requirements for integration
(Priority: High, Requirement: ARQIRQ4)

1. Performance:
    - People viewing the marks should not need to wait extensively, // however the system can only be as fast as the connection to it.// Therefor the system cannot be expected to be faster than the connection
      that is being used.
    - Documentation drawn up from the marks may take between
      10 - 30 seconds to be compiled.
    - Markers should be able to update marks within 10 seconds of each other.
2. Scalability:
    - The system will be designed with design patterns. These are used so that the it is easy to upgrade the different aspects independently.
3. Reliability:

- A software issue should not arise, but if it does the application must handle the problem and send an appropriate error message to the user. If data has failed to send the application must reattempt to upload the data for a specified amount of time (1 minute) or until such a time that the user aborts data transfer.

4. Security:

   - No user, who is unauthorized, may gain access to the system.
   - Authorized users can only access the aspects of the system as allocated to them by the administrator.
   - Security questions, as well as email addresses will be linked to each of the users. This will be used for password recovery.

5. Auditability:

   - Each time marks are changed it will be recorded in an audit trail which not even the system administrator would be able to edit.

4.4 Architecture constraints

Technologies which MUST be used
(Priority: High, Requirement: ARQ1)

- The system must support the following platforms:
  - Android (API 15)
  - Web browser (with HTML 5 compatibility)
- The system must strictly operate over HTTPS
- The following technologies and languages must be used for implementation purposes:
  - Python with Django (server-side programming)
  - Java (Android Module Development)
  - MySQL (Database)
- SOAP interface must be utilized for this system
- LDAP (account management)

5 Functional requirements

5.1 Introduction

The goal of the system is to provide the client with a secure, scalable and remotely accessible marking and mark management system. The system is comprised of four different facets which are listed below. It is intended to replace the current marking and mark management system employed by the client, which is currently paper and spreadsheet based.

Student mark retrieval facet:

The goal of this facet is to provide the "Students" with a secure and private means to a read-only view of their marks, for the markable items of a specified course on the system. The means by which they will view the markable items will be in the form of a website interface and android application.

Marker marking tool facet:

The goal of this facet is to provide "Markers", assigned to a particular course, with a mobile tool to allow then to be able to add and update the mark of a specified student on the mark list for a particular markable item of the particular course. The tool will allow "Markers" access to the mark list from any location where an internet connection is present. The tool will be in the forms of a website interface and android application.

Lecturer mark management facet:

The goal of this facet is to provide "Lectures", assigned to a particular course, with a means to manage the marks of each student assigned to the course and mark sturcture of the course. Where the term manage comprises of adding, modifying and removing markable items, mark lists and the individual marks of students. Further the facet also has the goal of providing a means to report on mark related data on different levels of granularity of a particular course.

Audit trail facet:

The goal of this facet is to give the system the ability to track all critical actions that occur on the system independent of any user interference and also provide a read-only view of such trails for authorised users of the system. Where critical actions comprise of adding, updating and removing any data on the database as well as login and logout actions.

5.2 Scope and Limitations/Exclusions

5.3 Required functionality

Course API
(Priority: High, Requirement: FRQ1)

- A course must be creatable.
- Lectures must be added to course by users with correct authorization.
- Lectures must be able to add Teaching assistants and Tutors to the course, aswell as be able to assign them to a practical time slot.
- Student information for each student that is assigned to a practical time slot of a course should be pulled from an existing database so that teaching assistants and tutors of the course can find the student to be marked.

Lecturer API (Priority: High, Requirement: FRQ2)

- A tasks must be creatable for the following:
  - Practicals
  - Assignments
  - Class tests
  - Tests
- Tasks should also contain the following information:
  - Starting time to open access.
  - End time if a specified time is required, else manual locking is required by lecture.
  - A rubric must be added to show the mark allocation.
- Adjust mark weight allocations.
- Lectures alone should be able to change marks of his own subjects.
- Assign security roles to teaching assistants and tutors.
- Move students to a different teaching assistant or tutor.

- Use the reporting API.

Auditing API (Priority: High, Requirement: FRQ3)

- Log file for following activities:
  - Marks added by whom and when.
  - Marks changed or removed by who, when and what is the reason.
  - Login and logout activites
- Lectures must be able view only the audit logs of the course(s) they are assigned to.
- Head of Department alone should assign a user that can view the entire change log.
- No edits to the audit log allowed.

Marking API (Priority: High, Requirement: FRQ4)

- Accessible through mobile application for teaching assistants or tutors.
- Display all available mark lists for teaching assistants or tutors to mark.
- Within mark list display all students registered for current session with a search option.
- Search filters should be available for student number, surname or name. The displayed results should contain all the results that match the filters.
- For the selected student the marking rubric should be displayed with type able fields.
- Marks should be submitted to the database directly after mark has been finalized for a student.

Reports API (Priority: Medium-High, Requirement: FRQ5)

- Marks can be exported in a .csv file containing the selected marks for each student.
- All reports should be based on a select set of marks.
- Numeric statistics can be exported about marks.
- Graphical reports should be exported to .pdf file.

Student API (Priority: Medium, Requirement: FRQ6)

- Accessible through mobile application or web interface.
- Landing page displays all subjects of current student.
- Subject marks should be viewable for separate assessments as well as a progress mark of a particular course.

5.4 Use case prioritization

- **Critical:**
  - Log in [1.1; 2.1; 3.1; 4.1]
  - Create assessment [1.10]
  - Set rubric [1.12]
  - Set constraints [1.13]
  - Set markers [1.14]
  - Select assessment [1.11; 2.3; 3.5]
  - Mark assessment [2.5]
  - Submit marks [2.6]
  - Select course [1.2; 2.2; 3.2; 4.2]
  - Select student [1.6; 2.4]
  - View marks [3.3]
  - View overall marks [1.3]
  - Export marks [1.4]
  - View entire audit trail [4.4]
- **Important:**
  - View student marks [1.5]
  - View students overall marks [1.8]
  - Update assessment [1.9]
  - Update rubric
  - Update markers

  - Update constraints
  - View assessment mark [3.4]

- **Nice-to-Have:**
  - Export entire audit trail [4.4]
  - Select user [4.5]
  - View individual audit trail [4.3]
  - Export user audit trail

5.5 Use case/Services contracts

- **Pre-Conditions:**
- **Post-Conditions:**
- **Request and Results Data Structures:**

5.6 Process specifications

5.7 Domain Objects

6 Open Issues

7 Glossary

- Student -
- Marker -
- Lecturer -
- Markable item -
- Mark list -
- Course -
- LDAP -
- API -
- HTTPS -
- HTML 5 -
- PDF -
- XML -
- CSV -
- SOAP - Simple Object Access Protocol
- WSDL - Web Service Definition Language
- Android -
- Django -
- Python -
- Java -
- MySQL -