Clinical Competency Calculator

Software Requirements Specification
Draft Document

Dylan Colburn
Jacq Lee
Tyler Muessig
Tyler Price

v0.0.2
Nov. 29, 2024

Contents

1 Introduction

1.1 Purpose

The purpose of the software requirements document is to create a comprehensive outline of the Clinical Competency Calculator (CCC) for the development team, project advisors, faculty advisors, stakeholders, and members of the Hershey Medical team that will be maintaining this software. This outline will describe the functions and high-level requirements of the CCC. The target audience is expected to know technical terminology related to software development.

1.2 Scope

※ ⟹ This document specifies the CCC as software designed to be used for the appraisal of **students** undergoing clerkship at Hershey Medical, as is to be utilized by specific department or clerkship. The CCC will provide a means of collecting feedback from **raters** on the **competencies** of students and performing useful, actionable analyses thereon. The CCC is not designed to interface with existing Hershey Medical systems or engage in information exchange therewith. The CCC is also not designed for utilization by other departments within Hershey Medical; should this software be used outside of the stated scope, further requirement elicitation should be carried out and relevant issues addressed.

1.3 Client information

※ ⟹ The development of the CCC is being sponsored by Anthony Dambro (adambro@pennstatehealth.psu.edu) from Hershey Medical. (Expand)

1.4 Users

Our intended audience for the CCC would be the students undergoing clerkship, raters providing feedback on students' competencies, and administrators of the entire system and its data.

1.5 Definitions

The definitions for terms required to properly interpret this document are listed below. The first mention of each term in this document (excluding this subsection) is highlighted in teal and boldface, and is a hyperlink that will redirect to this subsection when clicked.

Words listed in uppercase indicating requirement levels (MUST, MUST NOT, SHOULD, SHOULD NOT, and MAY) are adapted from RFC 2119 and follows the specification by RFC 8174 that these defined special meanings only apply to uppercase usage of these terms.

In general, the definitions of terms used in this document not listed below conform to the definitions provided in IEEE Std 610.12-1990.

This document is incomplete. The external file associated with the glossary 'main' (which should be called `srs-v0.0.2.gls`) hasn't been created.

Check the contents of the file srs-v0.0.2.glo. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like \gls or \glsadd) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

If you don't want this glossary, add nomain to your package option list when you load glossaries-extra.sty. For example:

```
\usepackage[nomain]{glossaries-extra}
```

Try one of the following:

- Add automake to your package option list when you load glossaries-extra.sty. For example:
  `\usepackage[automake]{glossaries-extra}`
- Run the external (Lua) application:
  `makeglossaries-lite.lua "srs-v0.0.2"`
- Run the external (Perl) application:
  `makeglossaries "srs-v0.0.2"`

Then rerun LaTeX on this document.

This message will be removed once the problem has been fixed.

1.6 Acronyms and abbreviations

The definitions of acronyms and abbreviations required to properly interpret this document are listed below. The first mention of each abbreviation in this document (excluding this subsection) will be accompanied by its full long form.

This document is incomplete. The external file associated with the glossary 'abbreviations' (which should be called srs-v0.0.2.gls-abr) hasn't been created.

Check the contents of the file srs-v0.0.2.glo-abr. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like \gls or \glsadd) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

Try one of the following:

- Add automake to your package option list when you load glossaries-extra.sty. For example:
  `\usepackage[automake]{glossaries-extra}`
- Run the external (Lua) application:
  `makeglossaries-lite.lua "srs-v0.0.2"`
- Run the external (Perl) application:
  `makeglossaries "srs-v0.0.2"`

Then rerun LaTeX on this document.

This message will be removed once the problem has been fixed.

1.7 References

Core Entrustable Professional Activities For Entering Residency: Faculty And Learners' Guide. Association of American Medical Colleges, Washington, D.C. Accessed: Nov. 27, 2024. [Online]. Available: https://store.aamc.org/downloadable/download/sample/sample_id/66/%20

IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, IEEE Standards Board, Sep. 1990. Accessed: Nov. 27, 2024. [Online]. Available: https://www.informatik.htw-dresden.de/~hauptman/SEI/IEEE_Standard_Glossary_of_Software_Engineering_Terminology%20.pdf

Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, RFC Editor, Mar. 1997. Accessed: Nov. 27, 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc2119

Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words, RFC 8174, RFC Editor, May. 2017. Accessed: Nov. 27, 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc8174

2 Overall description

2.1 Product perspective

The CCC is a grading tool to be used to assess the competencies of students undergoing clerkship. The CCC aims to replace and improve upon the preexisting method of collecting feedback from raters by aggregating more useful feedback and providing more insightful analyses. Via this system, ※ ⟹ students and administrators will have the ability to review student performance as reported by raters. Administrators will also be able to filter low-quality feedback. All of these options will be implemented to allow students, raters, and administrators to access them via any device that has a modern web browser, such as smartphones, laptops, and desktop computers.

2.2 Product functions

2.2.1 Student functions

※ ⟹ Students must be able to request a specific rater to fill out a **feedback form** for themselves. Confirm actual mechanism of requesting, re: anonymity, scope of request, etc. Students must also be able to view a full report of their performance, which displays where they land along the **Entrustable Professional Activities** (**EPA**s) and allows the student to view specific comments or other relevant feedback that supports the specific assessment.

2.2.2 Rater functions

The rater must be able to fill out new feedback forms by accepting requests from students. Additionally, they must be able to view past feedback forms that they have submitted.

2.2.3 Administrator functions

Administrators must be able to view all feedback forms submitted, along with the rater who submitted the forms. Administrators must be able to view the full report of any student. Administrators must also be able to audit and reject **form responses** that are of poor quality or otherwise not useful.

2.3 User characteristics

The **users** of the CCC will have a basic sense of computer literacy including minimal keyboard skills and experience interacting with a browser application prior. No advanced knowledge of computer systems is needed. A user should be able to learn the user interface swiftly. Additionally, raters must have sufficient professional knowledge regarding the questions that are asked in the feedback form and students must be able to understand the feedback and analysis given.

2.4 Constraints

The software is to be built using JavaScript/HTML. Any device with access to a modern web browser will be able to access and run this application assuming that the device has a monitor/screen to view/interact with the Graphical User Interface (GUI). An internet connection is required to access the web application.

2.5 Assumptions

The students, raters, and administrators are assumed always to have the correct information to access their respective accounts and the hardware capability to input that information. It is also assumed that the database being used is or will be heavily adopted and will likely have support in the near future to ensure an up-to-date system. The application is also assumed to be deployed in an environment with a stable network.

3 Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

※ ⟹ TBD

3.1.2 Hardware interfaces

The hardware interface will be any device, such as a computer or equivalent, that is capable of accessing the internet through a modern web browser.

3.1.3 Software interfaces

3.1.3.1 User authentication and authorization

※ ⟹ The system must allow user authentication and authorization features to ensure proper access. The system will interface with the Penn State authentication system for user role validation, which includes password access and two-factor authentication. (Confirm) Users will only be able to view data and perform functions as permitted by their authorization level.

3.1.3.2 Database management system

※ ⟹ Exact architecture TBD. The database must be able to retain data for a minimum of two (2) years.

3.1.3.3 Web application framework

※ ⟹ TBD

3.1.3.4 Server

※ ⟹ TBD

3.2 Functional requirements

In this section, "the system" refers generally to the CCC.

3.2.1. The system **MUST** allow each user to authenticate and log in to access their role-based permissions.

Importance: ■ HIGH

3.2.2. Administrators MUST be able to manage user roles for students, raters, and other administrators.

Importance: ■ HIGH

Dependencies: 3.2.1

3.2.3. The system MUST provide a feedback form for raters to assess the competencies of a particular student.

Importance: ■ HIGH

※ ⟹ 3.2.4. The system MUST generate supplemental questions based on gaps in form responses received from raters.

Importance: ■ HIGH

Dependencies: 3.2.3

3.2.5. The rater MUST be able to submit assessments of students via a feedback form.

Importance: ■ HIGH

Dependencies: 3.2.3

3.2.6. The student MUST be able to request a rater to fill out a feedback form.

Importance: ■ HIGH

Dependencies: 3.2.5

3.2.7. The system MUST notify raters when a student requests feedback.

Importance: MEDIUM

Dependencies: 3.2.6

3.2.8. The system MUST remind raters if feedback forms are not completed within a specified time.

Importance: MEDIUM

Dependencies: 3.2.7

3.2.9. The system MUST notify students when a rater has completed a feedback form for them.

Importance: LOW

Dependencies: 3.2.5

※ ⟹ 3.2.10. Students MUST NOT be able to view the rater for any form response.

Importance: HIGH

Dependencies: 3.2.5

3.2.11. The system MUST store form responses in a database for future access.

Importance: HIGH

Dependencies: 3.2.5

3.2.12. Administrators MUST be able to view all form responses.

Importance: HIGH

Dependencies: 3.2.11

3.2.13. Administrators MUST be able to view who submitted each form response.

Importance: HIGH

Dependencies: 3.2.12

3.2.14. The system MUST be able to generate a full report for each specific student.

Importance: HIGH

Dependencies: 3.2.11

3.2.15. The full report of a student MUST describe the level of competency they have been assessed to exhibit in accordance with EPAs.

Importance: HIGH

Dependencies: 3.2.11, 3.2.14

3.2.16. Students **MUST NOT** be able to view any other student's full report.

Importance: HIGH

Dependencies: 3.2.14

3.2.17. Administrators MUST be able to view every student's full report.

Importance: HIGH

Dependencies: 3.2.14

3.2.18. The system MUST be able to detect low-quality and unhelpful form responses, flag the corresponding responses and raters, and notify administrators.

Importance: ■ LOW

Dependencies: 3.2.11

3.2.19. Administrators MUST be able to view raters flagged for low-quality or unhelpful form responses.

Importance: ■ LOW

3.2.20. Administrators MUST be able to filter and remove low-quality or unhelpful form responses.

Importance: ■ LOW

Dependencies: 3.2.19

3.2.21. The software MUST be able to export data via PDF and Excel.

Importance: ■ LOW

Dependencies: 3.2.11

3.3 Non-functional requirements

3.3.1 Performance requirements

The system should have a response time of less than two seconds for all user interactions, including loading pages, submitting maintenance requests, and retrieving historical data. The system should also be able to handle concurrent usage by at least 100 guests without significant degradation of performance. The system should be designed to have an uptime of 99.9%. In the event of unforeseen issues or system maintenance, the system should be able to inform guests of the system status and an estimated resolution time. Finally, the system should utilize system server resources efficiently, ensuring that CPU and memory usage remains within acceptable limits under normal operating conditions.

3.3.2 Design constraints

The system should be accessible via all common web browsers such as Google Chrome, Firefox, Safari, and Microsoft Edge. The system should also be accessible from a wide range of devices such as laptops, smartphones, and tablets.

3.3.3 Regulatory compliance

The system must comply with all privacy regulations at Hershey Medical Center and Penn State Health as a company. The system must also comply with all relevant regulations set forth by the US Government.

3.3.4 Backup and recovery

The system should have a backup and recovery plan to protect data integrity in the event of system failure or data loss. Regular automated backups should be scheduled, with clear procedures for data restoration.

3.3.5 User training and documentation

User training and documentation should be provided to ensure that all users, including students, raters, and administrators, can navigate and use the system. This may include user manuals, FAQS, and/or training manuals.

Client signature
Anthony Dambro

Date

Appendix A

System models

A.1 Context model

※ ⟹ tbd

A.2 Use case diagram

※ ⟹ tbd

Appendix B

Software engineering tools

※ ⟹ tbd