

CMMI® Appraisal System

Report detailing the develpoment and deployment of a bespoke CMMI Appraisal System

Samuel Moore | Computing Systems

## Abstract

The inexorable and rapid onset of the digital age has resulted in organisations producing exponentially growing amounts of software. Software lies at the heart of modern society; from the devices in our pockets, to the satellites orbiting the earth. Software, in recent times, has taken on an absolutely pivotal role in society. With this vital importance in mind, the drive for high-quality software is of utmost importance for every organisation in the world today.

Quality software is not achieved through pot luck, uniquely talented individuals, or all-star teams. Rather, quality software is produced when quality processes are used to govern it during development and deployment. Software must rely on the strength of the process, rather than on the strength of the individual. Each piece of software is a complex tapestry, with each small element needing to be controlled and monitored in order to ensure the highest quality outcome. It should not be a series of individuals who control and monitor these elements, but rather well-designed and robust processes.

This project seeks to leverage the CMMI®, a well-known framework for process improvement, in a unique way - by combining it with repertory grid technique. This combination is intended to elicit tacit notions that otherwise may remain undetected when analysing processes for improvement. Moreover, this project explores the development and creation of a systematic solution, by developing a piece of software which successfully implements the two approaches, culminating in a feature-rich piece of software capable of instigating a cycle of improvement for any organisation.

## Acknowledgements

I would like to take this opportunity to offer my sincerest thanks to everyone involved in making this project a success.

Firstly, to the lecturing and teaching staff at Ulster University, for offering five years of excellent learning, which ultimately has surfaced in this dissertation project. Thank you for the consistent support and learning provided along my academic journey here at Ulster University.

I would also like to thank my peer support group; Neil, Sam and Rebecca for their valuable support on many group projects through my university career, but most notably in this dissertation project – undoubtedly the most challenging thus far.

I must also thank some key personnel at Citibank, without whom this project would not have happened. First of all; Mark McCormack, for allowing me to leverage his network to find a suitable team to use as a pilot for this project. Thanks to Barry Lau, who leads the surveillance securities team for allowing me to conduct the pilot on his team. Thanks also to, Rosemary Teggart, Chris Clokey, David Albery and Jonathan King for providing peer review throughout many stages of the project – this input was invaluable and helped shape the final outcome.

Some special and sincere thanks must also go to my supervisor, Don McFall, who has provided the highest level of support throughout this entire project. Don, thank you for constantly challenging me, for your availability for meetings amid a busy schedule, and for your expert steer provided during the project.

Lastly, I would like to thank my friends and family, most notably my fiancée Amy, who has been patient and supportive throughout this very large project. Thank you for your unwavering belief in everything I set out to do in my career, both academic and professional.

Contents

[Abstract 1](#_Toc511938959)

[Acknowledgements 2](#_Toc511938960)

[1. Introduction 13](#_Toc511938961)

[1.1 Background & Motivation for Project 13](#_Toc511938962)

[1.2 Context of the Project 14](#_Toc511938963)

[1.3 Project Aims and Objectives 15](#_Toc511938964)

[2. Literature Review 17](#_Toc511938965)

[2.1 CMMI® for Development v1.6 17](#_Toc511938966)

[2.2 Repertory Grid Technique 21](#_Toc511938967)

[2.3 How Repertory Grid Technique (RGT) Works 22](#_Toc511938968)

[2.4 Tools & Techniques – Lifecycles 23](#_Toc511938969)

[2.4.1 The IDEAL Framework 23](#_Toc511938970)

[2.5 Development Lifecycles 25](#_Toc511938971)

[2.5.1 The Waterfall Model 26](#_Toc511938972)

[2.5.2 Iterative Model 27](#_Toc511938973)

[2.5.3 Spiral Model 27](#_Toc511938974)

[2.5.4 Agile Model 28](#_Toc511938975)

[2.5.5 Choice of Development Life Cycle 29](#_Toc511938976)

[2.6 Literature Review Summary 30](#_Toc511938977)

[3. Requirements Capture & Analysis 31](#_Toc511938978)

[3.1 Elicit Needs 31](#_Toc511938979)

[3.1.1 Citibank Internal Audit 32](#_Toc511938980)

[3.2 Observation: The Need for a CMMI Appraisal System 32](#_Toc511938981)

[3.3 Interview: Citi Technology Audit 34](#_Toc511938982)

[3.4 Interrogation of Internal Audit’s Issue Database 35](#_Toc511938983)

[3.4.1 Choice of Process Areas to be Covered in CMMI Appraisal Tool 37](#_Toc511938984)

[3.5 Listing of Customer Requirements 37](#_Toc511938985)

[3.6 Develop Product Requirements 38](#_Toc511938986)

[3.7 Requirements Analysis & Validation 39](#_Toc511938987)

[3.8 Acceptance Tests for Requirements 42](#_Toc511938988)

[3.9 Kano Requirements Analysis 44](#_Toc511938989)

[3.9.1 Surprise and Delight Requirements 45](#_Toc511938990)

[3.9.2 More is Better Requirements 45](#_Toc511938991)

[3.9.3 Must Be Requirements 45](#_Toc511938992)

[3.9.4 Better Not Be Requirements 46](#_Toc511938993)

[3.9.5 Leveraging Kano Analysis in the CMMI Appraisal Project 46](#_Toc511938994)

[3.10 Requirements Weighting 47](#_Toc511938995)

[4. Design 49](#_Toc511938996)

[4.1 Model View Controller (MVC) 49](#_Toc511938997)

[4.2 Architecture Design 49](#_Toc511938998)

[4.2.1 User Devices 49](#_Toc511938999)

[4.2.2 MySQL Database 49](#_Toc511939000)

[4.2.3 Google Charts API 50](#_Toc511939001)

[4.2.4 HTML & CSS 50](#_Toc511939002)

[4.3 Interface Design (View) 51](#_Toc511939003)

[4.3.1 Use Case Model 51](#_Toc511939004)

[4.3.2 Wireframe Designs 51](#_Toc511939005)

[4.4 Entity Relationship Design (Model) 51](#_Toc511939006)

[4.5 System Behaviours (Controller) 53](#_Toc511939007)

[5. Implementation 54](#_Toc511939008)

[5.1 Website Structure & Appearance 54](#_Toc511939009)

[5.1.1 Using Bootstrap to Display Page Content 55](#_Toc511939010)

[5.1.2 Creating a Mobile-friendly Menu 55](#_Toc511939011)

[5.2 Implementation of MySQL Database 57](#_Toc511939012)

[5.2.1 Connecting to MySQL via PHP 57](#_Toc511939013)

[5.3 Authentication 58](#_Toc511939014)

[5.3.1 Registration Process 58](#_Toc511939015)

[5.3.2 Log-in Process 59](#_Toc511939016)

[5.3.3 Use of Permissions to Protect Sensitive Webpages 62](#_Toc511939017)

[5.4 Appraisal Interface 64](#_Toc511939018)

[5.4.1 Appraisal Interface Sub-menu 65](#_Toc511939019)

[5.4.2 Storing Appraisal Data 67](#_Toc511939020)

[5.4.3 Populating the Form with Saved Responses 69](#_Toc511939021)

[5.4.4 Displaying a Dynamic Practice Score 72](#_Toc511939022)

[5.4.5 Appraisal Results Overview 73](#_Toc511939023)

[5.4.6 Manage Appraisals Interface 76](#_Toc511939024)

[5.5 Google Charts API 78](#_Toc511939025)

[5.5.1 Making Google Charts Dynamic 79](#_Toc511939026)

[5.5.2 Generating the Process Capability Bar Chart 79](#_Toc511939027)

[5.5.3 Bar & Column Charts 83](#_Toc511939028)

[5.5.4 Preparing Data for the Scatter Chart 86](#_Toc511939029)

[5.5.5 Adding Event Listeners to Charts 88](#_Toc511939030)

[5.5.6 Comparison Dashboard 88](#_Toc511939031)

[5.6 Export Data Functionality 95](#_Toc511939032)

[5.7 Github & Atlassian 97](#_Toc511939033)

[6. Testing, Verification & Validation 98](#_Toc511939034)

[6.1 Testing Strategy 98](#_Toc511939035)

[6.2 Unit Testing 99](#_Toc511939036)

[6.3 Peer Reviews 99](#_Toc511939037)

[6.4 Requirements Acceptance Criteria 101](#_Toc511939038)

[6.4.1 Requirements not Met, or Partially Met 103](#_Toc511939039)

[7. Appraisal Execution 104](#_Toc511939040)

[7.1 Observations & Actions 104](#_Toc511939041)

[7.1.1 Observation 1: Requirements analysis activities should be mandatory 104](#_Toc511939042)

[7.1.2 Observation 2: The process for obtaining commitment to requirements, and the process for changing requirements are both very strong 106](#_Toc511939043)

[7.1.3 Observation 3: Requirements progress should be tracked through JIRA 107](#_Toc511939044)

[7.1.4 Conclusion & Next Steps 108](#_Toc511939045)

[8. Evaluation 109](#_Toc511939046)

[8.1 Project Aim 109](#_Toc511939047)

[8.1.1 Justification 109](#_Toc511939048)

[8.2 Project Objectives 109](#_Toc511939049)

[8.3 Areas of Strength 110](#_Toc511939050)

[8.3.1 Volume of Work 111](#_Toc511939051)

[8.3.2 Quality of Research 111](#_Toc511939052)

[8.3.3 Novel Approach 112](#_Toc511939053)

[8.3.4 Project Management 112](#_Toc511939054)

[8.3.5 Implementation of Google Charts API 112](#_Toc511939055)

[8.4 Areas for Improvement 114](#_Toc511939056)

[8.5 Future Development 114](#_Toc511939057)

[9. Conclusion 115](#_Toc511939058)

[References 116](#_Toc511939059)

Listing of Figures

[Figure 1 - Staged Representation of the CMMI [1] 15](#_Toc511676248)

[Figure 2 - Continuous Representation of the CMMI [1] 16](#_Toc511676249)

[Figure 3 - A repertory grid example [7] 19](#_Toc511676250)

[Figure 4 - The IDEAL Framework [8] 20](#_Toc511676251)

[Figure 5 - Waterfall Model [10] 22](https://d.docs.live.net/040a99f76da92c51/Documents/University/COM547%20Documentation/CMMI%20Appraisal%20System%20Final%20Report%20v9.docx#_Toc511676252)

[Figure 6 - Iterative Model [22] 23](https://d.docs.live.net/040a99f76da92c51/Documents/University/COM547%20Documentation/CMMI%20Appraisal%20System%20Final%20Report%20v9.docx#_Toc511676253)

[Figure 7 - Spiral Model [13] 25](#_Toc511676254)

[Figure 8 - Agile Model [14] 26](#_Toc511676255)

[Figure 9 - REQM Issues Raised 2011-2017 33](#_Toc511676256)

[Figure 10 - CM Issues Raised 2011-2017 33](#_Toc511676257)

[Figure 11 - Results of Kano analysis 44](#_Toc511676258)

[Figure 12 - Relative Weighting of Requirements 45](#_Toc511676259)

[Figure 13 - Architecture Diagram 47](#_Toc511676260)

[Figure 14 - Normalised View 49](#_Toc511676261)

[Figure 15 – zero-NF 49](#_Toc511676262)

[Figure 16 - wireframe for home page versus actual result 51](#_Toc511676263)

[Figure 17 - Bootstrap grid system 52](#_Toc511676264)

[Figure 18 - Menu on desktop device 53](#_Toc511676265)

[Figure 19 - Mobile menu 53](#_Toc511676266)

[Figure 20 - Expanded mobile menu 53](#_Toc511676267)

[Figure 21 - CSS Media Query 53](https://d.docs.live.net/040a99f76da92c51/Documents/University/COM547%20Documentation/CMMI%20Appraisal%20System%20Final%20Report%20v9.docx#_Toc511676268)

[Figure 22 - Connect to MySQL via PHP 54](#_Toc511676269)

[Figure 23 - User registration process 55](#_Toc511676270)

[Figure 24 - Log in process on home page 56](#_Toc511676271)

[Figure 25 - login function 57](#_Toc511676272)

[Figure 26 - check\_logged\_in function 58](#_Toc511676273)

[Figure 27 - checked\_logged\_in in use 58](#_Toc511676274)

[Figure 28 - Functions to check permission levels 59](#_Toc511676275)

[Figure 29 - Administrator menu 60](#_Toc511676276)

[Figure 30 - Manage Users 60](#_Toc511676277)

[Figure 31 - Appraisal interface, desktop 61](#_Toc511676278)

[Figure 32 - Appraisal interface, mobile 61](#_Toc511676279)

[Figure 33 - Appraisal sub-menu 62](#_Toc511676280)

[Figure 34 - Parameters built into destination URL 62](#_Toc511676281)

[Figure 35 - Menu parameter in URL 62](#_Toc511676282)

[Figure 36 - Case statement for URL parameter 63](https://d.docs.live.net/040a99f76da92c51/Documents/University/COM547%20Documentation/CMMI%20Appraisal%20System%20Final%20Report%20v9.docx#_Toc511676283)

[Figure 37 - display\_completion\_tick function 63](#_Toc511676284)

[Figure 38 - get\_answer function 64](#_Toc511676285)

[Figure 39 - Code to store answers from a submitted appraisal form 65](#_Toc511676286)

[Figure 40 - store\_answer function 66](#_Toc511676287)

[Figure 41 - HTML & PHP to display repertory grid option 67](#_Toc511676288)

[Figure 42 - display\_options function 68](#_Toc511676289)

[Figure 43 - option\_selected function 68](#_Toc511676290)

[Figure 44 - Code to display strength & weakness boxes 69](#_Toc511676291)

[Figure 45 - display\_strwk\_boxes function 69](#_Toc511676292)

[Figure 46 - Live score for specific practice 69](https://d.docs.live.net/040a99f76da92c51/Documents/University/COM547%20Documentation/CMMI%20Appraisal%20System%20Final%20Report%20v9.docx#_Toc511676293)

[Figure 47 - JavaScript to update live score 70](#_Toc511676294)

[Figure 48 - Results overview, capability chart 71](#_Toc511676295)

[Figure 49 - Results overview, strengths and weakness table preview. 71](#_Toc511676296)

[Figure 50 - get\_str\_wk function 72](#_Toc511676297)

[Figure 51 - display\_strwk\_row function 72](#_Toc511676298)

[Figure 52 - Manage appraisals interface 73](#_Toc511676299)

[Figure 53 - get\_appraisal\_table function 74](#_Toc511676300)

[Figure 54 - Dynamic search function JavaScript code 74](#_Toc511676301)

[Figure 55 - Importing Google Charts API 75](#_Toc511676302)

[Figure 56 - An example of static Google Chart data [19] 76](#_Toc511676303)

[Figure 57 - Code to display process capability chart 77](#_Toc511676304)

[Figure 58 get\_rm\_capability function 78](#_Toc511676305)

[Figure 59 - get\_rm\_capability\_remainder function 79](#_Toc511676306)

[Figure 60 - get\_rm\_capability\_label function 79](#_Toc511676307)

[Figure 61 - Column chart in Results Dashboard page 80](#_Toc511676308)

[Figure 62 – Data declaration for bar chart 81](#_Toc511676309)

[Figure 63 - get\_rm\_capability\_avg function 82](#_Toc511676310)

[Figure 64 - get\_column\_avg function 83](#_Toc511676311)

[Figure 65 - Scatter chart and table 83](#_Toc511676312)

[Figure 66 - populate\_scatter\_chart function 84](#_Toc511676313)

[Figure 67 - get\_rm\_capability\_score function 84](#_Toc511676314)

[Figure 68 - Google Chart event listener code 85](#_Toc511676315)

[Figure 69 - Comparison dashboard charts 86](#_Toc511676316)

[Figure 70 - Comparison dashboard table 86](#_Toc511676317)

[Figure 71 - populate\_dropdown1 function 87](#_Toc511676318)

[Figure 72 - Code to extract user choices in comparison dashboard 88](#_Toc511676319)

[Figure 73 - get\_project\_name function 89](#_Toc511676320)

[Figure 74 - Code to draw the custom line/bar chart 90](#_Toc511676321)

[Figure 75 - get\_comparison\_table function 91](#_Toc511676322)

[Figure 76 - display\_comparison\_table\_row function 92](#_Toc511676323)

[Figure 77 - export\_data function 93](#_Toc511676324)

[Figure 78 - Using Atlassian 94](#_Toc511676325)

[Figure 79 - Sample unit test 96](#_Toc511676326)

[Figure 80 - CSS Project - RM SP1 102](#_Toc511676327)

[Figure 81 - CSS Project - RM SP2 103](#_Toc511676328)

[Figure 82 - CSS Project - RM SP3 104](#_Toc511676329)

[Figure 83 - CSS Project - RM SP4 105](#_Toc511676330)

[Figure 84 - Use of Google Charts API 110](#_Toc511676331)

[Figure 85 - Google Charts API in Results Dashboard 110](#_Toc511676332)

Listing of Tables

[Table 1 - Customer Requirements 34](#_Toc511676242)

[Table 2 - Functional requirements 35](#_Toc511676243)

[Table 3 - Non-functional Requirements 36](#_Toc511676244)

[Table 4 - Requirements Quality Criteria 37](#_Toc511676245)

[Table 5 - Requirements Quality Criteria Results 38](#_Toc511676246)

[Table 6 - Acceptance tests for functional requirements 39](#_Toc511676247)

Appendices

|  |  |  |
| --- | --- | --- |
| **Appendix A** | Glossary of Terms | Page 120 |
| **Appendix B** | Volere Shells | Page 123 |
| **Appendix C** | Use Case Diagram | Page 131 |
| **Appendix D** | Wireframes | Page 132 |
| **Appendix E** | Sequence Diagrams | Page 137 |
| **Appendix F** | SQL Code | Page 141 |
| **Appendix G** | Unit Testing | Page 147 |

# 1. Introduction

**The purpose of the project is to create a CMMI®-DEV v1.3 (Capability Maturity Model Integration for Development) appraisal system for Citi Belfast. The project will ultimately provide Citi with; a holistic view of capability in several key process areas, the ability to benchmark results across departments, and the production of strength and weakness documents for teams that use the system.**

## 1.1 Background & Motivation for Project

Citigroup Inc. is an American multinational investment banking and financial services corporation, with a presence in more than 160 countries and currently employing more than 200,000 employees. Citi Belfast, which is based on the Sydenham Road in the centre of Belfast, is a key service centre to the bank globally. The Belfast hub features a vast array of second and third line functions for the bank. Citi Belfast is a key location for technology services.

While Citi has received a CMMI-Dev level 3 Maturity certification on a restricted scope of projects using the staged representation of the CMMI model, there is no tool or system within the bank to assess the capability levels of the process areas for individual projects throughout the organisation using the continuous model of CMMI. The absence of such a system means that team leads are unable to assess the individual process capabilities of their team, or benchmark them against other teams in Citi. Team leaders are also unable to identify a “roadmap to improvement” for their specific team, because the current CMMI rating is granted at an organisational level – meaning that the idiosyncrasies of their team with regards to process capability may not be accounted for or addressed. Currently, any process reviews at a team-level are done on an ad-hoc basis and do not follow any particular framework or model. This lack of standard approach leads to inconsistency, and an inability to compare results across teams.

Currently it is not possible for individual projects to seek a CMMI appraisal in a chosen process area, meaning that it is difficult for regional and divisional heads to produce an informed strategy with regards to process improvement, given that they are **lacking detail on a team-by-team basis**.

## 1.2 Context of the Project

The current situation demonstrates an obvious shortcoming with regards to continuous improvement at Citi. This project aims to create a CMMI appraisal system to assess processes using the continuous representation. This project would create a prototype, using Citi Technology Belfast as the target audience and use only 2-3 key process areas as an initial proof of concept. This in mind, the system should be designed with a larger scope in mind, so that it could easily be expanded in future to include all process areas of CMMI, and be applicable to other organisations beyond Citi. The system that this project will create should ideally solve the consistency issues across Citi with regards to continuous improvement.

The new CMMI system would allow project experts within Citi to log into an online system and answer scaled questions regarding a team’s process capability. A bespoke algorithm would be incorporated into the system to calculate a team’s initial capability level and suggest improvements. These improvements will be based upon strengths and weaknesses that the assessor can enter into the system as the assessment progresses, which will help build a more qualitative aspect into the system. The system will include a database with querying power, which should be able to draw out benchmarking information for department leads to use from a strategic standpoint. The system should also be developed to be compatible on mobile devices, meaning that a practitioner could simply carry a tablet to a stakeholder meeting to record process information, or display reports on the spot.

## 1.3 Project Aims and Objectives

**The aim of the project is to create a CMMI appraisal tool which generates sub-practice-based questions to establish the extent to which a specific practice is enacted. The extent scale shall be based on a classification technique known as repertory grid. The tool will establish strengths and weaknesses based on the body of knowledge contained within the CMMI.**

The main objectives of the project are:

1. Project Analysis & Planning

* Analyse risks, costs, personnel and skills required to complete the project
* Decide upon a project lifecycle and create a detailed project plan with dated milestones.
* Draft an initial scope of requirements and conduct prioritisation according to the Kano method. Using this method, ‘delighters’, ‘satisfiers’ and ‘basic expectations’ shall be identified.
* Design project success metrics
* Create WBS and Gantt chart.
* Conduct detailed research into process areas

1. Requirements Gathering

* Engage with all necessary stakeholders to solicit requirements
* Conduct final prioritisation of requirements, and obtain sign off.
* Generate functional requirements document.

1. Design

* Produce use-case diagrams which detail user and system interactions
* Design and storyboard the system
* Create sub-practice questionnaire
* Design algorithm scoring method

1. Implementation

* Design front-end website
* Write code to support the features made in the design phase
* Implement code into front-end website
* Implement SQL database and write queries as appropriate

1. Testing

* Develop test plan and execute it.
* Record exceptions and introduce fixes
* Perform regression testing where necessary

1. Pilot

* Deploy product to pilot group
* Gather feedback, and use feedback to update metrics
* Review product and make any necessary changes post-pilot

1. Deployment

* Launch the product into its final deployment environment

1. Documentation

* Create user guide
* Complete project documentation

1. Review & Evaluation

In the next section of this report we will conduct a thorough literature review focusing on repertory grid techniques, CMMI and systems development lifecycles. This will be followed by the requirements capture and analysis chapter, the design and development chapter, implementation and testing, appraisal execution, an evaluation of the project, and finally, the conclusion.

# 2. Literature Review

## 2.1 CMMI® for Development v1.6

The CMMI (Capability Maturity Model Integration) for development is a collection of best practices aimed at helping organisations improve their process capability, and as a result, produce higher quality work products. The CMMI-DEV model, which is derived from the CMMI, provides guidance for applying CMMI best practices in a development organization. CMMI-DEV model practices focus on the activities of the developer organization. Five process areas focus on practices specific to development: addressing requirements development, technical solution, product integration, verification, and validation. [1] Margarito et al. [2] noted that “*Organisations that implemented CMMI typically improve performance in terms of quality of products and processes, schedule and costs. Consequently, processes become more predictable and customer satisfaction increases*”. The predictability of a process area becomes a key part of the CMMI, especially when implementing improvements, because improvement cannot be measured on an unpredictable process. In order to survive in the competitive market, software companies must optimise their processes so that the process is less dependent on people, but rather on the standardised set of processes. [3]

The CMMI-DEV contains 22 process areas in total. Of these process areas, 16 are core process areas (meaning that they are common across all CMMI models), 1 is a shared process area (meaning that is shared by at least two CMMI models, but not all of them) and 5 are development specific process areas, these are; requirements development, technical solution, product integration, verification and validation.

The process areas documented in the CMMI are not prescriptive instructions on how an organisation should perform their processes, rather, they provide guidance and best practice to use when companies are developing their processes. This notion is outlined clearly by the Software Engineering Institute (SEI) – “*Like other CMMs, CMMI models provide guidance to use when developing processes. CMMI models are not processes or process descriptions. The actual processes used in an organization depend on many factors, including application domains and organization structure and size. In particular, the process areas of a CMMI model typically do not map one to one with the processes used in your organizatio*n” [1].

A key part of the CMMI is the use of levels. Levels are known either as a maturity level (ML), or a capability level (CL) – the type of level used will depend upon which representation of CMMI has been implemented. The SEI describes levels as “*evolutionary path recommended for an organization that wants to improve the processes it uses to develop products or services*” [1]. The CMMI offers two paths of improvement;

1. Staged representation (Figure 1), which will award a Maturity Level (ML) to the company as a whole
2. Continuous representation (Figure 2), which will award a Capability Level (CL) to the individual process area(s) being assessed.

Figure 1 - Staged Representation of the CMMI [1]

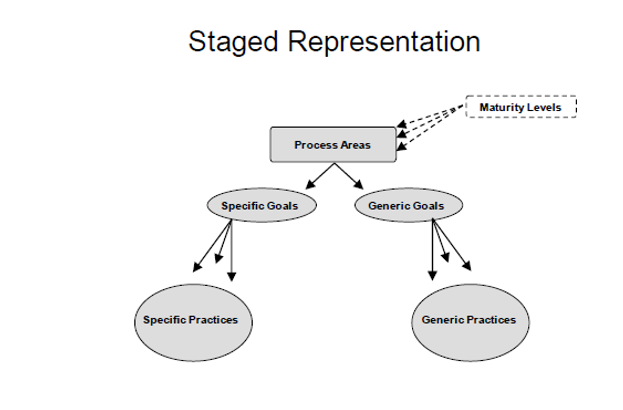
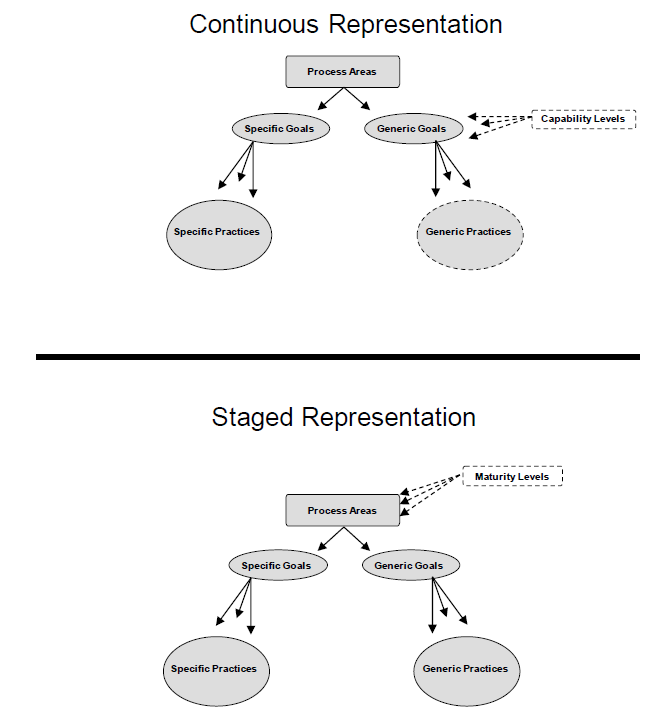


Figure 2 - Continuous Representation of the CMMI [1]



The staged representation of the CMMI will assess the organisations maturity as a whole by looking at the full set of process areas and assigning a process area. Conducting a full staged appraisal can be expensive and time consuming, however, the benefits of having been assessed up to ML3 come to light in terms of not only the company’s process capability, but also in being able to use this high maturity when bidding for work, in particular from public sector organisations.

The continuous representation allows companies to improve their processes incrementally by looking at a smaller set of chosen process areas that are the most pertinent to the company. The continuous representation is particularly beneficial for companies seeking to gain improvement quickly without having to spend large sums of money on the staged representation.

Given the nature of this project, the continuous representation will be the implementation of choice. This is largely driven by restrictions on time and resources and the fact that this is a student-led project, where a staged representation would require an experienced and certified assessor.

In determining the level of a capability for a process area, the CMMI has designed goals and practices. Ultimately, generic goals are what determine the capability level of the process area, but these generic goals are influenced directly from the specific and generic practices. It is the “practices” that describe in greater detail the expectations surrounding the process area. It should be noted, that a generic or specific goal is not considered enacted unless there is evidence that the specific or generic practices are enacted in the organisation.

Using the continuous representation, there are 4 capability levels which describe the extent to which a process area has been enacted:

* Incomplete
  + The specific goals of the process area are only partially performed, or not at all performed
* Performed
  + The specific goals of the process area are performed
  + This means that the “required” components (i.e. the specific goals) have all been met, and the “expected” components (i.e. the specific practices) are all being performed, or suitable alternatives have been identified.
* Managed
  + The process is performed
  + The required components for a managed process are met, meaning that the process is *“planned and executed in accordance with policy; employs skilled people having adequate resources to produce controlled outputs; involves relevant stakeholders; is monitored, controlled, and reviewed; and is evaluated for adherence to its process description.”* [1]
* Defined
  + The process is managed
  + The required components for a defined process are met, meaning that the process is “*tailored from the organization’s set of standard processes according to the organization’s tailoring guidelines; has a maintained process description; and contributes process related experiences to the organizational process assets.”* [1]

## 2.2 Repertory Grid Technique

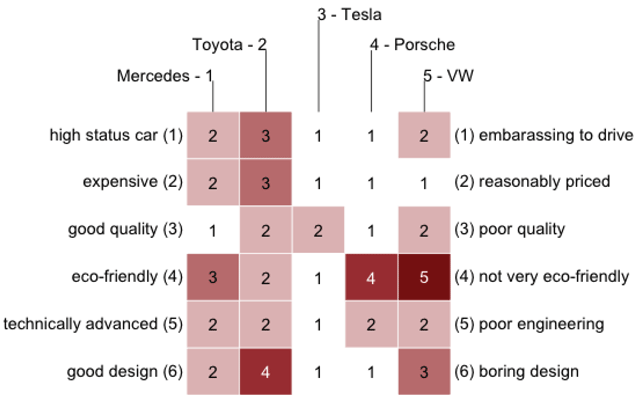
The assessment of whether or not a project sufficiently enacts the generic and specific practices of the CMMI is the cornerstone of the CMMI appraisal system. Without a robust method of identifying the extent to which a practice is enacted the system cannot be deemed to be fulfilling its primary aim. The assessment of whether or not a process has been enacted will be achieved through a series of sub-practice based questions with regards to the project. In theory, it would be possible to simply present a list of binary yes/no questions regarding whether or not the specific or generic practice has been enacted for each given sub-practice. This, however, would not be in the intended spirit of the system and lacks insight. While this kind of system would be straight-forward to implement, it does not serve as a robust method for establishing if a specific or generic practice has been enacted. The primary reason for this is that, by only presenting a binary yes/no question, the assessment functionality would fall victim to ambiguity generated by individuals’ unique and biased viewpoints. Niu and Easterbrook [4] describe this phenomenon eloquently in the opening line of their study into repertory grid: “*People filter their observations of the world according to their interests, each person using his or her own conceptual framework*” going on to state that people “*often express themselves using ambiguous and conflicting terms*”. Niu and Easterbrook cite George Kelly’s Personal Construct Theory as the basis for this argument, which describes how individuals build “personal constructs” to help them make sense of an environment. [5]

An example of this can be conceived quite clearly when discussing the extent to which a specific practice has been met. Specific practice 1.1 of REQM in the CMMI states “Understand Requirements”. There is a natural ambiguity associated with this specific practice - how is the understanding of requirements assessed here? One assessor may use stringent criteria to determine if this is enacted, where another assessor may be inclined to be lenient based on some innate knowledge that he or she possesses. As such, it is necessary to use repertory grid techniques here to elicit tacit knowledge held by the assessor. The Repertory Grid Technique (RGT) forms an essential ingredient in unlocking this puzzle. Leon and Guild [6] state that “*repertory grid seems to be a tool able to bring the experts’ self-constructed knowledge to the surface – making explicit the valuable heuristic knowledge that experts possess but are frequently unable to articulate*”. Fundamentally, what this means is that RGT can be used to translate an expert’s “hunch” into reliable quantitative information, while avoiding subjectivity and ambiguity.

### 2.3 How Repertory Grid Technique (RGT) Works

The RGT relies upon presenting constructs to the user to establish how the user views a given object in a chosen area. In RGT a construct is a bipolar dimension representing opposing poles with a rating of 1-5 representing the scale of where the object falls on the pole. For example, with regards to criteria lists for user requirements, one pole of the construct may be “no criteria exists”, and the other end of the pole may be “criteria is documented, and always used”. The middle ground between these two poles may be something akin to “criteria exists, but is not always used”, although this middle ground is not displayed as a part of the construct. Presenting this construct in a grid with other constructs related to the understanding of requirements will help the assessor establish if requirements truly are understood, without relying on subjective judgement. It also means that when the grid is re-used on another project by a different assessor, the same objective reasoning is applied each time to determine the outcome.

Figure 3 - A repertory grid example [7]



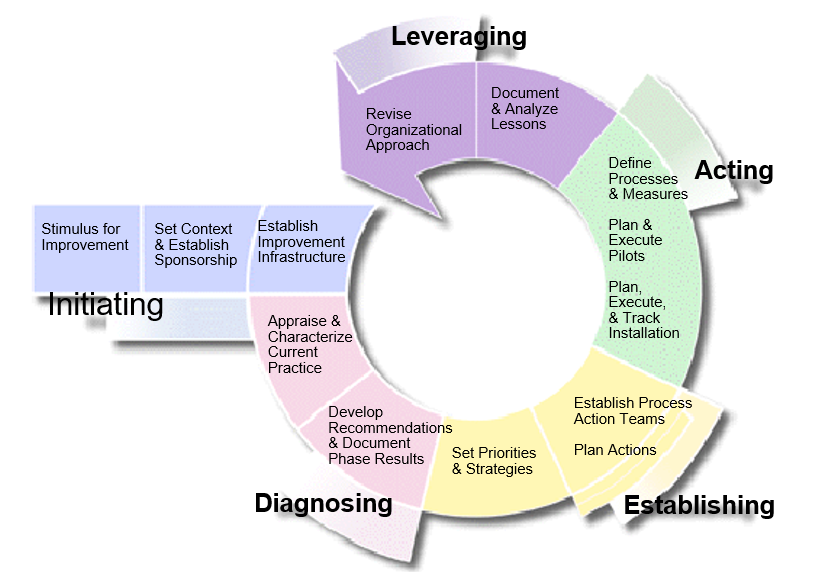
## 2.4 Tools & Techniques – Lifecycles

This project represents a fairly unique undertaking in that it is more accurately a project within a project. As such, it is appropriate to employ two separate lifecycles to achieve the desired result. The overarching goal is to conduct a CMMI appraisal on Citi and analyse the findings, and in order to achieve this goal it is necessary to build a bespoke tool to collect quality information. Therefore, there needs to be a lifecycle to complete the appraisal and analysis, and within that, a development lifecycle to design, implement and test a piece of software.

### 2.4.1 The IDEAL Framework

IDEAL, which stands for Initiate, Diagnose, Establish, Act, Leverage is a Software Process Improvement (SPI) framework developed by the Software Engineering Institute. In the opening remarks of his handbook, McFeeley [8] states that IDEAL “*can be used to guide development of a long-range, integrated plan for initiating and managing a SPI program*”. The IDEAL framework represents a phased approach to implementing software process improvement, consisting of five key phases.

Figure 4 - The IDEAL Framework [8]



#### Phase 1: Initiating

The initiating phase is where the IDEAL framework begins. The initiating phase largely puts in place the building blocks and organisational support required to get the project underway. It begins with establishing the business rationale for undertaking the SPI in the first-place. While this may seem an obvious task, outlining these business rationales form the basis of the work going forward, much like a “problem statement”, it describes the work that needs to be carried out in the SPI. These business rationales will then inform the general goals of the SPI, which are later refined during the Establish phase.

The initiating phase is also responsible for outlining the improvement infrastructure. Within this improvement infrastructure; roles and responsibilities are assigned and initial resources are assigned.

#### Phase 2: Diagnosing

McFeeley [8] states that “*The Diagnosing phase of the IDEAL model starts the organization on the path of continuous software process improvement*”. In this phase, the SPI action plan is initiated, which will incorporate lessons learned from previous improvement projects. This phase is key for understanding the current processes, and with regards to CMMI, will involve some scoping to determine which process areas should be included in the improvement project. This phase focuses primarily on diagnosing strengths and weaknesses.

#### Phase 3: Establishing

In the Establishing phase the problems or areas that the organisation has decided to address are prioritised and strategies to achieve improvement in these areas are developed. In this phase, the general goals that were established in the Initiating phase are now refined into specific measurable goals. In accordance with these goals, metrics are also developed during this phase to monitor the progress of these goals.

#### Phase 4: Acting

The Acting phase is really where actions to effect change in the organisation are taken in order to make sustained improvements. In this phase, solutions for improvement that were identified in the Diagnosing phase are created, piloted and deployed.

The implementations of the improvements are managed and tracked to ensure success. Once tests have taken place to determine the readiness for organisation-wide adoption and institutionalisation, plans for the roll0ut are developed.

#### Phase 5: Leveraging

McFeeley [8] states that “*the objective of the Leveraging phase is to make the next pass through the IDEAL model more effective*”. The leveraging phase completes the process improvement cycle, and is where benefit can be seen across the organisation. Lessons learned are documented to inform the next cycle, and data is analysed from the pilot projects. The process improvement strategy is then improved and updated ready for the next run. The business needs are also reviewed to determine if the needs have been met, or if any additional funding or resources are required. The sponsorship is then renewed, and revised goals are set in the next SPI cycle.

## 2.5 Development Lifecycles

There are several different software development lifecycles available. The selection of model has very high impact on the testing that is carried out. It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use [9]. It is important that these are evaluated and that a prudent choice is made for the lifecycle of the proposed project.

### 2.5.1 The Waterfall Model

The waterfall model, also referred to as a linear-sequential life cycle model is simple to understand and easy to implement. In this model, each phase must be completed fully before moving on to the next. It is best implemented in a project where there is a very small probability of requirements changing once development begins, as this is costly in the waterfall model. At the end of each phase, a review takes place to determine if the project is on the right track. [10]

Figure 5 - Waterfall Model [10]

#### Advantages

* The model is simple to understand and easy to implement.
* The simplicity of the model makes it easy to manage – each phase has specified deliverables meaning that progress is easy to track.
* The fact that the phases do not overlap means that it is easy to gauge the progress of the project.
* This model excels in projects where the requirements are locked down and well understood.

#### Disadvantages

* Once the project reaches the testing phase, it is extremely costly in terms of time and money to go back into development to change something that was not well thought out at the concept stage.
* Working software is not produced until late in the life cycle.
* The inability to make changes once development has passed introduces a level of financial risk.
* The model is not well suited to complex projects.
* The model is not well suited to projects with stakeholders that have a proclivity for changing requirements.

### 2.5.2 Iterative Model

A close up of a logo

Description generated with high confidenceThe iterative model does not attempt to start development with a full specification of requirements agreed. Rather, development begins with specifying and implementing a part of the software – which is then reviewed and further requirements added and addressed. This process is repeated until development is complete. [11]

Figure 6 - Iterative Model [22]

#### Advantages

* Working software is produced early
* A high level of involvement helps keep stakeholders engaged.
* This model encourages reliable end-user feedback
* Less time is spent on documenting, and more time on developing
* It is useful in projects where requirements may be prone to change, or are not clearly defined

#### Disadvantages

* There is a higher risk of costly system architecture flaws due to requirements engineering not being conducted up front, or not being conducted thoroughly
* There is a tendency for scope-creep due to the iterative nature
* Deadlines can easily begin to slip as the team moves through the iterations.

### 2.5.3 Spiral Model

The Spiral model is similar to the incremental model due to its iterative nature, but with more emphasis placed on risk analysis. The spiral model consists of four phases; planning, risk analysis, engineering and evaluation. These phases are cycled through in iterations (called spirals in this model). [12]

A close up of a logo

Description generated with high confidence

Figure 7 - Spiral Model [13]

#### Advantages

* High degree of risk analysis, meaning that the project is risk-adverse.
* Useful in large projects, or “mission critical” projects where unnecessary risk is to be avoided
* Strong approval and documentation control
* Software is produced early
* Requirements can be added after development has started

#### Disadvantages

* Can be costly to implement
* Requires a high level of project management by an expert manager
* Risk analysis generally requires a qualified expert
* Does not suit small or individual projects

### 2.5.4 Agile Model

An agile model is another type of incremental model, with a heavy emphasis teamwork and testing. There are several different implementations of agile frameworks, such as Extreme Programming, SCRUM and Lean. Agile is often used for time-critical applications. [14]

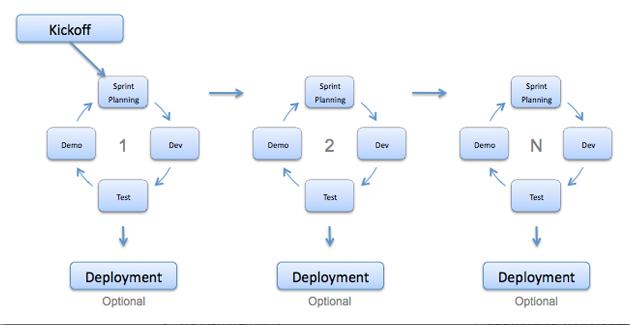


Figure 8 - Agile Model [14]

#### Advantages

* Customer satisfaction is achieved by rapid and continuous delivery of functional software.
* People and interactions are emphasised as opposed to processes and tools
* Working software is delivered early and often
* Close co-operation with the team of developers
* Works well in rapidly changing circumstances
* Changes in requirements are welcomed, even at a late stage.

#### Disadvantages

* Lack of emphasis on documentation
* The project can be very easily taken off track if the customer is not sure what they want
* Level of effort for deliverables is difficult to asses
* There is a risk that architectural flaws can creep through due to the absence of a formal requirements process
* Requires an expert project manager, and well-functioning team.

### 2.5.5 Choice of Development Life Cycle

After having conducted extensive research into the life cycles identified in this section, the life cycle of choice for the proposed project will be a modified **Waterfall Model**, where some overlap will be accommodated for in the phases. There are several reasons for the choice of model:

1. The fundamental reason is that this is an individual project, so there will be no development team or dedicated project manager. This means that it is important to choose a lifecycle that can adapt to the nuances of individual working and not require dedicated and extensive project management.
2. There will be limited involvement of the stakeholders. The nature of this project means that Citi will only be involved a small amount at the start in deciding the appropriate process areas of the CMMI model and then at the pilot stage – the project will benefit from the independence that the waterfall model emphasises.
3. Requirements will be very clearly understood. The unique nature of this individual style of project means that the requirements engineering and the development will be done by the same person, meaning that there is no chance of requirements being mis-understood or miscommunicated.
4. Given that this is an academic project, being undertaken on a part-time basis, scope-creep and time-slippage needs to be kept to an absolute minimum. The rigid nature of the waterfall method will help in this area.
5. The minimal engagement of the stakeholders means that the added benefit of early and frequent software delivery in the iterative models is nullified.

## 2.6 Literature Review Summary

This project is engineered to assist Citi in the diagnosing phase of the IDEAL cycle for implementing software process improvement. While Citi will use IDEAL to bring about an iteration of improvement, we are now providing a tool to help analyse the processes in Citi. In order to build this tool, we need to implement a development lifecycle, which is where the modified waterfall lifecycle will sit. The tool will use a novel implementation of RGT to elicit tacit knowledge about Citi’s enactment of the process areas within the CMMI.

# 3. Requirements Capture & Analysis

This chapter documents the requirements elicitation and analysis processes that were carried out for the project. In this chapter we will begin by discussing the various fact-finding techniques employed to generate requirements, then the needs of the business, followed by a listing of the requirements. Quality criteria will then be discussed with regards to these requirements. A Volere shell listing of the full set of requirements is available in appendix B.

## 3.1 Elicit Needs

The first step in any requirements development process is to elicit the needs of the various stakeholders involved in the project. The CMMI specific practice for this is described as follows; *“Eliciting goes beyond collecting requirements by proactively identifying additional requirements not explicitly provided by customers.”* [1]. This means that it is not sufficient to simply ask the customer to supply a list of their own requirements. There are various techniques available for the elicitation of requirements, such as; interviews, brainstorming and questionnaires. Questionnaires are an often-used method of developing requirements for software – using a questionnaire it is possible to gain the opinions of a wide range of potential users and stakeholders in a quantitative format, the idea being that this quantitative data would then inform the requirements that are developed. The unique challenge that Citi presents, however, is largely driven by its diverse and often siloed structure. For example; if a questionnaire was issued to a team in Citi Private Banking, they might cite requirements changes as their biggest stumbling block. While this may be true for one team, it is not representative of Citi overall – it is related to the specific customers in this area of the bank, which are known to be very demanding. Moreover, this team represents a very small proportion of headcount in Citi and as such this problem is limited to a small group of individuals and is not pervasive across the organisation. It is also difficult to expect a team of developers to be truly objective regarding their own processes, given their intimate involvement with them.

Given the difficulties regarding approaching the teams themselves to elicit needs, it was therefore important to gain an audience with stakeholders in Citi who were able to provide a comprehensive, holistic and unbiased view.

### 3.1.1 Citibank Internal Audit

The Internal Audit department in Citibank is primarily concerned with viewing the processes of the bank to ensure they meet any regulatory requirements and manage risk effectively. Belfast has an audit team that deals specifically with technology. The Technology Audit team conduct more than 200 audits annually across the global business. This team is responsible for reviewing the entire technology process of any given team, focusing on areas such as; software development lifecycles, software entitlements, code reviews, testing processes, requirements management and development, quality assurance, control environment, and risk mitigation. When an audit team raise an issue with an area of the business, it is stored in an issue-tracking system and tracked until its resolution. Given the diverse scope of this team, they would provide an excellent opportunity to gain an understanding for the key areas for improvement within Citi. It would also be beneficial to interrogate the database and understand how many issues were being raised by audit in particular process areas.

With the issues outlined above in relation to eliciting needs in mind, the primary techniques that have been used to elicit and develop requirements have been interviews and observation.

## 3.2 Observation: The Need for a CMMI Appraisal System

Citibank is a multinational company with more than 200,000 employees across the globe. With a company of this size, it becomes very difficult to understand how process improvement should be distributed across the organisation. It is often the case that, when a strategy is outlined at the board-level it becomes ineffective at a team-level because of a lack of specific knowledge of the processes of each team. Citi operate an employee survey annually; this survey illustrates almost identical issues each year. Relevant issues include that staff do not feel they have the appropriate tools to work effectively, and that staff feel that they are laboured by slow and redundant processes. Worse yet, these laboured and slow processes do not yield effective results, as software is often not fit for purpose once it is released. To draw on an example from my own experience within Citi, one of the software tools I use was developed by a team in Citi. This particular system has been live for 3 years and is still undergoing extensive improvement exercises to make it suit the purposes of the business. There have been countless examples regarding this system where key client deliverables have not been delivered due to bugs manifesting in the system meaning that work could not be completed. This is one of many examples in Citi of software that has not been fit for purpose. As such, the need for improvement is clear. However, the problem still remains that strategy issued from the board level is not sufficient at a team level. It is common in Citi for departments to have headcounts that measure in the thousands, rather than hundreds. Therefore, if a team-lead of a team of 30 developers wished to instantiate a process-improvement exercise, the answer will certainly not be found in strategy disseminated from board level. Team-leads then often try to improve processes using ad-hoc methods, often simply derived from their own experience of the team – this approach lacks the security and benefits that come with the use of a framework. However, this often ends up being the only approach available because there is no tool in Citi that a team-lead can use to analyse the processes of a specific project. The problem then scales up from team-leads, back to the heads of department. How can somebody at this level in the organisation really analyse performance without a systematic way of viewing process capability benchmarks across teams? At the moment, if a senior member of management wished to gain a holistic view of process capability across their department they would need to instruct each team lead to conduct a review of their own team, which would create a resources burden on the team and result in findings that were derived from inconsistent methods of appraisal.

With the problems outlined above several key needs of the business come to light:

1. Team-leads need a tool to understand process capability specific to their team
2. Team-leads need a tool to generate specific plans of improvement for their team
3. Heads of department need a tool to compare and benchmark results across projects, in order to understand where the greatest problems are so that further resources can be provided in this area

## 3.3 Interview: Citi Technology Audit

Attendees: Rosemary Teggart (Vice President, Technology Audit), Christopher Clokey (Vice President, Technology Audit)

This meeting was held to gain insight as to which process areas would be most beneficial to Citi if they were to be included in the CMMI tool. The areas of interest were as follows:

* **Configuration Management**
* It was noted that configuration management issues have been a recurring theme in technology audits across the bank
* Citi currently operate three separate configuration management systems, meaning that there is a large disconnect and inconsistency across teams in the bank
* **Requirements Management**
* Citi has recently put in effort to standardize requirements management globally onto one system. Therefore, it may be appealing to see if there has been a positive impact across the bank and if the standardization has been implemented effectively to remove outliers.
* **Requirements Development**
* Technology audit noted that there is often inadequate software produced in the bank due to incomplete requirements being set at the start
* Technology audit suspect that this is due to teams not often consulting all relevant stakeholders on each project to determine customer requirements
* **Process and Product Quality Assurance**
* It was noted that there is a high volume of audit issues raised on quality assurance of projects, specifically around plans to remediate software defects.

## 3.4 Interrogation of Internal Audit’s Issue Database

Citi Internal Audit holds a database documenting every issue that has been raised against the Citi businesses as a result of an audit. Each time an audit team raise an issue, it is recorded on the system along with a categorisation for the issue. Analysis was conducted on the issues raised against technology in the system and used the categories of the issues to determine where the greatest areas for concern were. Through this next sub-section, we will discuss the findings and how they influenced the choice of process area.

#### Finding A: Requirements Management and Configuration Management Account for more than a quarter of all technology issues raised in the system.

An exercise was conducted on on the database where classification was given to all issues raised that were relevant to their REQM or CM. It was found that 12% of all issues raised related to requirements management, and configuration management accounted for 15% of all issues. Given that there are 118 individual issue categories available in the system, and a total of 3,364 issues recorded, the 15% and 12% represent a significant population of the issues. PPQA, by contrast only accounted for 2% of all issues raised.

#### Finding B: Requirements Management issues are set to rise again

An exploration into the requirements management issues in the system revealed an interesting finding. Firstly, figure 9 demonstrates a large spike of issues raised in 2015. This was investigated with technology auditors and it was found that this was a result of several high-profile global audits exposing problems in systems that were operating globally, this resulted in an increased number of issues being found. The next notable feature of figure 9 is the immediate drop in issues raised in 2016 compared to 2017 – this correlates with the implementation of Citi’s Engineering Excellence programme, which was implemented globally to increase the quality of systems developed by the company. It is clear that this system has had an immediate positive impact on the issues being raised. However, the most salient and most concerning feature in figure 9 is the small upward trend found in 2017 compared to 2016. Rather than continuing on a downward trend, internal audit has started to find and raise more issues related to REQM. This could mean a number of things, but likely suggests that the implementation of the Engineering Excellence programme has only had a short-term effect on process quality, and unless a systematic change is implemented, the issues may continue to rise.

Figure 9 - REQM Issues Raised 2011-2017

#### Finding C: The implementation of a new Configuration Management system has had a positive effect on the number of CM-related issues being raised.

For the same reasons as mentioned in finding B, the number of CM-related issues explodes in 2015, as shown in figure 10 below. CM issues have been similarly improved by the implementation of the Engineering Excellence programme, however the key difference is that the downward trend continues into 2017, which suggests that long-term continuous improvement is likely. I consulted with Citi Internal Audit, and learned that a new Configuration Management system was implemented in 2016, which has contributed to the overall quality of CM process increasing across the organisation.

Figure 10 - CM Issues Raised 2011-2017

### 3.4.1 Choice of Process Areas to be Covered in CMMI Appraisal Tool

As a result of the interview with technology audit, the process areas to be included in the appraisal tool have been defined as follows:

1. Configuration Management
2. Requirements Management

## 3.5 Listing of Customer Requirements

The customer requirements in table 1 have been derived from interviews with the group of stakeholders from Citi outlined in the project plan.

Table 1 - Customer Requirements

|  |  |
| --- | --- |
| No. | Requirement |
| CR1 | The ability to conduct a CMMI appraisal against the chosen process areas for a given scope of projects. This scope of projects will likely represent an individual team within Citi Technology Belfast. |
| CR2 | A strengths and weaknesses report as a result of each CMMI appraisal |
| CR3 | The ability to compare appraisal results across teams to observe an overall picture of capability in Belfast. Ideally, Citi would like to be able to pin-point which particular teams have the weakest capability in the chosen process areas so that they can begin improving these areas. |
| CR4 | The ability to see gap analysis as a result of each appraisal in order to assess the extent to which a process area is enacted |
| CR5 | The ability to query the database of results for further analysis |
| CR6 | If possible within the time constraints, Citi would like a dashboard built into the system to allow for detailed analysis of results across the organization. |

## 3.6 Develop Product Requirements

Once the initial elicitation phase with the client was completed, the final set of functional (table 2) and non-functional requirements (table 3) were derived. The full listing of Volere shells for these requirements can be found in Appendix B.

Table 2 - Functional requirements

| No. | Requirement |
| --- | --- |
| FR1 | The system shall have a registration facility |
| FR2 | The system shall have a log-in facility |
| FR3 | The system administrator shall be able to assign layered permissions to users on the system. |
| FR4 | The system shall allow users to conduct the CMMI appraisal by filling in a questionnaire. |
| FR5 | The system shall produce a strengths and weaknesses document upon the completion of the questionnaire |
| FR6 | The system shall use a bespoke algorithmic scoring mechanism to establish the extent to which a process is enacted |
| FR7 | The system shall be capable of displaying a gap analysis style of chart once the appraisal is complete. |
| FR8 | The system shall include a back-end database capable of storing appraisal information |
| FR9 | The system shall include a visual dashboard feature |
| FR10 | The system shall be capable of displaying predesigned queries from the database as “key metrics”. |
| FR11 | The system shall feature a sand-box style “custom query builder” |
| FR12 | The system shall feature a sand-box style “custom graph builder” |
| FR13 | The system shall be web-hosted |
| FR14 | The system shall be mobile and tablet compatible |

Table 3 - Non-functional Requirements

|  |  |
| --- | --- |
| No. | Requirement |
| NF01 | The website’s design should be clean and uncluttered |
| NF02 | The website’s layout should remain consistent |
| NF03 | The website should be compatible with IE, Chrome, Safari and Firefox |
| NF04 | The website’s navigation should be easy to understand |
| NF05 | The website should confirm to HCI and usability principles |
| NF06 | The style of questioning for assessment will be based upon the repertory grid technique |

## 3.7 Requirements Analysis & Validation

Quality requirements form the foundation of a successful project. With poorly designed requirements, the system is far less likely to succeed. Firesmith states in the abstract of his quality requirements journal that “it is how well the quality requirements are engineered and implemented that tends to determine the success or failure of mission critical systems” [15]. Good requirements should remove any sense of ambiguity, be atomic, traceable, verifiable, consistent and tied to customer value.

In order to protect the system from poorly-formed requirements, a quality gateway has been introduced to ensure that each requirement fulfils the six key quality criteria. Suzanne and James Robertson describe the use of a quality gate as an *“organizational point through which all requirements must pass. The intention of having this gateway into the specification is to trap incorrect requirements as early as possible, thereby preventing incorrect requirements from passing into the design and implementation. To get through the quality gateway a requirement must satisfy a number of tests. The tests are devised to make sure that the requirement is an accurate statement of the business needs.”* [16]. As such, a set of six key quality criteria for requirements was developed to ensure the highest standard of requirements as possible. Implementing this gateway helps to reduce the risk of poor understanding of requirements, inconsistent design and investment in requirements that bring no value to the customer. Table 4 outlines a more detailed description for each requirement.

Table 4 - Requirements Quality Criteria

| Criteria | Review |
| --- | --- |
| Atomic | Each requirement contains only one traceable element, and avoids the usage of words such as “and” or “but”. The danger of having more than one traceable element per requirement means that the requirement may only be partially met, or met without a particular stipulation resulting in an ineffective system. |
| Unambiguous | Each requirement is stated in clear language which leaves no room for alternate interpretation. The meaning of a requirement may be clear to one stakeholder, but may mean something entirely different to another stakeholder. Kelly’s theory of personal constructs confirms this notion. [5] |
| Traceable | As the project moves forward, each requirement should be able to be traced continuously through the software development lifecycle. |
| Verifiable | Each requirement should have a suitable acceptance test outlined in the test plan once the software is complete. |
| Consistent | Each of the requirements are consistent with the overall aims of the system, and there are no clashes between requirements. |
| Tied to Customer Value | Each requirement features a rationale, which ultimately will provide value to the end user. This criterion prevents unnecessary work going into a requirement that will not provide benefit to the end user. This criterion is explored and validated further in the prioritisation process where we observe the relative weighting technique. |

Using the criteria developed in figure 12, each requirement was then passed through the quality gate and measured against the six key criteria. Ultimately, a requirement would not be accepted into the requirements specification until it was deemed to have met each quality criteria. Table 5 shows the results of each functional requirement against these criteria. There are no exceptions to any of the criteria in the requirements specification.

Table 5 - Requirements Quality Criteria Results

*Key: (A: Atomic, U: Unambiguous, T: Traceable, V: Verifiable, C: Consistent, TCV: Tied to Customer Value)*

| No. | Requirement | A | U | T | V | C | TCV |
| --- | --- | --- | --- | --- | --- | --- | --- |
| FR1 | The system shall have a registration facility | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR2 | The system shall have a log-in facility | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR3 | The system administrator shall be able to assign layered permissions to users on the system. | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR4 | The system shall allow users to conduct the CMMI appraisal by filling in a questionnaire. | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR5 | The system shall produce a strengths and weaknesses document upon the completion of the questionnaire | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR6 | The system shall use a bespoke algorithmic scoring mechanism to establish the extent to which a process is enacted | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR7 | The system shall be capable of displaying a gap analysis style of chart once the appraisal is complete. | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR8 | The system shall include a back-end database capable of storing appraisal information | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR9 | The system shall include a visual dashboard feature | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR10 | The system shall be capable of displaying predesigned queries from the database as “key metrics”. | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR11 | The system shall feature a sand-box style “custom query builder” | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR12 | The system shall feature a sand-box style “custom graph builder” | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR13 | The system shall be web-hosted | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| FR14 | The system shall be mobile and tablet compatible | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |

## 3.8 Acceptance Tests for Requirements

In order to ensure that requirements have been met once the product has been developed, the acceptance tests outlined in table 6 will be applied.

Table 6 - Acceptance tests for functional requirements

| No. | Requirement | Acceptance Test |
| --- | --- | --- |
| FR1 | The system shall have a registration facility | A user is able to register an account on the system. This registration facility must prevent users from registering the same account twice. |
| FR2 | The system shall have a log-in facility | A user is able to log in and out of the system using verified credentials gained from the registration process. Unauthenticated users should be denied access to the system |
| FR3 | The system administrator shall be able to assign layered permissions to users on the system. | The administrator (and only the administrator) is able to delegate access to view dashboard results to a registered user |
| FR4 | The system shall allow users to conduct the CMMI appraisal by filling in a questionnaire. | The user is able to conduct a full appraisal on the system |
| FR5 | The system shall produce a strengths and weaknesses document upon the completion of the questionnaire | The user is able to view a strengths and weaknesses document upon completing the questionnaire |
| FR6 | The system shall use a bespoke algorithmic scoring mechanism to establish the extent to which a process is enacted | An algorithmic score shall be displayed demonstrating the capability of a process once an appraisal is complete |
| FR7 | The system shall be capable of displaying a gap analysis style of chart once the appraisal is complete. | The user can view a gap analysis chart for each completed appraisal |
| FR8 | The system shall include a back-end database capable of storing appraisal information | The mySQL database can successfully store information entered from the front-end system |
| FR9 | The system shall include a visual dashboard feature | The user is able to view results of appraisals in a dashboard format |
| FR10 | The system shall be capable of displaying predesigned queries from the database as “key metrics”. | The user is able to view key metrics in the dashboard |
| FR11 | The system shall feature a sand-box style “custom query builder” | The user is able to execute a custom query from within the system |
| FR12 | The system shall feature a sand-box style “custom graph builder” | The user is able to design and view a custom graph from within the system |
| FR13 | The system shall be web-hosted | The user can access the system using a web browser |
| FR14 | The system shall be mobile and tablet compatible | The system can be used on smaller-screen devices |

## 3.9 Kano Requirements Analysis

Achieving customer satisfaction through requirements development is an essential phase of any project. The most salient aspect of achieving customer satisfaction is understanding the customer’s needs in the first place, and then prioritising requirements that will best meet these needs. Kano analysis is a method developed by Noriaki Kano that allows us to prioritise product requirements as a function of customer satisfaction – which is the ultimate end goal. [17]

Kano identified four general categories that a requirement can fall into; surprise and delight, more is better, must be, and better not be.

### 3.9.1 Surprise and Delight Requirements

These requirements are those that act as a key differentiator in a product from its competition. Surprise and delight requirements are often new innovations in technology, but this is not their sole aim, rather their aim is to add extra value to the product in a way that positively effects the end user. These requirements score highly on satisfaction, and in some cases may not require much investment. However, the most challenging aspect of a surprise and delight feature, is that with time it will eventually migrate to a basic expectation after existing in the system for a while.

### 3.9.2 More is Better Requirements

These requirements are generally concepts that are easy to grasp, for example bigger, faster, stronger. A good example of this type of requirement is response time in a system. If response time can be decreased from 5 seconds to 4 seconds, this has a positive influence on user satisfaction, and in theory, will continue to increase satisfaction the faster it gets. The key challenge with a more is better requirement is found in the theory of diminishing returns. Consider again the example of system speed: if a company spends £10,000 to improve the response time of the system to 1 second, this will have an enormous impact on satisfaction. However, it may cost a further £2o,000 to reduce that response time to 0.8 seconds, which ultimately will not produce the same increase in satisfaction, and as such would not be a worthy investment.

### 3.9.3 Must Be Requirements

These requirements are often the easiest to elicit from the customer, and generally describe the key functional aspects of a system. For example, the CMMI appraisal tool must be able to allow the user to conduct an appraisal. Without the must be requirements having been fulfilled, the system will never fulfil its purpose.

### 3.9.4 Better Not Be Requirements

Better not be requirements are generally the opposite to surprise and delight requirements. These are the features of the system that will cause the highest dissatisfaction if they were not included, but are generally not things that a customer would think to describe in their ideal system. For example, if a person was to book a hotel, they most likely would not ask if the hotel had hot water before booking – rather, the customer would simply expect this to be the case. Therefore, if the customer arrived at the hotel and found no hot water, they would be highly dissatisfied.

### 3.9.5 Leveraging Kano Analysis in the CMMI Appraisal Project

Using Kano’s four categories, each functional and non-functional requirement for the project has been classified to better inform the system development and identify which areas were worthy of the most investment. The requirements specification was discussed with a group of stakeholders at Citi to gain their input on what is most valuable to them. The results of the analysis are shown in figure 11. The results of the analysis confirm some useful findings:

1. The inclusion of sand-box style query and graph-building will greatly increase customer satisfaction; however this needs to be balanced with the risk it proposes to development in terms of time and impact on the speed of the system.
2. The dashboard feature will continue to increase customer satisfaction as more features and depth are added to it – adding extra features to the dashboard do not pose a large investment in terms of time.
3. Browser compatibility will be key, if users cannot access the tool on all browsers it will lead to frustration.
4. While the Kano model is highly useful in determining the desires of the customer, this needs to be translated into feasibility and balanced with risk, which can be done using Weigers’ weighting model.

Figure 11 - Results of Kano analysis

|  |  |
| --- | --- |
| **Surprise and Delight** | **More is Better** |
| FR06  FR07  FR11  FR12 | FR05  FR09  FR10 |
| **Must Be** | **Better Not Be** |
| NF01 NF02 NF05 FR01 FR02 FR03  FR04  FR13 | NF03 NF04  FR08  FR14 |

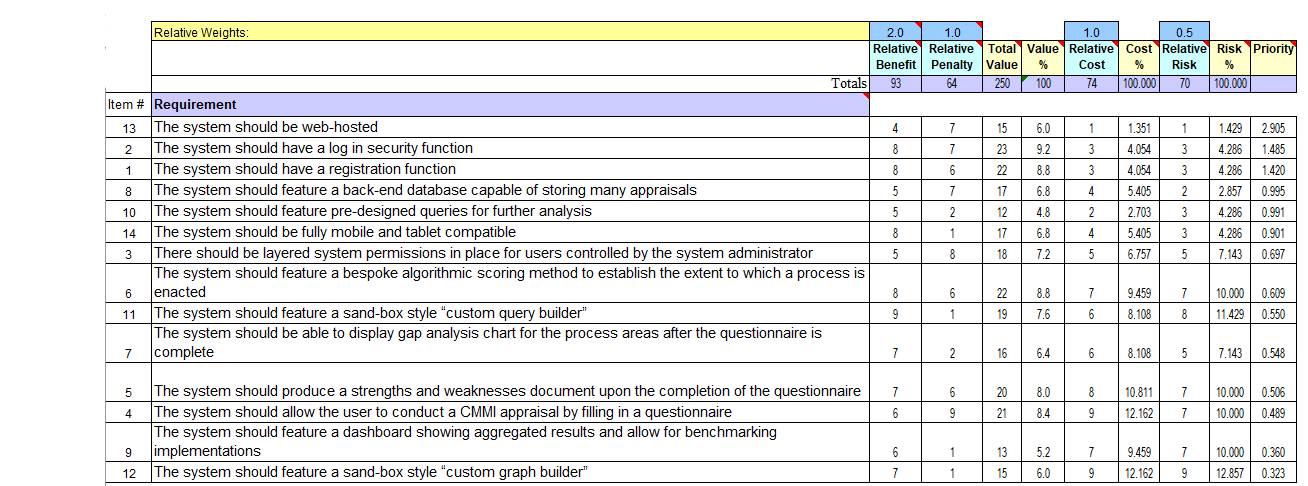
## 3.10 Requirements Weighting

With the strict time constraints in place to produce the system, it is important that requirements are prioritised with value to customer in mind. Weigers outlines the rationale for prioritising requirements as follows: *“When customer expectations are high, timelines are short, and resources are limited, you want to make sure the product contains the most essential functions.”* [18]

Weiger’s weighting technique takes four parameters into account; relative benefit, relative penalty, relative cost and relative risk. Relative benefit is the benefit that this particular requirement would provide to the end user, for example, being able to use the system on a mobile phone is a large benefit to the end user. Relative penalty describes the dissatisfaction that would be incurred if the requirement was not included in the system or was not implemented correctly, for example, if the user was not able to log into the system this would incur a high penalty because it would render the whole system inaccessible. Relative cost describes the cost that the requirement will incur on the development team, normally viewed in terms of time or money. Relative risk describes the risk that is waged on the overall system by implementing the requirement, for example, implementing the custom graph builder might cause rendering issues for the rest of the system due to the extensive code libraries involved in implementing the requirement. Weigers’ method takes these four parameters as an input and mathematically weights the requirements accordingly, resulting in a set of weighed requirements.

Weigers’ relative weighting technique has been employed on the functional requirements in figure 12. Unsurprisingly, the most basic functions of the system appear to be the highest priority. These requirements, while basic, are required for the system to function in the first place. While some requirements, such as the custom graph builder, may have a high for the customer, there is realistically no penalty if they are not implemented and the cost/risk scores are very high for the requirement. Whereas a login function may pose a low benefit to the user, but features a high penalty if the feature was not included and the cost/risk scores are very low.

Figure 12 - Relative Weighting of Requirements



# 4. Design

The following chapter provides information on the design of the system, and lays the foundation for the implementation of the system. This chapter is based upon a Model View Controller approach (MVC). This section will include interface design, use case diagrams and activity diagrams.

## 4.1 Model View Controller (MVC)

MVC is an architectural pattern that provides a way of documenting the design of a system. It is separated into three components: the model, the view and the controller. The model refers to the data-related logic that the user interacts with, for example, a customer object will retrieve customer information from the database, alter it, and then send it back to the database in an update. The view component refers to the user interface (UI) logic of the application. Controllers act as an interface between the model and the view, to process the business logic and incoming requests, manipulating data using the model component, and interacting with the views to render the output.

## 4.2 Architecture Design

Figure 13 below shows the high-level architecture design diagram for the system.

### 4.2.1 User Devices

The user will be able to access the web application from a web-browser across three main device types: mobile phones, tablets, and a computer. Given that this is a web application accessed through a browser, and not a native mobile application, there should be seamless accessibility between Apple, Android and Windows devices.

### 4.2.2 MySQL Database

The system shall store and retrieve data from a web-hosted MySQL database. This shall be storage and retrieval from this database shall be managed by PHP database-specific functions which will be deployed for the project.

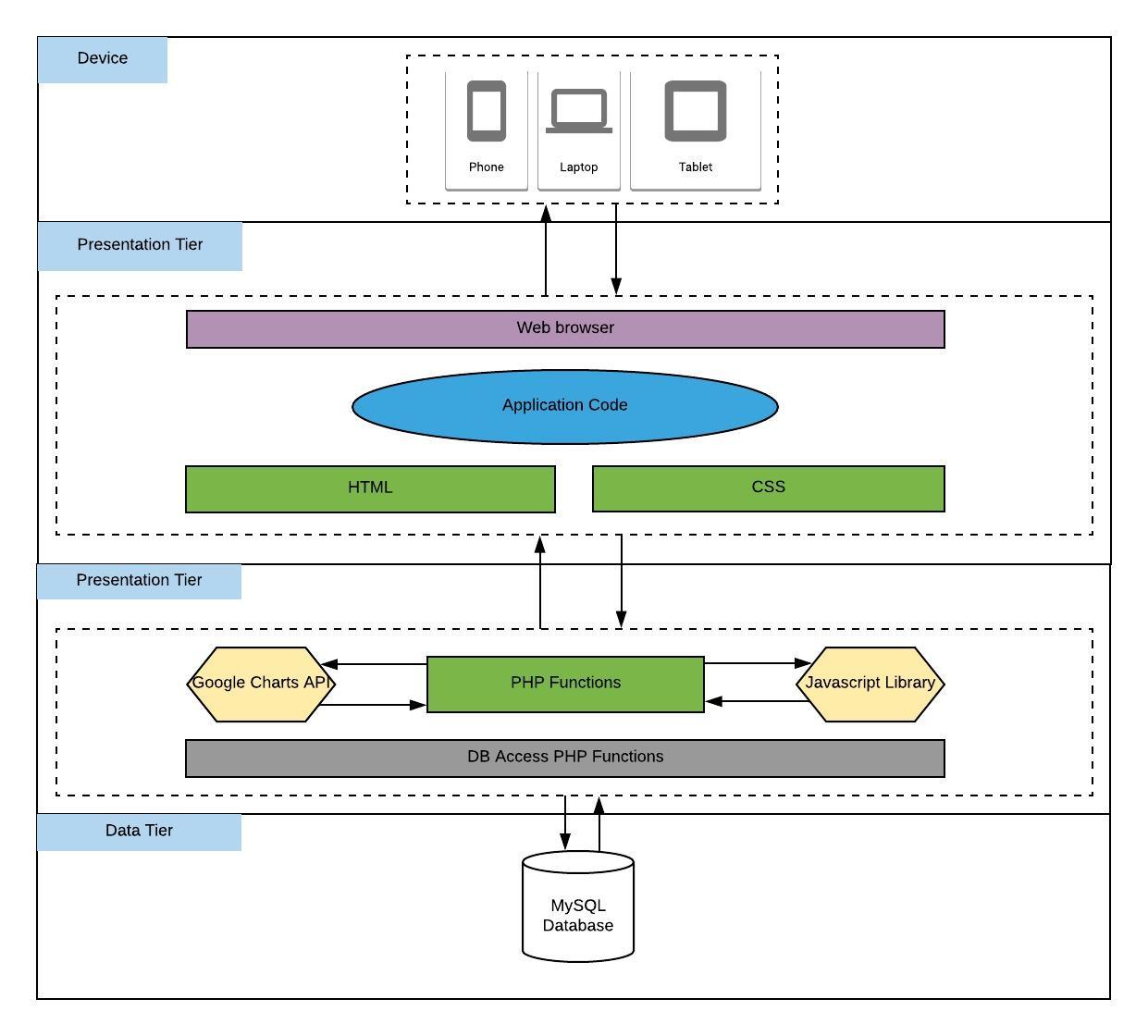
### 4.2.3 Google Charts API

The Google Charts API shall be implemented into the system in order to present the data visualisation features of the system. PHP will be used to retrieve data from the database in order to populate these visualisations.

### 4.2.4 HTML & CSS

HTML and CSS code will be used to render the structure and presentation of the web application on the user’s web browser. In particular, the bootstrap library will be employed for improved CSS features and mobile adaptivity.

Figure 13 - Architecture Diagram



## 4.3 Interface Design (View)

The interface design was carried out primarily with use case diagrams and wireframes. Key functional requirements have been illustrated in use case diagrams, while salient design aspects of the system have been depicted in the form of wireframes.

### 4.3.1 Use Case Model

The first step in the design process was to create a use case diagram showing the main functional aspects of the system. The diagram in figure 20 illustrates the functions that the user should expect to be able to perform in the system, and details how functions extend in certain circumstances. Using this diagram, it was then possible to derive the functionality of the system. The use case diagram is shown in appendix C.

### 4.3.2 Wireframe Designs

After the production of the use case model, wireframes were designed to illustrate the basic visual structure of the system. Wireframes are useful for the developer in providing a visual template to work towards in terms of UI design. Wireframes were created for the most salient UI features of the system, these include: the home page, the interface for entering appraisal information, appraisal results overview, results dashboard, and comparison dashboard. Wireframes for each of these UI interfaces are shown in appendix D.

## 4.4 Entity Relationship Design (Model)

Entity relationship mapping is an important part of designing a data storage and extraction facility in a system. The CMMI system presents a rather unique scenario, however, where having related entities in the database may not be of the most benefit.

Principally speaking, there are only two entities involved in the CMMI appraisal system: appraisal data, and user data. Inherently, these entities are not related: an appraisal does not belong, nor is associated, to a particular user or set of users. Conversely, a user does not belong, or is associated to, an appraisal. As such, users and appraisals shall not be linked in the data model but remain as two separate entities. However, when examining the appraisal entity, we can see one small area of redundancy, which would usually call for a normalisation to take place. This redundancy comes in the form of the “Line of business”, or L1 as it is referred to in Citi, associated to the appraisal. For example, an appraisal will be associated to exactly one (never more than one) line of business, such as ICG (Institutional Clients Group). A line of business, however, can be associated to more than one appraisal. In effect, in a normalised view, we have a one to many relationship, which would be reflected as follows in Figure 14.

Figure 14 - Normalised View

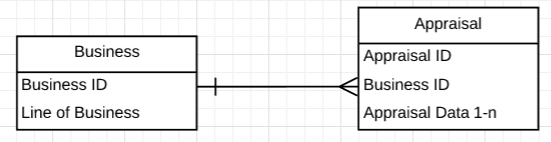
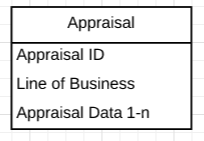


Figure 15 – zero-NF



Typically speaking, this is regarded as best practice when designing a database. However, in the lense of this particular project, this “best practice” would not be appropriate, and would be more memory intensive. In the normalised version shown in figure 20, there are two entities and five attributes. Whereas, in the un-normalised version of the ERD shown in figure 15, we see only one entity and 3 attributes. Both models will achieve the correct and desired result, but normalising the entity requires an extra entity to be created, and extra attributes. Further to this, any queries that are to be executed from the PHP script would have to consider joins in the SQL syntax if the line of business was required, this leads to more complex code, with little benefit in return.

In conclusion, if there were more than one redundant attribute present in the appraisal entity, then it would likely be worthwhile normalising the entities. However, given that there is only one redundant attribute in play, there is no benefit to be gained. Ultimately, normalising the tables would lead to much more work, with no benefit attached. As such, it was decided that the system would work from the data model in figure 21, using a “flat” approach to the ERD. The resultant design of the SQL database is shown in appendix F1, which shows the code used to construct the SQL databse.

## 4.5 System Behaviours (Controller)

The final part of the MVC approach is to map out controllers, which can also be referred to as behaviours. There are a variety of ways to do this, such as sequence diagrams, pseudo code and activity diagrams. For this project, activity diagrams have been completed to map out the behaviours of the system. There are four key behaviours that require a mapping: user registration, user log in, appraisal result storage and access to permissioned sections of the application. These four key areas have been presented in the form of activity diagrams in Appendix E.

# 5. Implementation

This chapter demonstrates the implementation of the chosen technologies and techniques in the project. The MVC model discussed in the previous chapter forms the foundation on which this implementation was based. In this chapter we will discuss: the implementation of the view by discussing how the structure and appearance of the website has been generated, the implementation of the model by discussing the creation and use of the MySQL database, and the implementation of the many controllers across the system by examining the code that has been used to implement the desired behaviours in the system.

## 5.1 Website Structure & Appearance

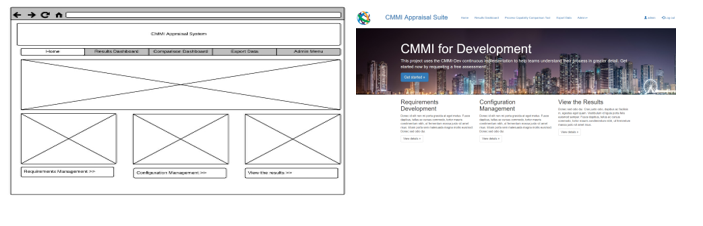
The home page of the website provides a good starting point to discuss the structure of the website. Figure 16 below demonstrates the wireframe for the homepage that was drawn up in the design chapter versus the actual result produced. The resultant web page adheres strictly to the design outlined in the wireframe. 

Figure 16 - wireframe for home page versus actual result

Functional requirement 14 states that the website needs to be able to adapt and display correctly on mobile devices, this means that the implementation needs to consider adaptive display for the website. This has primarily been achieved through the use of CSS, most notably through the use of the Bootstrap library.

### 5.1.1 Using Bootstrap to Display Page Content

Figure 17 below demonstrates the use of the bootstrap library classes to manage the content of the web page. Bootstrap operates a “grid” system, allowing the user to define the structure of the webpages in the form of rows and columns. If the page is resized (for example, if the user is viewing on a mobile device), then these columns will automatically drop underneath each other to create a mobile-friendly view.

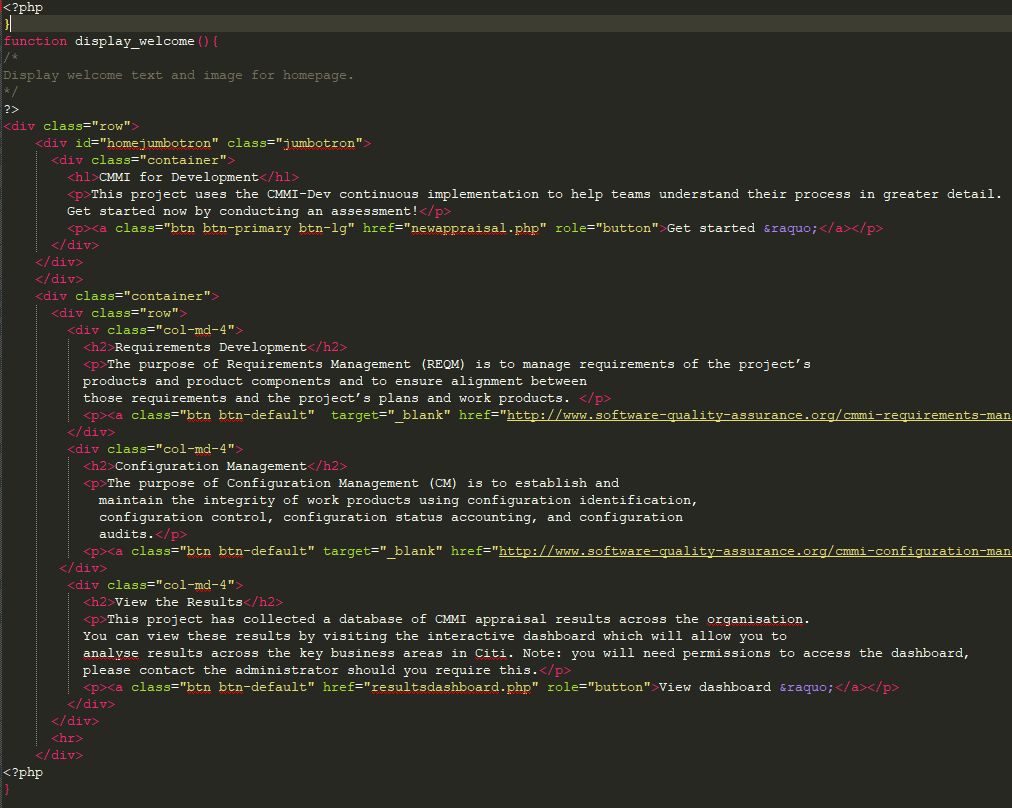


Figure 17 - Bootstrap grid system

The grid system has been used throughout the entire CMMI website in order to preserve and enhance the mobile adaptivity of the system.

### 5.1.2 Creating a Mobile-friendly Menu

One of the key challenges for a website developer is creating a website menu that behaves well on a mobile device. It is not possible to simply rely on the grid system that bootstrap offers to do this. If we relied solely on the grid system then the menu items would simply drop underneath each other when viewed on a mobile device, resulting in the screen being consumed entirely by the menu, and the user having to scroll past it each time to view the page content. The menu implemented in the system works well on both desktop and mobile devices. Figure 18 below demonstrates the menu while being displayed on a large-screen device, such as a desktop or laptop computer.

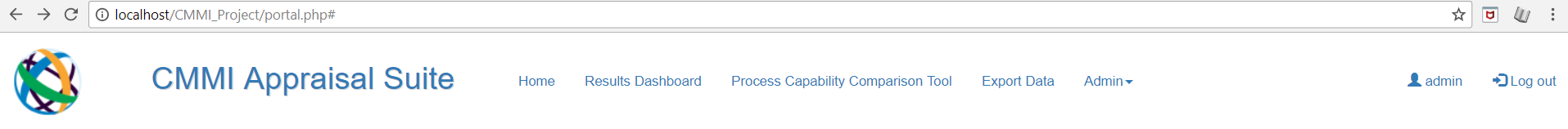


Figure 18 - Menu on desktop device

|  |  |
| --- | --- |
| Figures 19 and 20 below show the menu when displayed on a mobile device. Rather than the full menu being displayed, only the page title is displayed and the menu is replaced with a mobile menu button. The user can then click this button to expand out the menu, as shown in figure 20.  Figure 19 - Mobile menu | Figure 20 - Expanded mobile menu |

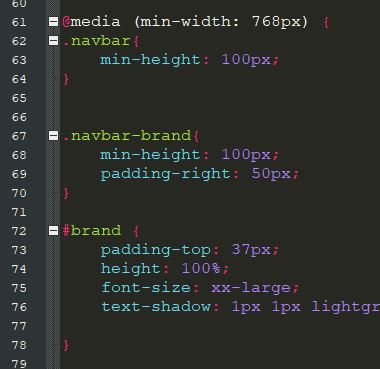
This behaviour has been achieved through using CSS to alter the menu based on a media query, shown in figure 21. The media query determines the width of the user’s device and then displays the mobile menu if the device width is less than 768 pixels, or the full menu if the width is greater than 768.

Figure 21 - CSS Media Query

## 5.2 Implementation of MySQL Database

The MySQL database that has been implemented is the single point of storage for all data held in the system. As described in the design chapter of this report, there are two tables that have been created in the database, a user table and an appraisal table. These two tables do not have a relationship with each other. The user table is responsible for storing information about the end-users of the system, such as names, passwords and permission levels. The appraisals table is responsible for holding all data that is to be collected relating to the two process areas. The appraisal table is the largest table of the two and holds a large amount of information, due to the implementation of the repertory grid to gather process information. Over all the specific practices in both process areas, there is a total of 252 unique pieces of information that need to be collected and stored in a unique location in the database to be available for further analysis. Due to this large amount of information, the appraisals table is very large, holding a total of 258 columns of data for each appraisal. The code used to create both the appraisals and users table can be found in appendix F.

### 5.2.1 Connecting to MySQL via PHP

In order to facilitate connection between the website and the MySQL database, I have written the db\_connect function. This function uses the creates a php msqli object, which accepts the server name, php log in information, and table names as parameters. This creates a connection object in PHP which connects the website to the database. This object is then used in any MySQL queries throughout the system.

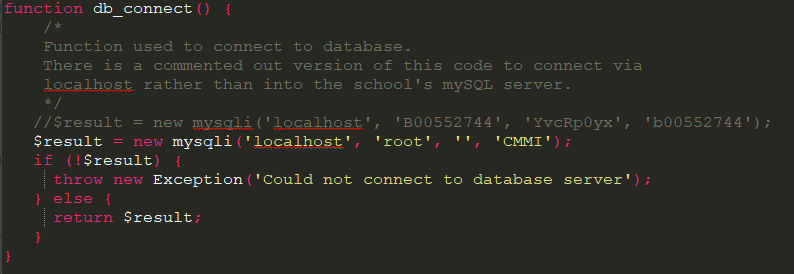


Figure 22 - Connect to MySQL via PHP

## 5.3 Authentication

Due to the potentially sensitive information being stored in the system, there is a need for authentication protocols to be in place, this ranges from requiring permissions to access sensitive web pages, log in facilities and password hashing.

### 5.3.1 Registration Process

Certain controls and restraints needed to be placed upon the registration facility in order to protect the integrity of the data. One of the most important controls here is checking that the username does not already exist in the system. If two users shared the same username, it could create problems when the administrator assigns permissions, resulting in users having access to areas of the web application that they should not.

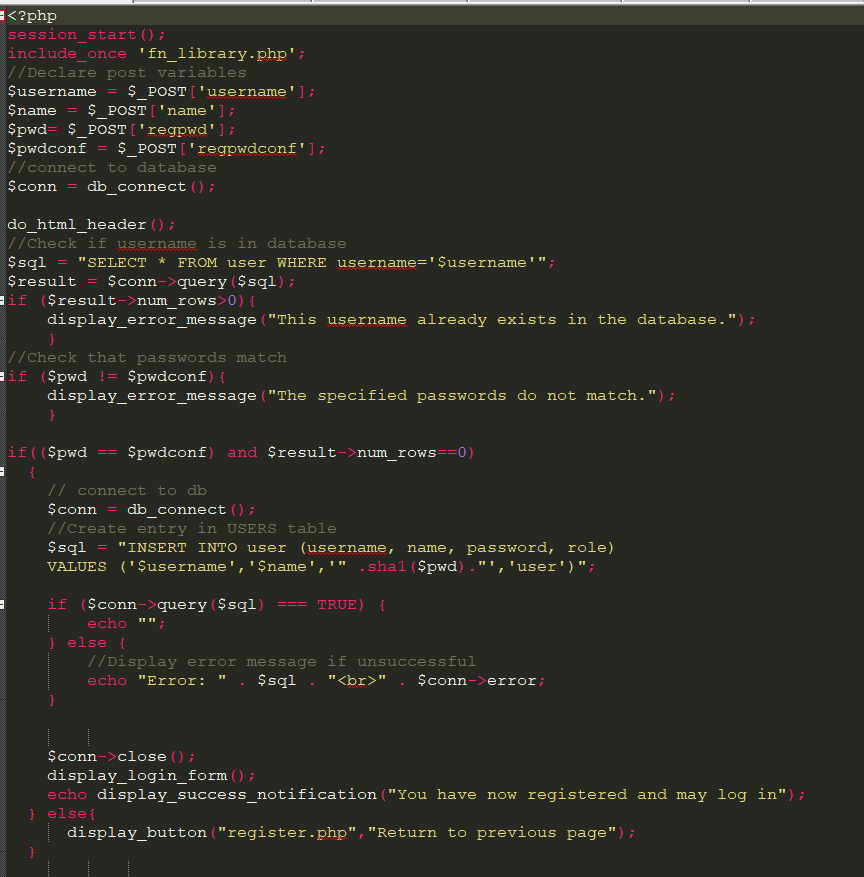


Figure 23 - User registration process

Figure 23 illustrates the PHP code that is used to process a registration request. The user entered information is collected from the $\_POST variables submitted by the form and then ran through a series of checks. A SQL query is ran to check if there are any other users in the database that have the same username, and a check is ran to ensure that the passwords match. If either of these tests fail, then the user will be shown an error message and the registration will not take place. If the data passes both tests, then the new user will be inserted into the user table in the database, and the user can now log in with that account. Note that the password is stored into the database after being hashed by the “Sha1” method.

### 5.3.2 Log-in Process

When the user clicks the log-in button, they are directed to the home page, where a section of code runs to check if they have arrived at the page as a result of a log-in process.

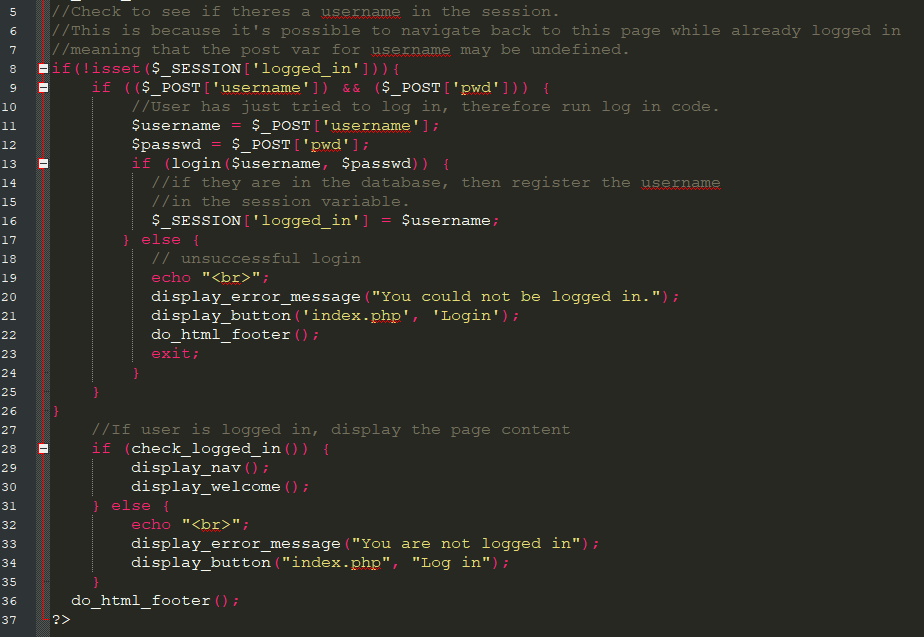


Figure 24 - Log in process on home page

Figure 24 shows the code that runs on the home page to check if a user has just been logged in. Line 8 of the code checks if the session variable logged\_in has been set – this check is designed to find out if the user has already been logged in, and if they have not, then the log-in code is ran below. The post variables from the log-in form are checked, and the user entered data is taken from them and stored in two username and password variables. These two variables are then ran through the login function, which is shown below in figure 25.

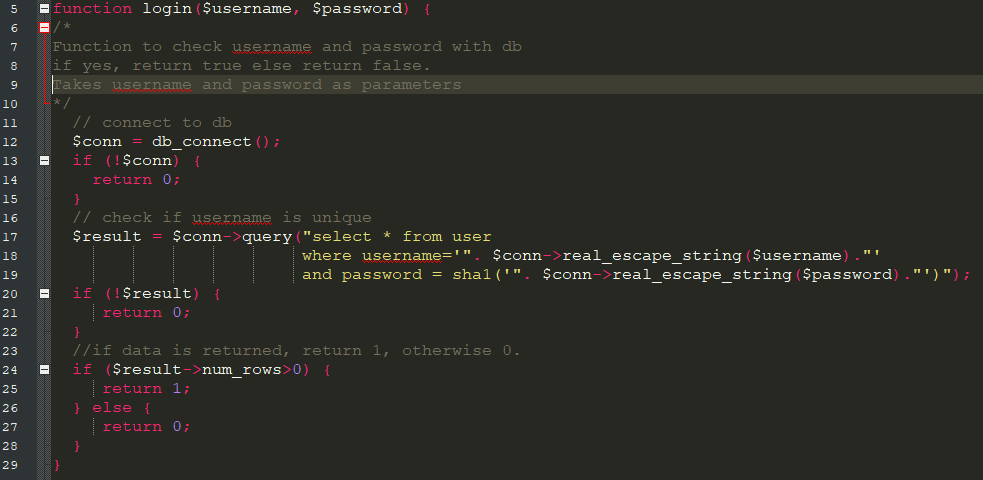


Figure 25 - login function

The function in figure 25 accepts the username and password as parameters and checks to see if a row in the user table matches the entered results. If a match is found then we know the log in has been successful, and the function returns a 1, and if no match is found then the user/password combination does not exist and the user is not logged in. The result of this function is used in figure 24 to determine if the login has been successful on line 13 of the code. If the login has been successful, then the session variable logged\_in is set with the username, which is used throughout the rest of the web application to identify the currently logged in user.

#### Preventing Unauthorised Access to Pages

A successful login process is only one part of the authentication paradigm. The next step is to ensure that unauthorised users do not have access to protected pages of the website. This is achieved through the use of the check\_logged\_in function shown in Figure 26.

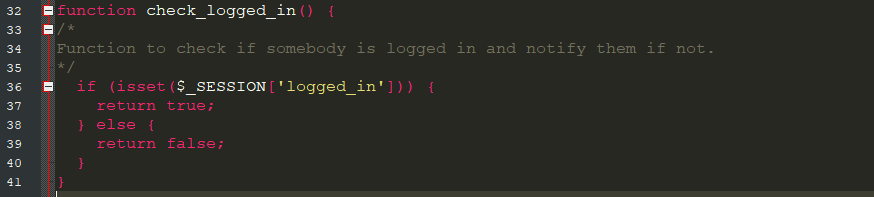


Figure 26 - check\_logged\_in function

This function simply checks to see if the session variable logged\_in has been set. We know that the only way that this session variable can be set is as a result of a successful login process. If the variable is set, a true is returned, and a false if it is not set. This simple function can be used in the form of an if statement to prevent users from seeing a page when they are not logged in. Figure 27 demonstrates the use of this function to only show the portal.php web page to users who are logged in.

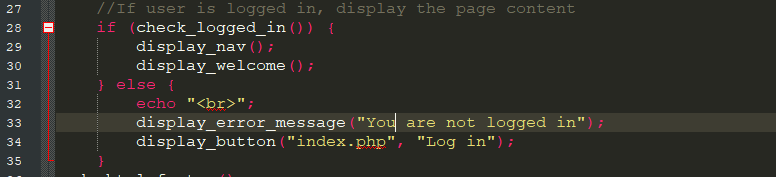


Figure 27 - checked\_logged\_in in use

### 5.3.3 Use of Permissions to Protect Sensitive Webpages

Some of the data held in the website is not suited to network-wide access, so it was necessary to implement a permission based solution to ensure that only select users have access to the pages that involve specific data about a project, or a set of projects. Only the system administrator, or a user who has been granted administrator permissions, is able to assign permissions to other users. There are 3 levels of permissions in effect in the system:

1. User – this is the most basic account, and only has access to the homepage
2. Results access – this permission level allows the user to access the comparison and results dashboard
3. Administrator – this permission level allows the user to conduct and manage appraisals, alter user passwords and assign permissions to other users on the system.

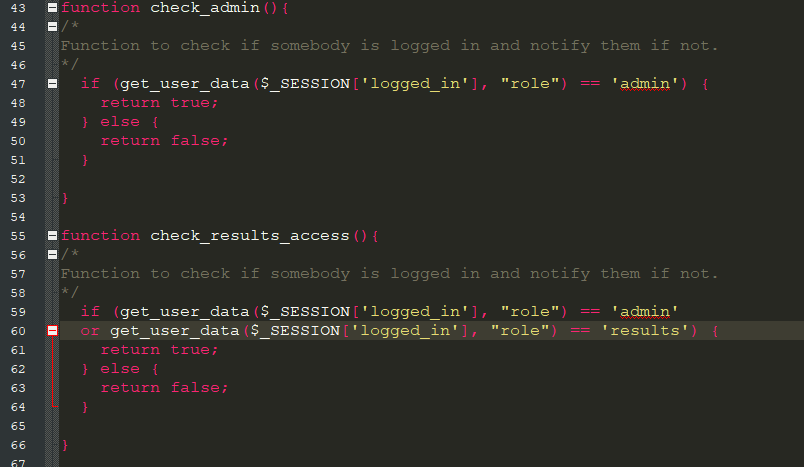


Figure 28 - Functions to check permission levels

Figure 28 shows the two functions that are used throughout the system to check the permission level of users. The first function checks to see if a user has administrator access, while the second function checks that the user has at least results access.

#### Administrator Menu & Assigning Permissions

If the user is logged in as an administrator, a special administrator-only menu will be shown in the navigation bar. This admin only menu is shown in figure 29. It has three administrator-only options: begin a new appraisal, manage appraisals and manage users.

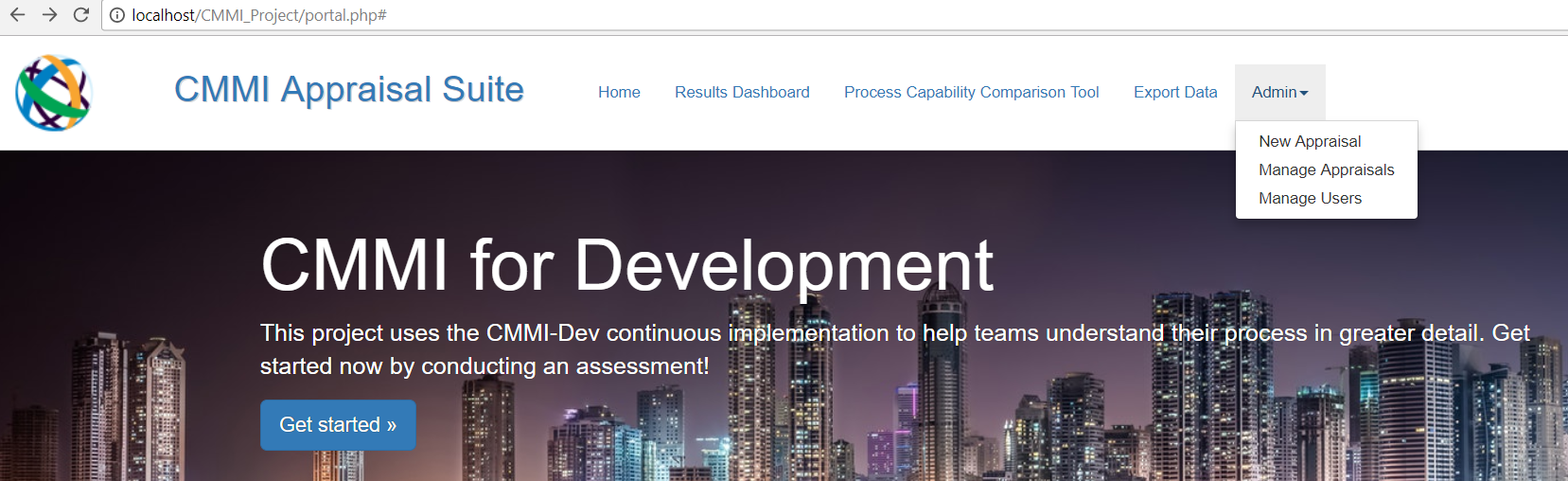


Figure 29 - Administrator menu

In the manage users section (figure 30), the administrator is able to search through the list of users and edit their password and permission level. The search bar is controlled by JavaScript and automatically filters the list of users while a search term is being typed. The manage appraisals section offers similar functionality and the JavaScript for this will be explained later in the chapter.

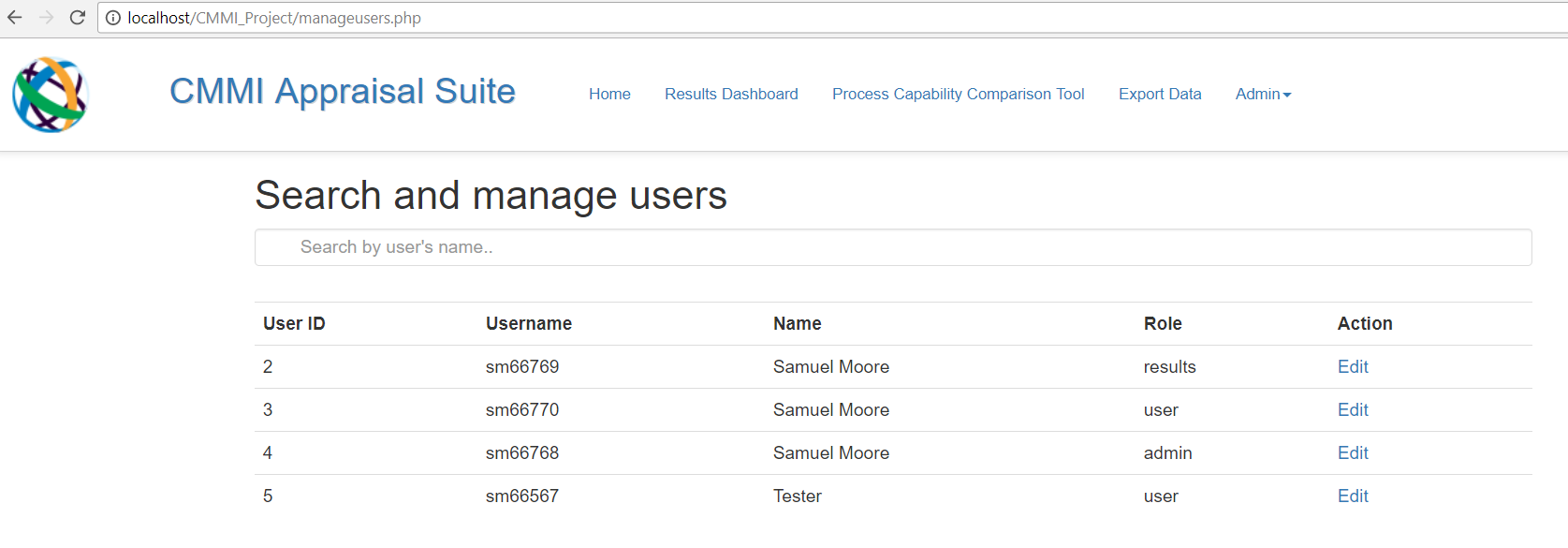


Figure 30 - Manage Users

## 5.4 Appraisal Interface

The appraisal interface is the most utilised section of the application. In the process of conducting an appraisal, the assessor would need to assess more than 30 different process areas, meaning that the appraisal interface will be interacted with more than 30 times per appraisal. As such, it was vital to ensure that the ergonomics of this facility were excellent. After all, even a small nuisance in filling out a form can become an enormous bother when you have to fill that form out 30 times. Figures 30 and 32 demonstrate the layout of this form on both a desktop and mobile device.

|  |  |
| --- | --- |
| Figure 31 - Appraisal interface, desktop | Figure 32 - Appraisal interface, mobile |

The repertory grid questions are answered in the form of a drop-down menu, which is a useful validation technique that prevents the user from entering invalid values. Radio buttons provide a similar validation property; however, these do not scale well to mobile devices. Therefore, to solve this scalability issue, drop-down boxes were introduced between each of the constructs, then the Bootstrap grid functionality aligns the items correctly when being displayed on a mobile device. The strength and weakness text boxes are free text entries that can be resized by the user when entering text, should they wish to have a wider view.

### 5.4.1 Appraisal Interface Sub-menu

Given the large number of process areas to be assessed, there was a need to implement a sub-menu that allowed the user to easily navigate between process areas. This sub-menu is shown in figure 33.

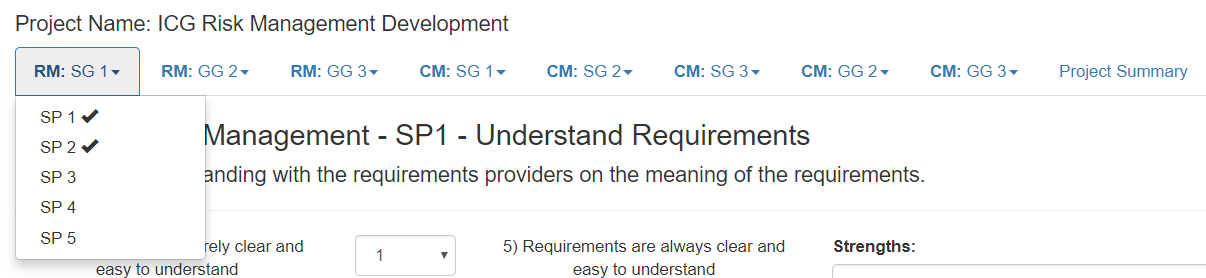


Figure 33 - Appraisal sub-menu

The sub-menu is organised by a series of drop down menus, which collate the process areas together by specific and generic goals, according to the process area. There are two important aspects of this menu to discuss; how the system reacts to a menu choice, and how the system displays a tick beside each menu item once the practice has been assessed.

#### Handling Menu Choices

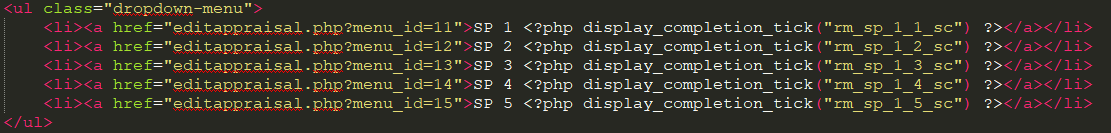


Figure 34 - Parameters built into destination URL

Figure 34 shows a snippet of the code for 5 of the links in the sub-menu. In each URL that corresponds to a menu choice there is a unique menu\_id parameter associated to that menu choice. When the user selects this hyper link, the web-browser will navigate to the editappraisal.php page while holiding a URL parameter. Figure 35 demonstrates this in action, where the menu\_id is set to 11.



Figure 35 - Menu parameter in URL

|  |  |
| --- | --- |
| This menu parameter is then handled in editappraisal.php through the use of a switch statement, using the URL parameter as the input (shown in figure 36). The benefit of using this particular method of showing each process area form is that rather than building 30 individual web pages, there is only one. Instead, there are 30 different versions of the form that are unique to each specific practice which are dyanimically displayed according to the value held in the URL parameter. | Figure 36 - Case statement for URL parameter |

#### Handling Completed Menu Items

Given the large amount of process areas, it was necessary for the user to be able to see which process areas had already been assessed at a quick glance. In figure 34 we see a PHP call to the function display\_completion\_tick. This function is shown in figure 37.

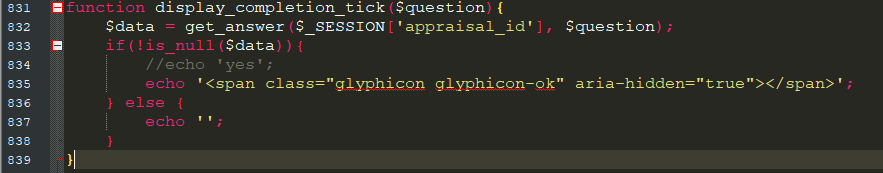


Figure 37 - display\_completion\_tick function

This function accepts the question label as a parameter (note: this question label corresponds to the variable name held in MySQL). The function then calls another function, get\_answer, this time with two parameters: the session ID (which corresponds to the appraisal currently being executed) and the question label, which was passed in as a parameter to the display\_completion\_tick function.

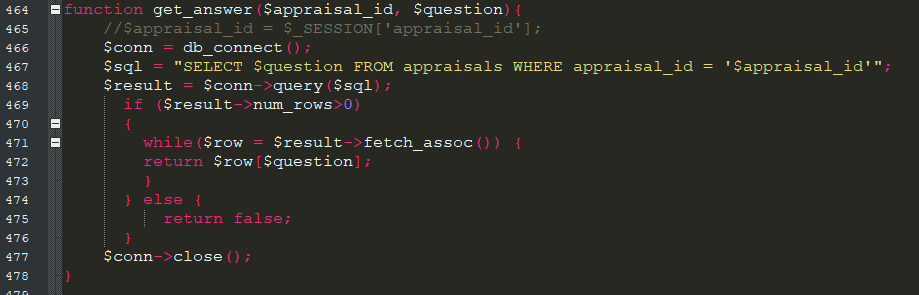


Figure 38 - get\_answer function

The get\_answer function takes the two parameters and uses them to identify a particular response in the SQL database. The SQL query uses both parameters in the WHERE clause, which narrows the query results down to a single attribute, which is then returned. This returned value is then received in the display\_completion\_tick function, where an if statement is used to check if there is any data in the return. If there is data in the return, then we know that responses have been entered for that section. If there is no data, then no responses have been entered and the section is not complete. If the section is complete, an HTML glyphicon tick is displayed beside the menu item, as shown in figure 35.

### 5.4.2 Storing Appraisal Data

Each time the user clicks the save button after entering responses on an edit appraisal form, a store procedure is executed in the system to extract the results from the form and place them into their relevant locations in the database. This process begins by extracting the answers from the $\_POST variables populated when the form is submitted, and then running the data from these variables through a store\_answer function.

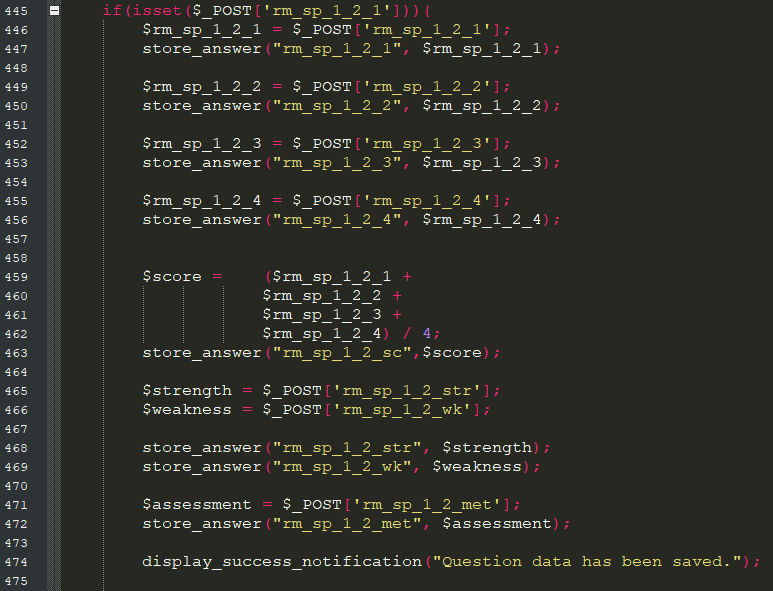


Figure 39 - Code to store answers from a submitted appraisal form

Figure 39 shows an example of the code for a specific practice that is ran when the user clicks the save button. This code actually executes each time the form is loaded, so it was necessary to place an if statement around it to instruct the code not to execute if the form has not been filled in. This if statement can be seen on line 445 – it checks the $\_POST variable for one of the repertory grid responses. If the $\_POST variable is set, then we know the user has submitted the form. If it is not set, then we know the user has not submitted the form, and therefore we do not run any of the storage code.

When lines 446-474 are ran, the system works through each of the form entries and runs them through a function called store\_answer, which we will discuss in further detail below. Note that the variable $score is actually a calculated variable which holds the average of the 4 repertory grid responses for this specific practice. This is how the specific practice score is generated and stored, which can then be used for analysis in the dashboard. Once all of the storage procedures have been executed, a success message is displayed to let the user know that their data has been saved.

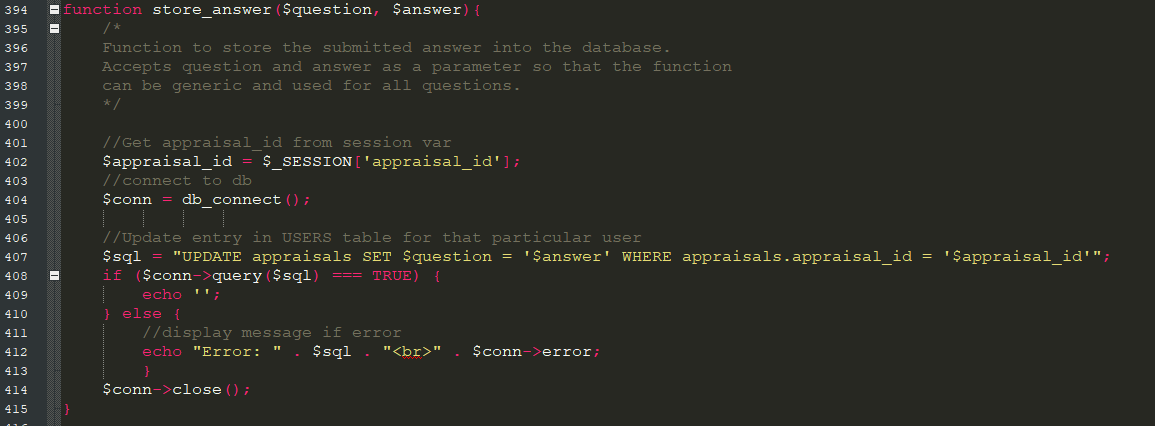


Figure 40 - store\_answer function

Figure 40 shows the store\_answer function. This is a generic function that has been developed so that it can store any data into any variable in the database. It was developed this way so that it can be used repeatedly so that there is no need to write repeating or redundant code. The function accepts two parameters. The first parameter is the question label, which will correspond to an attribute name in the MySQL database. The second parameter is the answer data that is to be stored into the attribute specified in the database. The function then requests the $\_SESSION variable appraisal\_id, which specifies which row of the table that the data should be stored in. We then execute an UPDATE statement in the MySQL database, which updates the attribute specified in the $question parameter with the data specified in the $data parameter, where the appraisal\_id in the SQL table is equal to the appraisal\_id in the $\_SESSION variable.

### 5.4.3 Populating the Form with Saved Responses

One of the challenges presented with the appraisal interface is that when a user enters data into the form, saves it and ends the session, then returns to the form at a later date, we need to be able to present the data in the form that is currently saved in the database. While this does not affect the data stored in the database, or impact the functionality of storing responses, it is an important ergonomic concern for the user. If a user navigates to a form wishing to simply change some typing errors in a strength entry or change some slight details in any of the responses, we need to be able to present the data that is currently stored. Referring to the point made earlier in this chapter, a small annoyance in an interface has the potential to become a great hardship if it is repeated 30 times to the user. Aside from this ergonomic concern, the user may think that there is a system error if they enter responses and return to the form later only to find it blank. There was significant challenge involved in populating the form with the currently-held data, it required a unique blend of PHP and HTML within the form to get the desired result. There were two different form entry types that needed to be handled, drop-down responses and free-text responses. These two differing form options required different handling procedures, which are discussed further below.

#### Handling Saved Drop-down Responses



Figure 41 - HTML & PHP to display repertory grid option

Figure 41 shows an example of the code used to present a repertory grid option in the appraisal form. This HTML structure presents one construct-pole on the left, a drop-down option in the middle, and the other construct-pole on the right. Note, that there is a call to a PHP function called display\_options – it is this function that is actually displaying the repertory grid construct. This generic function was developed because there are more than 200 repertory grid constructs that need to be handled and presented in the system, so having a generic function reduces code redundancy. In order to understand how the saved responses are presented, we need to examine this function further.

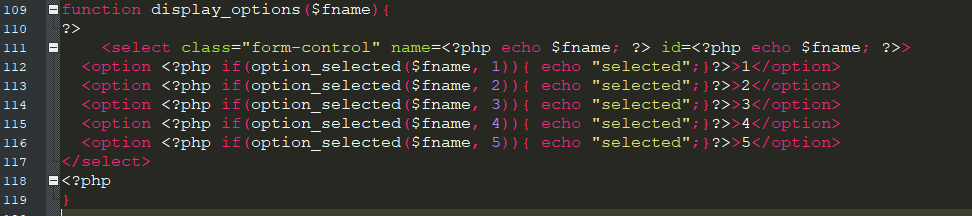


Figure 42 - display\_options function

The PHP function for display\_options is shown in figure 42. This function accepts the $fname as a parameter, which will correspond to the name of the attribute in the database that we wish to save the repertory grid response to – this parameter is used in the id of the select option in the form, which is then used in the store procedure to save into the database. The 5 options in the repertory grid are coded as options in the drop-down menu. Note that within each option there is a call to the PHP function option\_selected and an if statement that uses the result of this function to determine which of the 5 repertory grid responses was last selected and saved by the user.

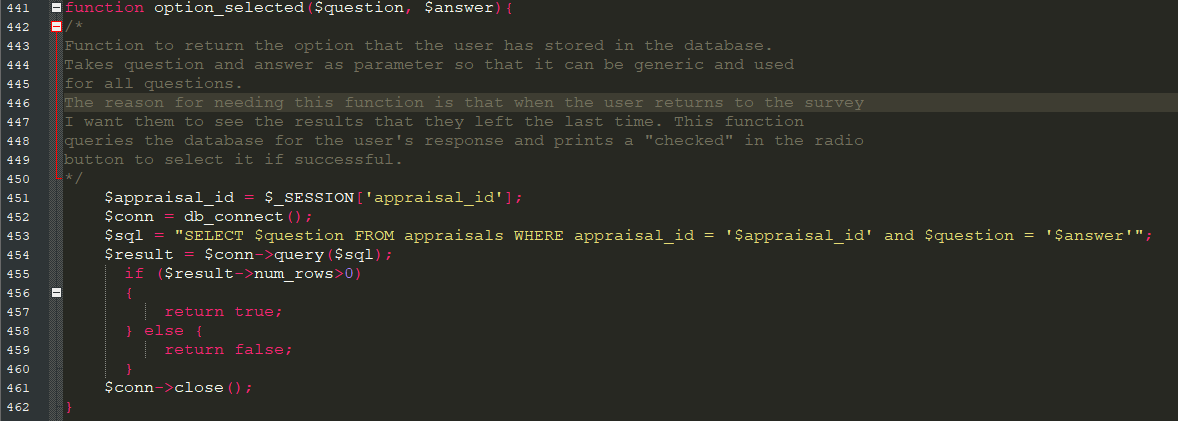


Figure 43 - option\_selected function

Figure 43 shows the option\_selected function, which takes the attribute name specified in the display\_options function to determine the current response to that attribute, which will be between 1 and 5. Option\_selected requires two parameters, the attribute name, and a number between 1 and 5, which will be used to determine if the repertory grid response option matches the response stored in the system. If a match is found, then the function returns a true and the drop-down box is updated to show that number.

#### Handling Saved Free-text Responses



Figure 44 - Code to display strength & weakness boxes

The line of code shown in figure 44 is used to display the strength and weakness text boxes for any given entry. Again, this has been programmed in such a way to reduce redundancy and repeating code. The boxes shown will depend upon the parameters entered in the display\_strwk\_boxes function – these parameters must correspond to the relevant strength and weakness attributes in the database.

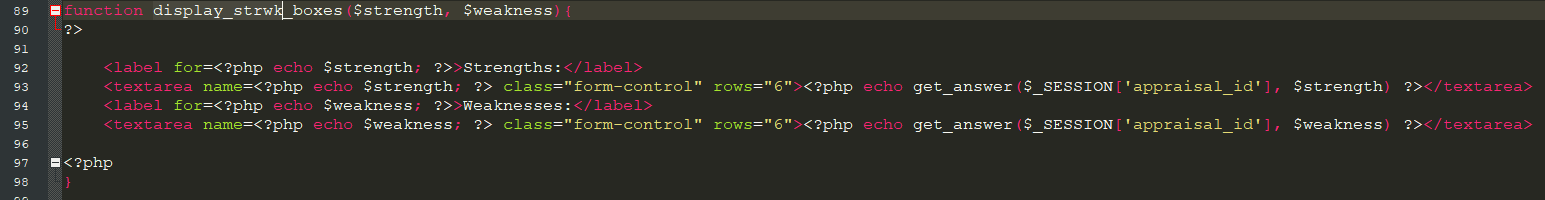


Figure 45 - display\_strwk\_boxes function

This function is shown in figure 45. It uses the parameters entered to define two text boxes with IDs corresponding to those entered in the parameters, which are then used later in the data storage process. A default text value is entered in the text box, which is gathered using the get\_answer function, which was described previously in figure 38.

### 5.4.4 Displaying a Dynamic Practice Score

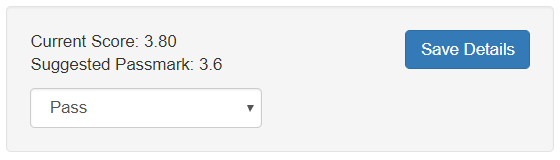
When the user is using the appraisal form to enter scores for a specific or generic practice, a live version of the current score is shown at the bottom right of the form, see figure 46 for an example of this. This allows the user to see the current score as a result of their scoring for each of the constructs in the repertory grid structure, and allows them to see what score will be entered into the database for that specific or generic practice. This functionality is provided through the use of JavaScript, which is shown and described further in figure 47.

Figure 46 - Live score for specific practice

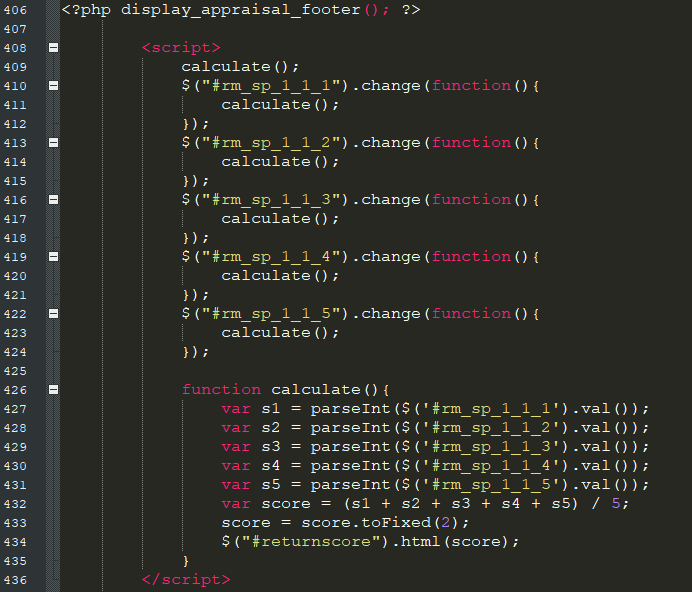


Figure 47 - JavaScript to update live score

In figure 47 we can see the JavaScript being used to control the live score for the specific practice. Event listeners are placed on each of the variables being used in the repertory grid construct. When an event is triggered, the calculate function is called, shown in line 426 in figure 47. This function calculates the new score based on the values in the repertory grid and returns it as a numeric value to the HTML, where it is then displayed on the webpage.

### 5.4.5 Appraisal Results Overview

For every appraisal conducted in the system, a results overview is generated which shows the capability level for both process areas and a table is generated compiling the strengths and weaknesses entered by the assessor for every specific practice. An example of this page is found in figures 48 and 49.

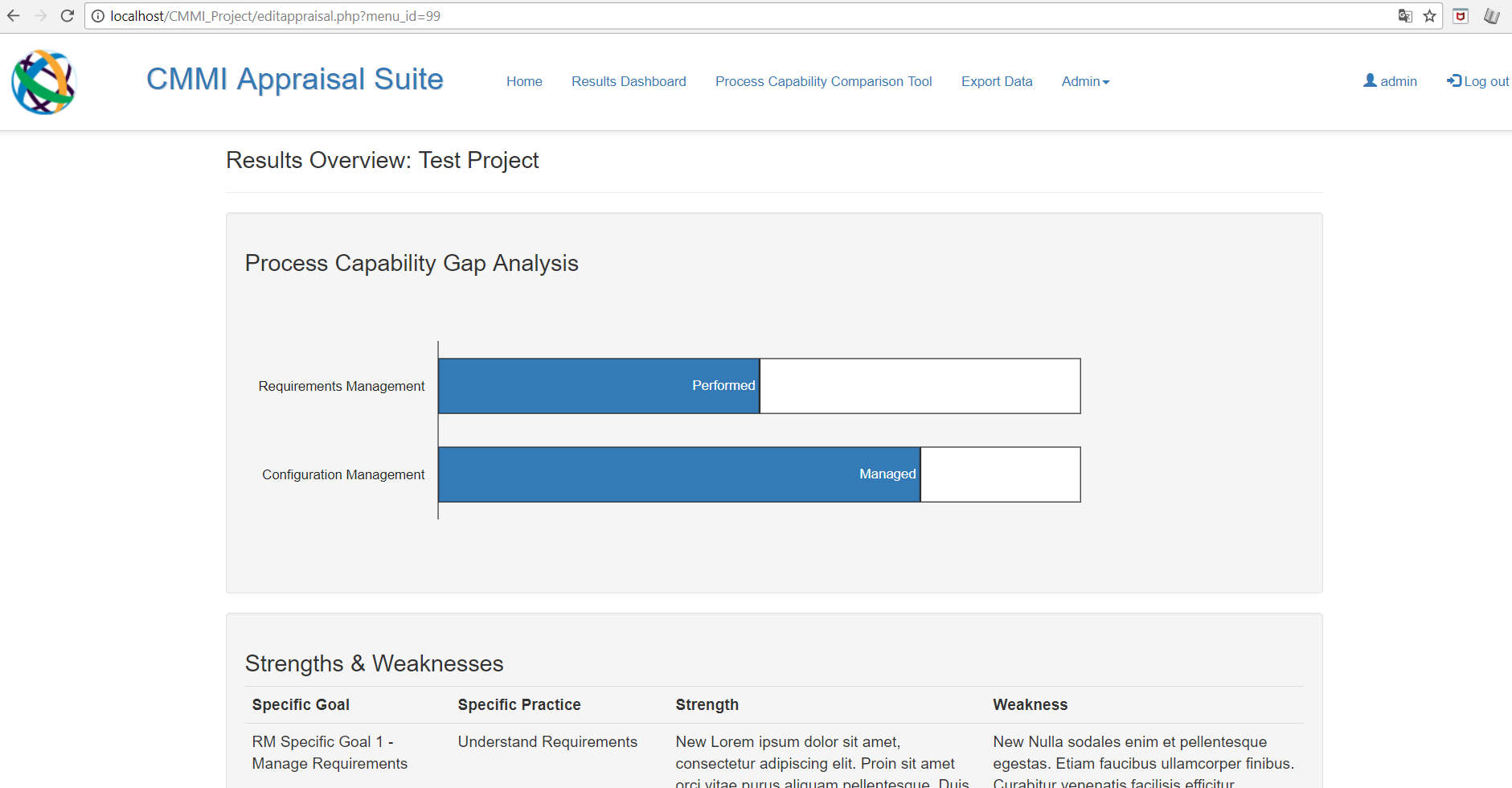


Figure 48 - Results overview, capability chart

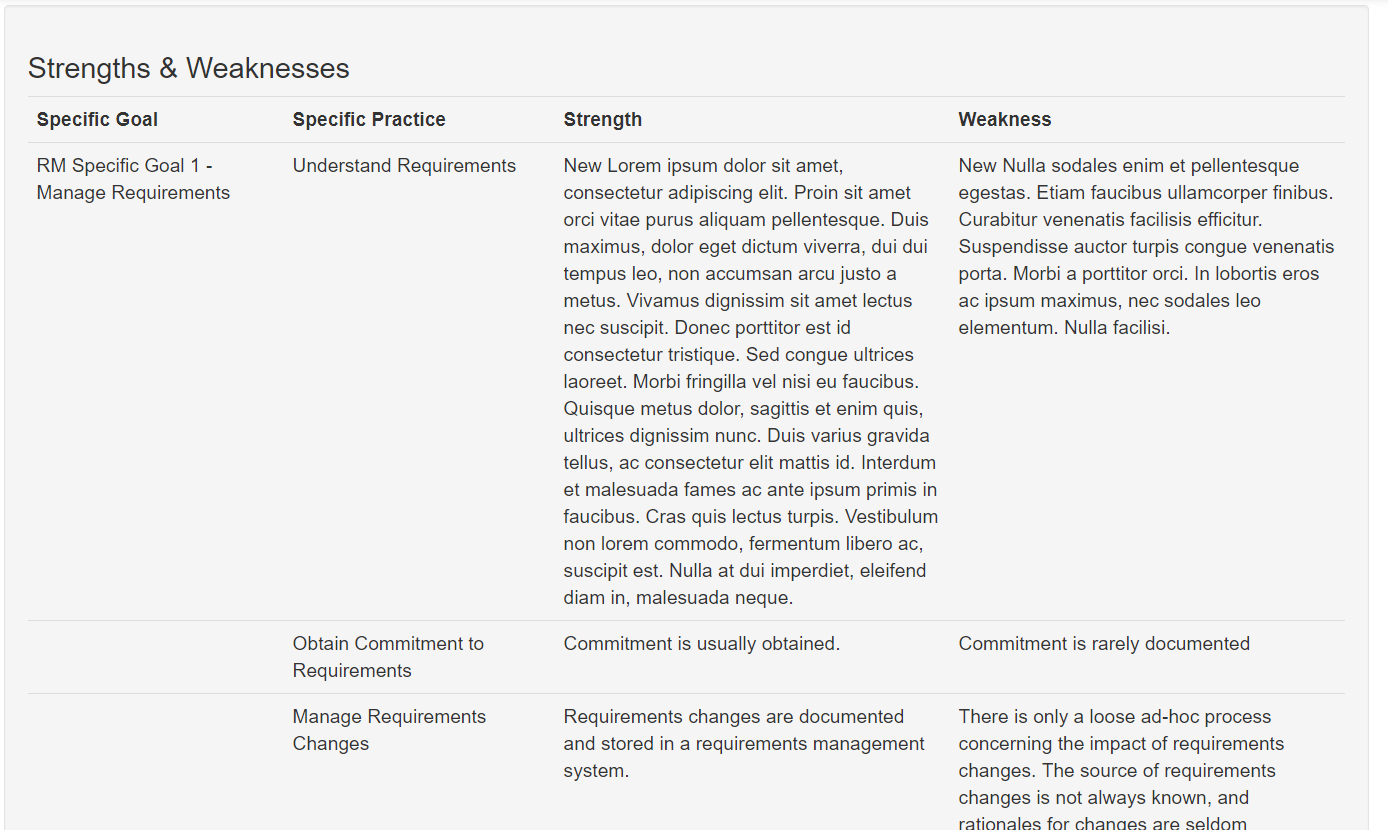


Figure 49 - Results overview, strengths and weakness table preview.

The gap analysis chart is generated using the Google Charts API, which will be discussed later in this chapter when we focus specifically on this API. The strengths and weaknesses table is generated by using HTML and PHP to extract the relevant data from the database and presenting it in a table format on the page.

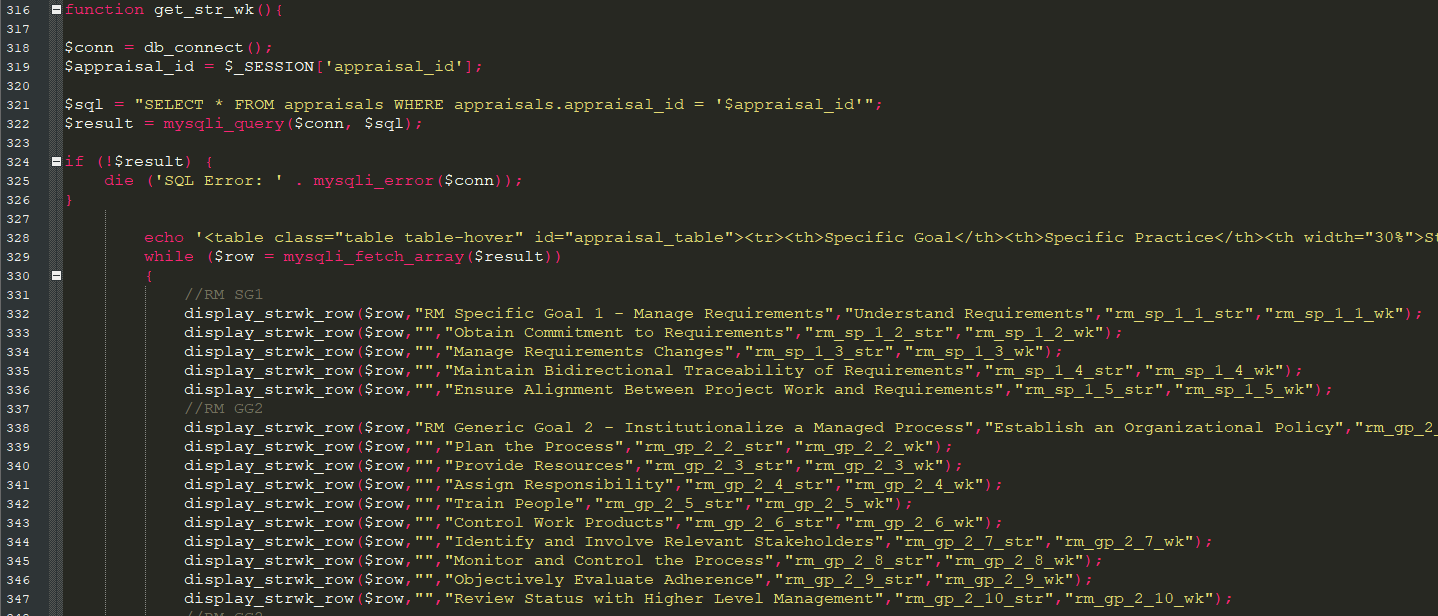


Figure 50 get\_str\_wk function

Figure 50 shows the get\_str\_wk function, which is primarily responsible for gathering the strength and weakness responses and presenting them in a table format. A table header is defined in row 328, and then a series of calls are made to the display\_strwk\_row function, which is used to generate any given row in the strengths and weakness table according to the parameters given.

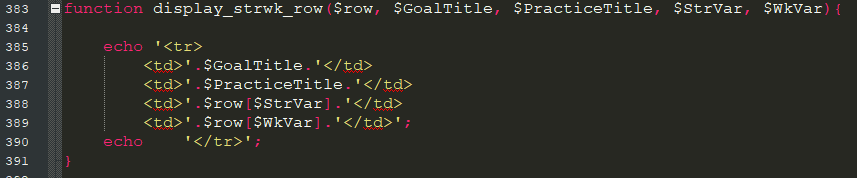


Figure 51 - display\_strwk\_row function

The display\_strwk\_row function is shown in figure 51. It takes five parameters:

1. Row. This parameter is taken from the parent function and is used to identify the row in the query results from which we will return the result. Row represents an array of results from the query
2. The goal title. This is an optional parameter used to define the heading for a specific practice
3. Practice title. This parameter is the title for the specific practice to which the strength and weakness belong
4. StrVar and WkVar. These specify the attribute name from the database from which the data should be extracted. This function is placed within a connection to the SQL database, so when $row[$StrVar] is called, this is calling the attribute name within the row that the query has returned from the database.

### 5.4.6 Manage Appraisals Interface

Given that the system is built to store a large number of appraisals it was necessary to create an interface that an administrator can use to search through the appraisals in the database and choose which appraisals to edit, along with seeing the completion status of any appraisal. This interface is shown in figure 52.

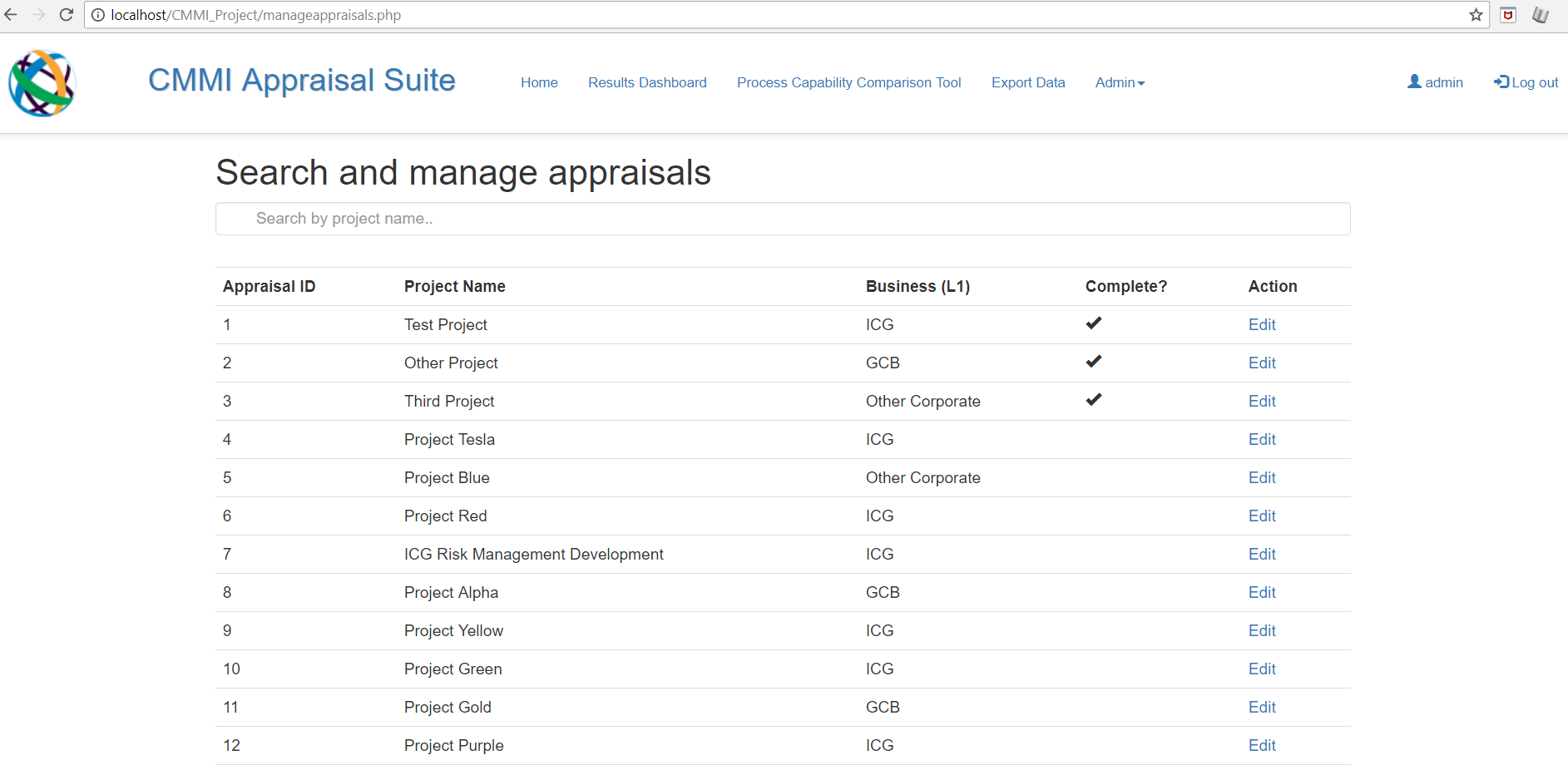


Figure 52 - Manage appraisals interface

There are 3 important aspects of this particular interface: how the table is generated, the dynamic search function, and the ‘edit’ links for each appraisal. These three aspects are discussed in further detail below.

#### Creating the Manage Appraisals Table

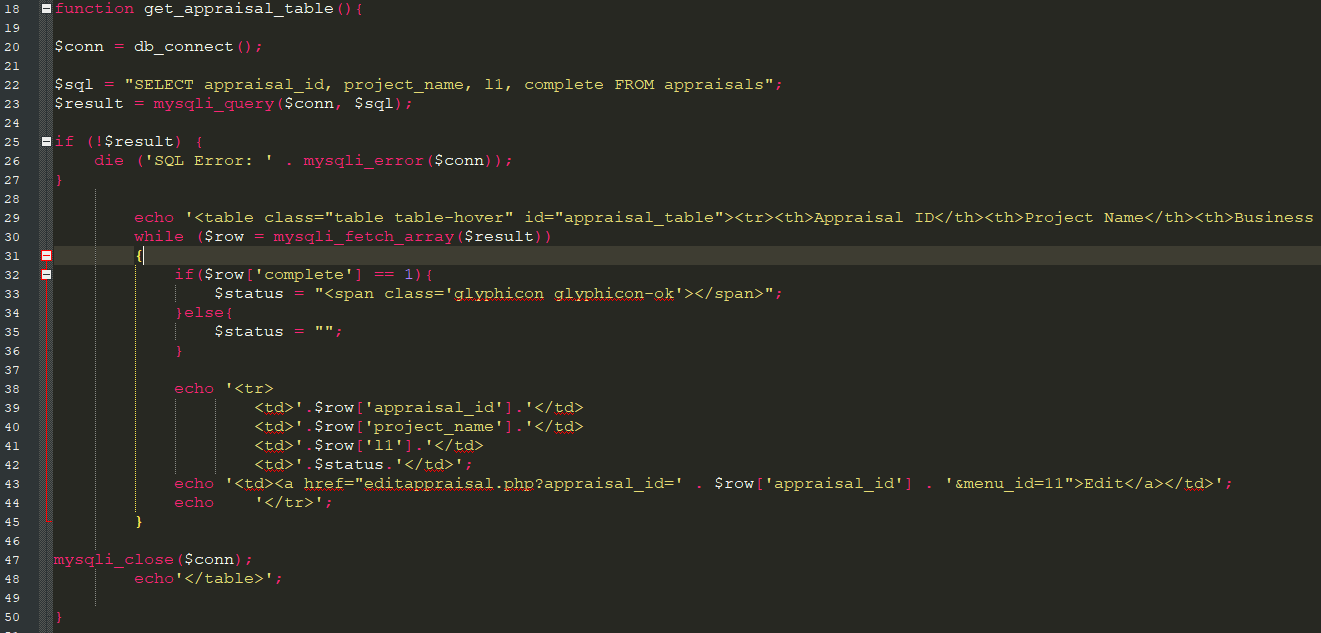


Figure 53 - get\_appraisal\_table function

The manage appraisals table is generated using the get\_appraisals\_table function, which is shown in figure 53. A SQL connection is established, which queries the appraisal ID, name, business name and completion status for each table. The results of this query are then placed into a loop, which loops through each row of the table converting the data into an HTML table row.

#### Dynamic Search Function



Figure 54 - Dynamic search function JavaScript code

The code used to program the dynamic search function for the appraisals table is shown in figure 54. This snippet of JavaScript defines the table structure (i.e. the rows and columns) as variables, while also declaring a filter variable (which uses the input from the text input box). The script then loops through every row of the table and hides any row that does not match the search term. This function executes every time the user updates the search bar, meaning that the filter updates after every character that is typed into the input box.

#### ‘Edit’ Links in Appraisal Table

Line 43 in figure 53 is the code that creates the hyperlink that the administrator can click to edit a particular appraisal, it is a blend of HTML and PHP. This link needs to be unique in every row of the table because each link must correspond to an individual appraisal. This has been achieved by using the appraisal ID (which is taken from the current row in the SQL query) as a URL parameter. This URL parameter is then stored as a session ID when the user navigates to the editappraisal.php page after clicking the link. This session ID then controls the data that is brought into the input and drop-down boxes, and is the ID used to determine which row of the database to update when the user clicks the save button.

## 5.5 Google Charts API

The Google Charts API has been used extensively throughout the system to produce visualisations of data in several key areas of the tool. The API itself is freely available online, however it cannot be downloaded and hosted directly on the website’s server. Instead, the library is called via a URL, which is shown in figure 55.



Figure 55 - Importing Google Charts API

The Google Charts API has been combined with PHP code that communicates with the database to produce an extensive set of live, responsive and dynamic charts. The techniques used to create these charts are discussed in further detail in the next part of this chapter.

### 5.5.1 Making Google Charts Dynamic

By their nature, Google-generated charts are static. Their data source is defined as a two-dimensional array which is then used to generate the chart. As such, a developer needs to create significant extra functionality to make these visualisations dynamic and to have them work with a SQL database.

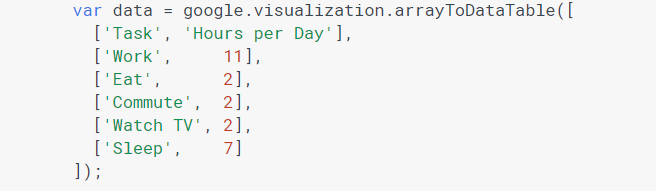


Figure 56 - An example of static Google Chart data [19]

In figure 56 above we see an example of a static dataset used to generate a Google Chart. In order to make these charts dynamic, there was a need to develop bespoke functions in PHP to query and extract results from the database that can be formatted and placed into the array. The functions created to do this should become evident as this chapter progresses.

### 5.5.2 Generating the Process Capability Bar Chart

When an appraisal is completed, the results overview generates a gap-analysis bar chart (seen in figure 48) to illustrate the capability of both process areas, and demonstrate the gap, if any, between how the process is currently performing and how a defined process should perform. Figure 57 shows the code used to generate the chart and display it on the webpage. Lines 30 and 31 of figure 57 load the packages from the API and instruct the webpage to execute the drawMaturityChart function on page-load. Within the drawMaturityChart function we see the chart data is defined and stored in a data variable in lines 35-38. On line 42 we see the declaration of an options variable; this variable contains commands to alter the appearance of the chart. Line 62 is where we declare and instantiate a chart object and instruct it to be placed into the HTML element with the ID “maturitychart”. On line 63 the chart object is drawn with the data and options variables as parameters.

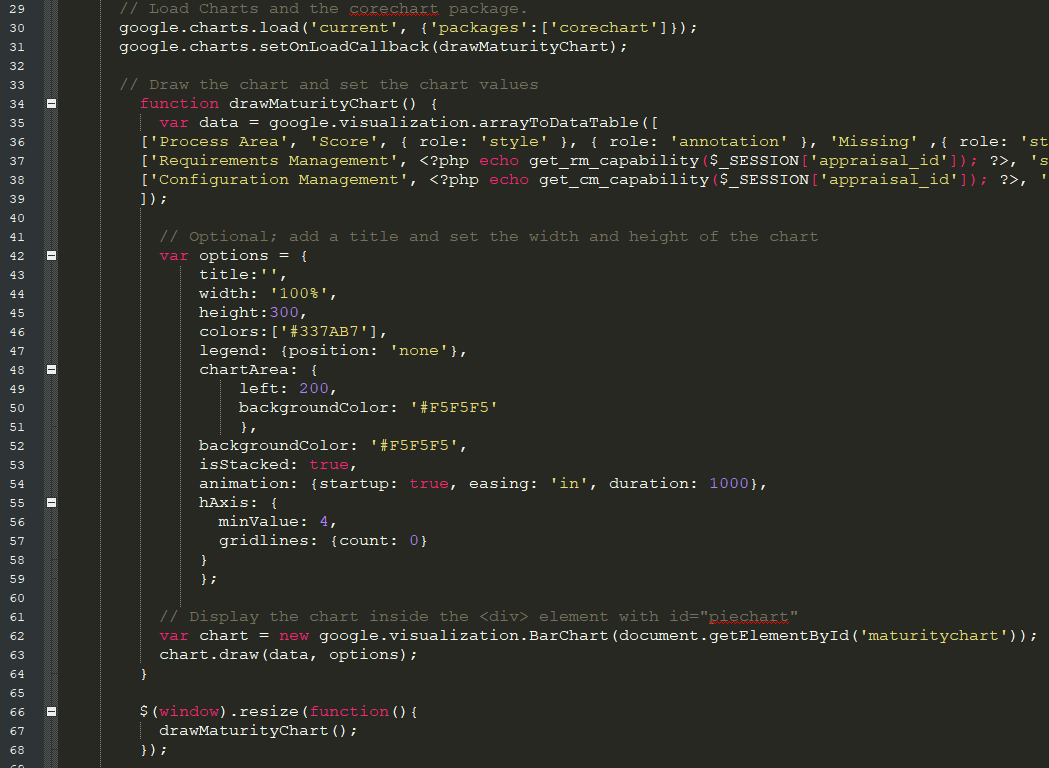


Figure 57 - Code to display process capability chart

In terms of preparing the data for this particular chart, there are three vital items that must be extracted from the database to inform the chart which information to present:

1. The capability level (either 1, 2, 3 or 4, ranging from not performed to defined)
2. The remainder of the stacked bar. Given that a stacked bar must always total the same amount (4 in this case), we must calculate the remainder to show as ‘missing’. In other words, if the process area is showing as a 2 (i.e., performed), then we must populate the other half with a 2.
3. The label of the performance level to be placed on the bar, i.e. not performed, performed, managed and defined.

#### Calculating the Capability Level

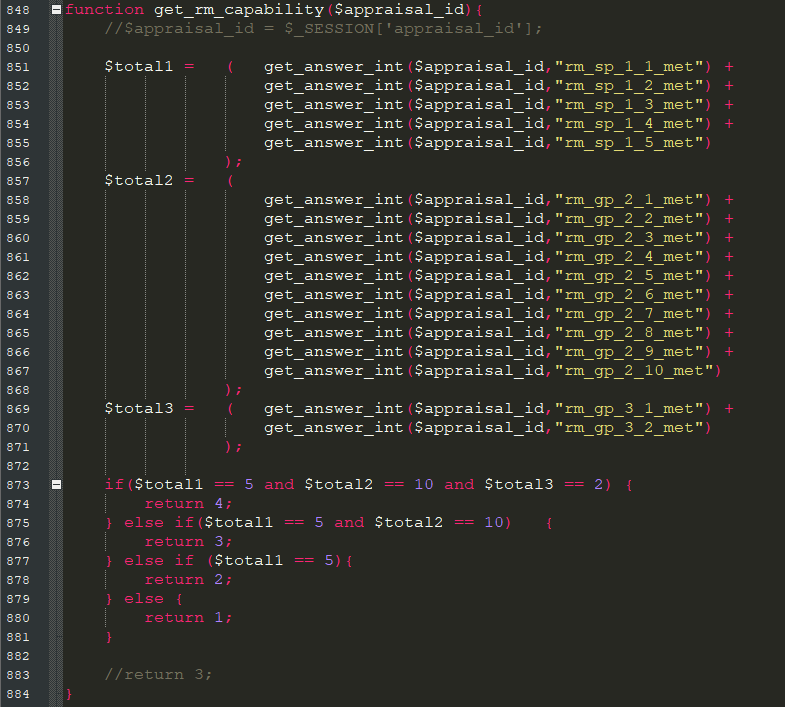


Figure 58 get\_rm\_capability function

The function used to calculate and return the capability number to be placed into the chart is shown above in figure 58. This function accepts the appraisal ID and question label from the session variable as parameters and determines which level the process is operating at be examining the \_met variables stored in the database for every specific practice and determining through a series of if statements what level the process is operating at. The data is extracted using the get\_answer\_int function, which is almost identical to the get\_answer function described in figure 38, except this particular function returns the result as an integer, as opposed to a string.

#### Calculating the Remainder

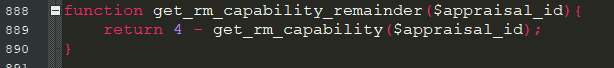


Figure 59 - get\_rm\_capability\_remainder function

The simple function in figure 59 calculates the capability remainder by subtracting the result of the get\_rm\_capability function (which will be between 1 and 4) from the number 4.

#### Extracting the Label

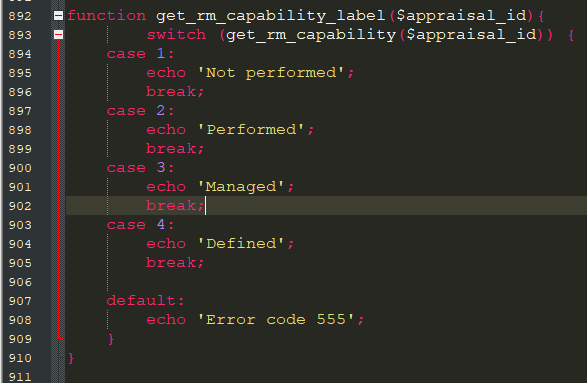


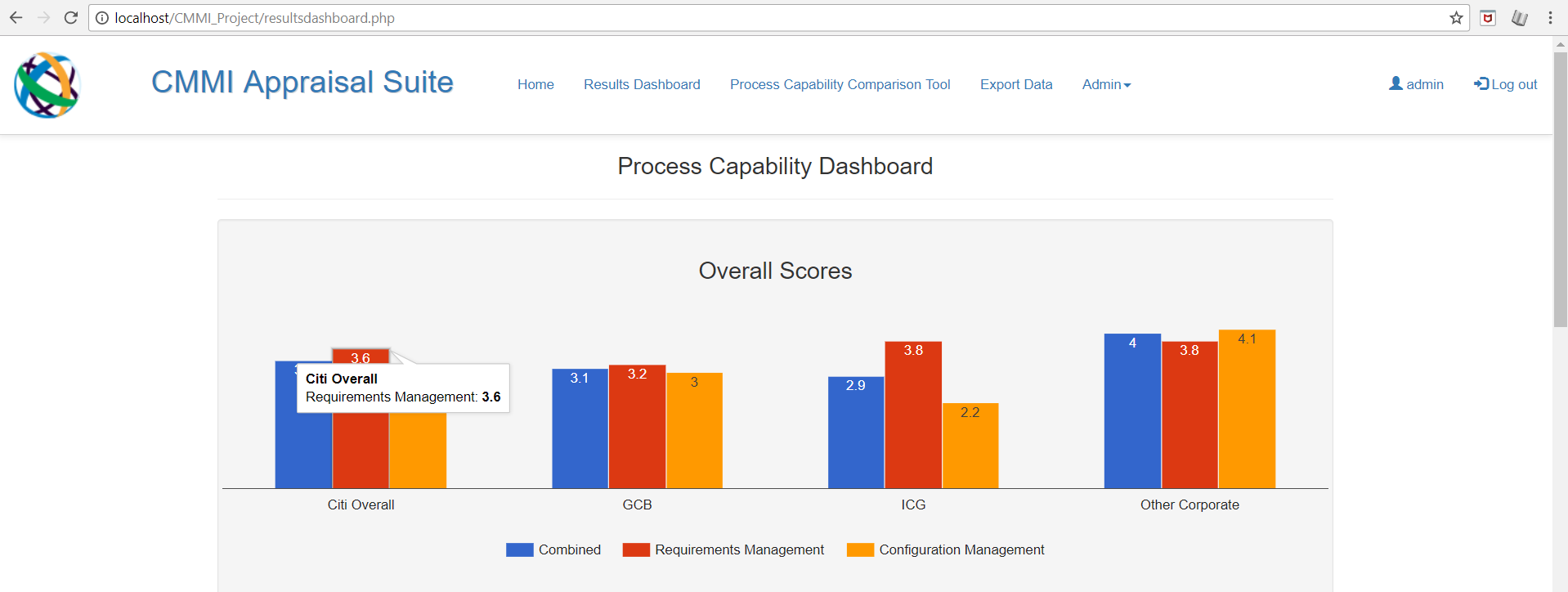
Figure 60 - get\_rm\_capability\_label function

The function shown in figure 60 determines which label should be shown on the bar chart by taking the result of the get\_rm\_capability function and passing it through a switch statement to determine which label should be returned and shown on the chart.

### 5.5.3 Bar & Column Charts

Several bar and column charts have been implemented in the tool to provide an avenue for qualitative analysis of the capability scores across Citi. Bar charts and column charts follow a similar implementation process in terms of leveraging the API, so rather than discussing the implementation of each instance of a bar or column chart across the tool, we will instead take a detailed look at the implementation of one particular column chart, shown at the top of the results dashboard, shown below in figure 61.

Figure 61 - Column chart in Results Dashboard page



This chart allows the user to see overall process capability scores across the 3 main lines of business in Citi. Note, that when we refer to a capability “score” that this is not the same as a capability “level”. A capability level will be either performed, managed, or defined. Whereas a capability score represents the average score from the repertory grid responses across all practices, or a chosen set of practices. So, in figure 61, we see the tool-tip above is displaying a “Citi Overall” score for Requirements Management as 3.6. This means that across the whole of Citi (GCB, ICG and Other Corporate combined), the average score across all of the repertory grid constructs relating to the specific and generic practices for Requirements Management is 3.6. While this number does not determine an organisations capability “level” in Requirements Management, it can serve as a useful KPI to understand the quality of the processes being executed across the business in relation to requirements management.

The process for declaring and presenting the chart is much the same as the chart discussed previously in this chapter. The main difference lies in the extraction, definition and transformation of the data from the SQL database.

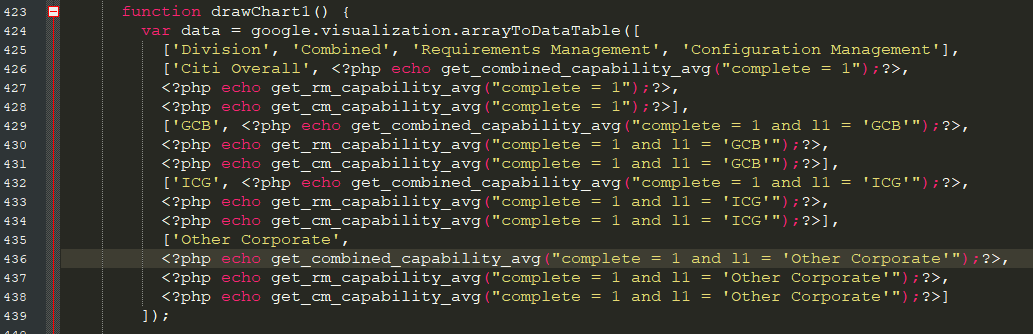


Figure 62 – Data declaration for bar chart

Figure 62 shows the data variable that has been declared for the bar chart shown in figure 61. There are three main functions that have been developed with the responsibility of extracting the data for this chart from the SQL database. These functions are get\_combined\_capability\_avg, get\_rm\_capability\_avg and get\_cm\_capability\_avg. These three functions operate in exactly the same way except one retrieves data for all practices, one retrieves requirements-only practices, and the other retrieves configuration-only practices. Rather than examine all three of these functions, we will take a deeper look into one function, which should provide the understanding of how the other two functions work also.

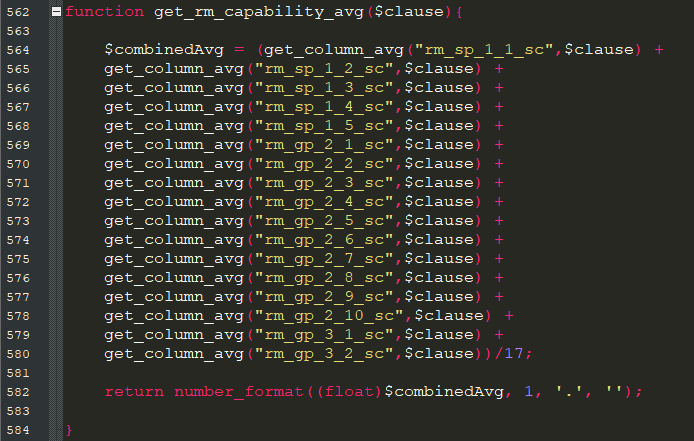


Figure 63 - get\_rm\_capability\_avg function

The get\_rm\_capability\_avg function is shown in figure 63. There is one parameter required: “clause”. This parameter accepts a string, and its purpose is to add additional filtering functionality to the function. Using the clause parameter, the user can specify a WHERE statement to be added to the SQL query to extract a specific set of results – this will be seen more clearly when we examine the get\_column\_avg function. This function collects an average score for each specific practice, and then produces a final score by taking an average of all the combined practice scores. While this is actually an “average of an average”, which is usually not mathematically robust, in this case the impact of this method is negated because each dataset has the same number of results in it (i.e., the total number of completed appraisals).

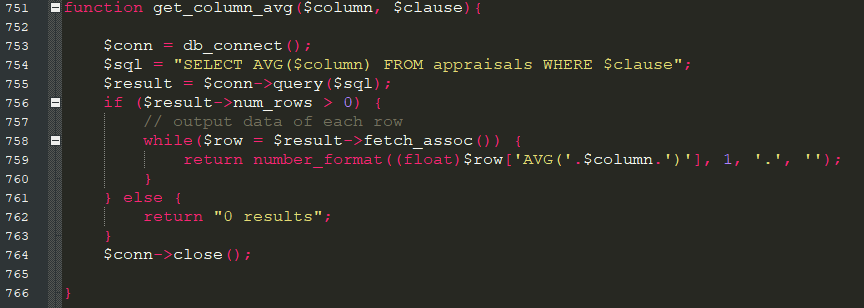


Figure 64 - get\_column\_avg function

The get\_column\_avg function (figure 64) uses two parameters to extract and return an average of a column (attribute) from the SQL database. The first parameter, $column, refers to which column we need to extract the data from – this can be any numeric column in the SQL database. The second parameter, $clause, is a parameter of string type that allows the user to specify a specific WHERE clause to be appended to the SQL query to retrieve a specific set of results.

### 5.5.4 Preparing Data for the Scatter Chart

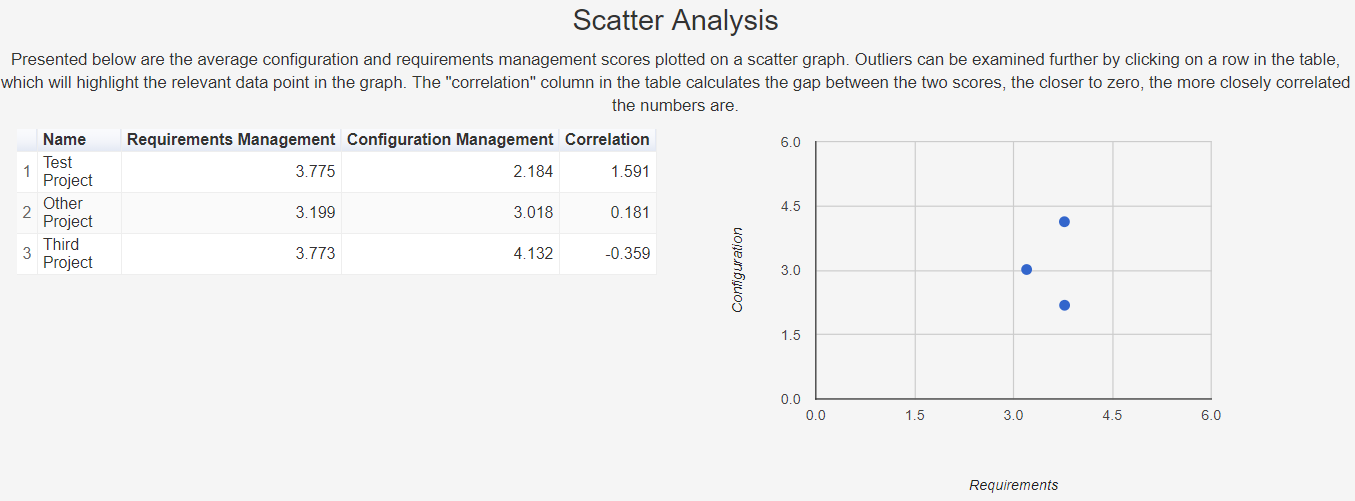


Figure 65 - Scatter chart and table

For the scatter chart displayed in figure 65, a more complex data format was required which called for some further bespoke functions to be created that were capable of extracting and transforming the data from SQL.

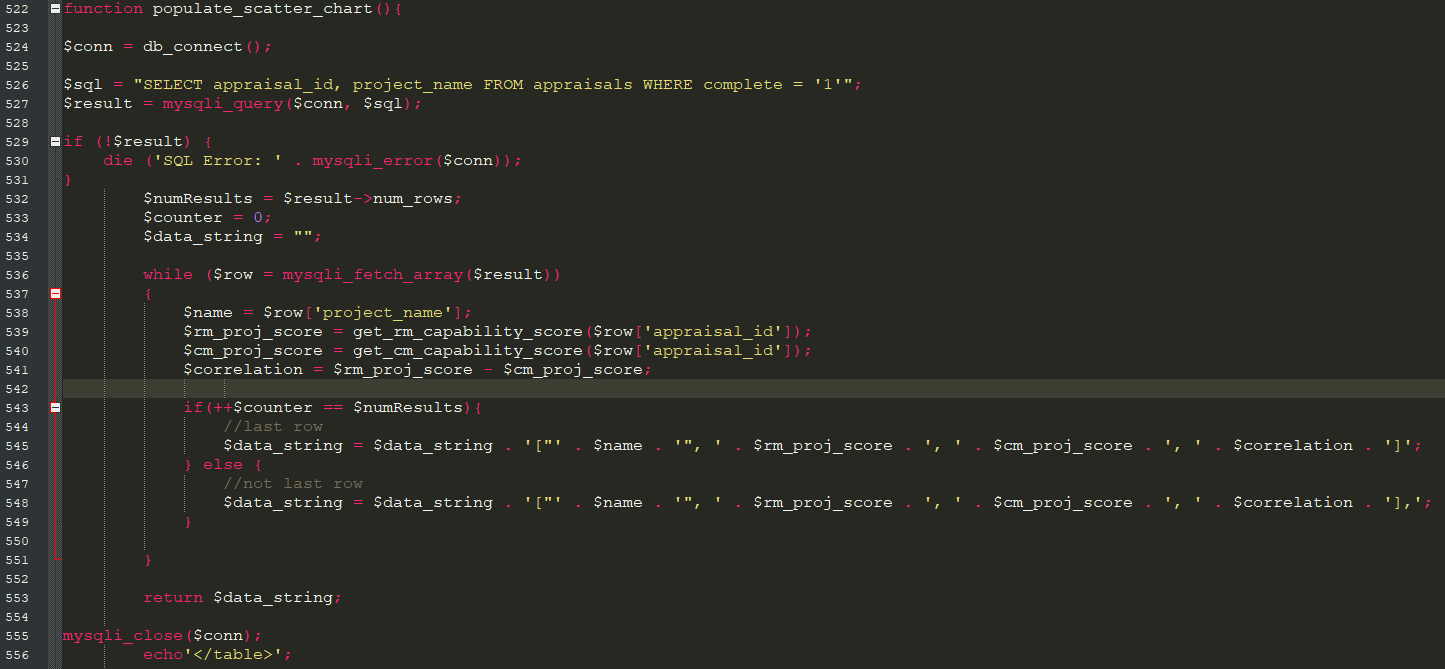


Figure 66 - populate\_scatter\_chart function

The function shown in figure 66 is used to extract the data for the scatter chart, and is to be called inside the data array for a Google Chart object. This function creates a connection to the SQL database and performs a loop which formats the chosen data into a comma separated arrary (which is the format required for the Google Chart). This function places calls to two other functions inside the loop which perform the mathematical functions to obtain the scores: get\_rm\_capability\_score and get\_cm\_capability\_score.

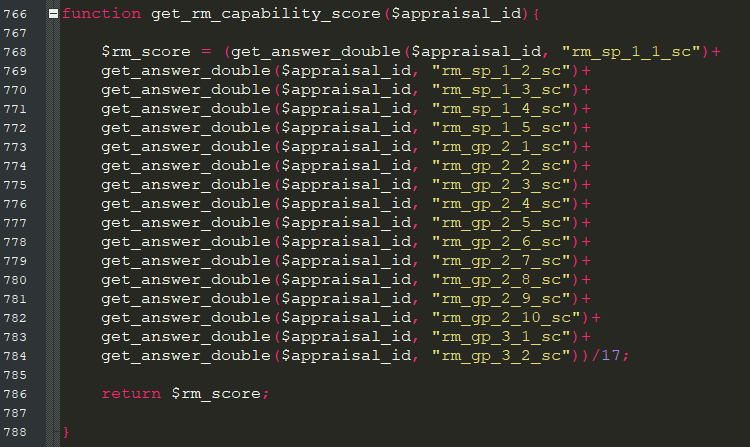


Figure 67 - get\_rm\_capability\_score function

The get\_rm\_capability\_score requires one parameter, which is the appraisal ID corresponding to the row in the database that we are going to extract the information from. The data is extracted using the get\_answer\_double function, which is almost identical to the function described in figure 38, except the data is returned in double format.

### 5.5.5 Adding Event Listeners to Charts

Additional functionality was added to the chart and table shown in figure 65 by adding an event listener to the chart. When the user clicks on a row in the table, this highlights the data-point in the chart that the table row represents. This allows the user to analyse outliers by sorting the table on the “correlation” column and selecting the highest value, this will be the project with the largest gap between requirements and configuration management scores. The code to achieve this functionality is shown in figure 68.

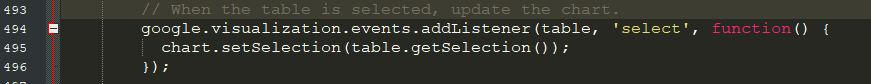


Figure 68 - Google Chart event listener code

### 5.5.6 Comparison Dashboard

One of the features implemented in the tool for further analysis is the comparison dashboard, shown in figure 69. This feature allows the user to specify two different projects for comparison against each other. The user fills in the form at the top of the page, and presses submit. The system then calculates the numbers and visualisations for those two chosen projects and presents it back to the user. There is also a facility that allows the user to choose a chart type to present the data, in this case, either a line or a bar chart. This functionality was added to partially satisfy the custom chart requirement which was identified as a “nice-to-have”. The comparison dashboard also offers a tabular view of the data, which presents the scores for all practices for the two chosen projects side-by-side. The data is also colour coded to demonstrate at a glance which project has the highest score for that practice. A tick or a cross is added into each score-box to illustrate if a practice has either passed or failed.

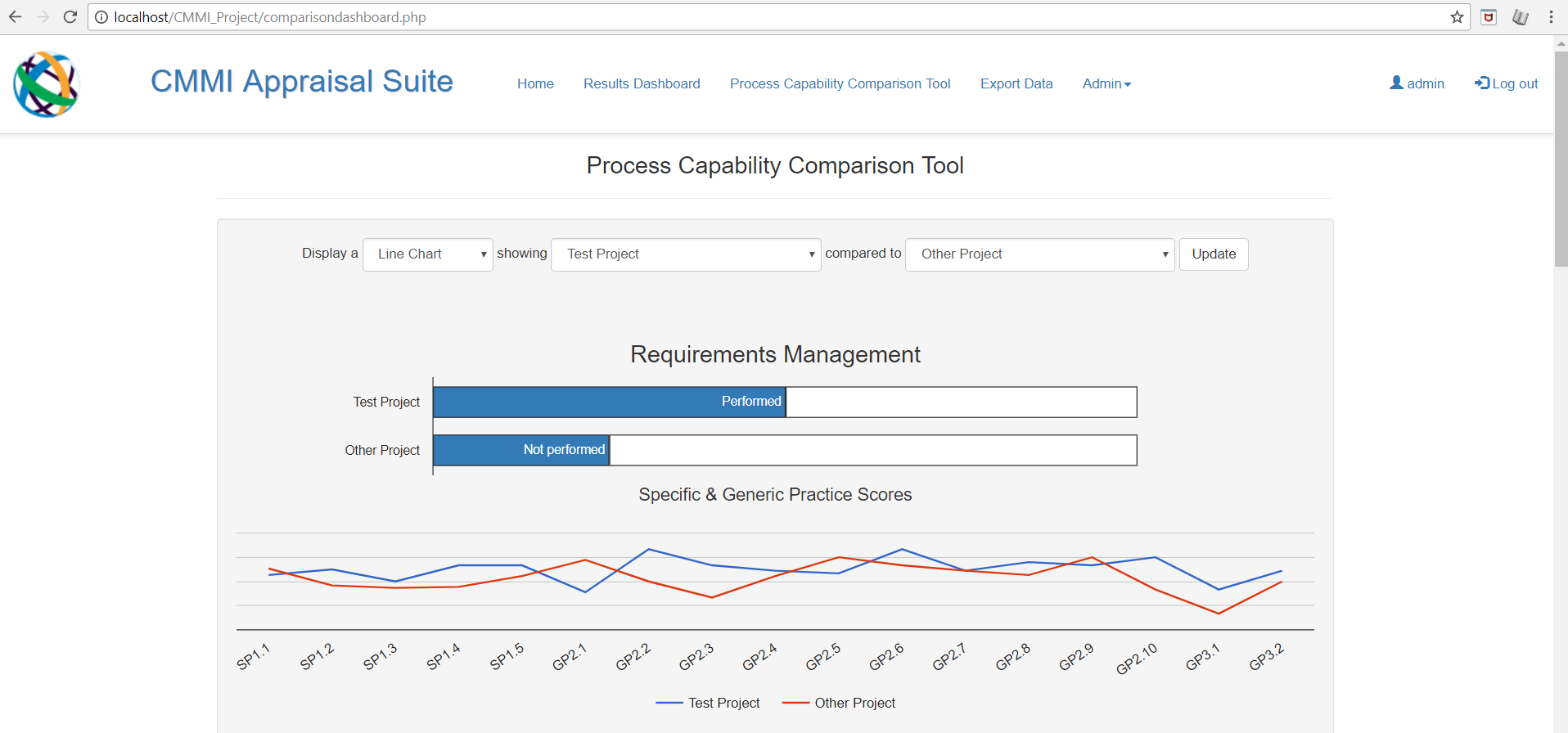


Figure 69 - Comparison dashboard charts

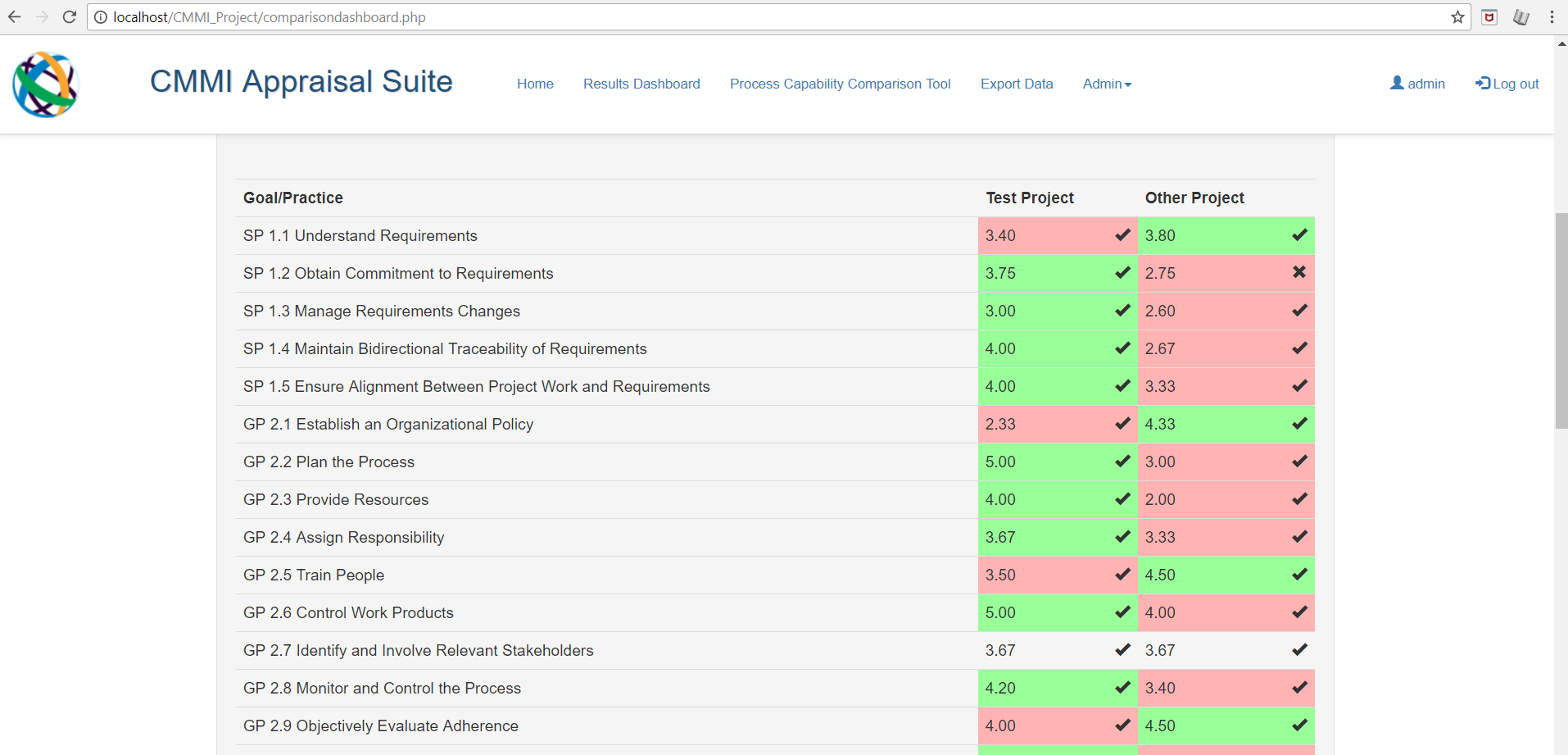


Figure 70 - Comparison dashboard table

#### Creating the Form for the Comparison Dashboard

Creating a form to capture user entry is ordinarily a straight-forward affair. However, in this case, a complication was added in that the project choices chosen from the dropdown menu (which can be seen in figure 69) needed to somehow be capable of relating back to the database by a primary key. All of the “get” methods in the system that are capable of extracting information from the database in relation to a particular project use the appraisal ID as the identifier to match with the row in the database to be extracted. As such, simply populating the drop-down menus with the project names would not be sufficient, because we cannot use the project names in the “get” methods to extract the data. Populating the drop-down boxes with IDs alone would not be user-friendly either, because this would require the user to compare the list of IDs to another source to find the project they were looking for. Therefore, the solution is to extract the list of project names and IDs at the same time from the database, and present them in the drop-down menus with the project name acting as the “label” and the appraisal ID acting as the “value”. This solution works because the user can select a project name from the list, which is associated to an ID, and it is the “value” part of the drop-down list that is actually stored in the $\_POST variables after the form is submitted. Therefore, we can use this data in the existing “get” methods.



Figure 71 - populate\_dropdown1 function

The function shown in figure 71 is responsible for displaying these drop-down lists, there are two functions responsible for populating the drop-down lists, one for requirements management and one for configuration management. The function creates a SQL connection that extracts all of the project names and appraisal IDs that are stored in the system. A PHP echo statement then returns these pieces of information in HTML format, with the appraisal\_id acting as the value and the project\_name acting as the label. This echo statement is placed inside a while loop that will cycle through all of the results of the query, meaning that we create a drop-down entry for every project in the database.

#### Obtaining and Storing User Selections

The comparison dashboard is inherently interactive, allowing the user to make choices and have those choices reflected in the data being displayed back to them. With this in mind, it was necessary to develop a form to collect the users’ choices so that they could be received by the system and stored in variables for use in displaying the visualisations according to the users chosen parameters.

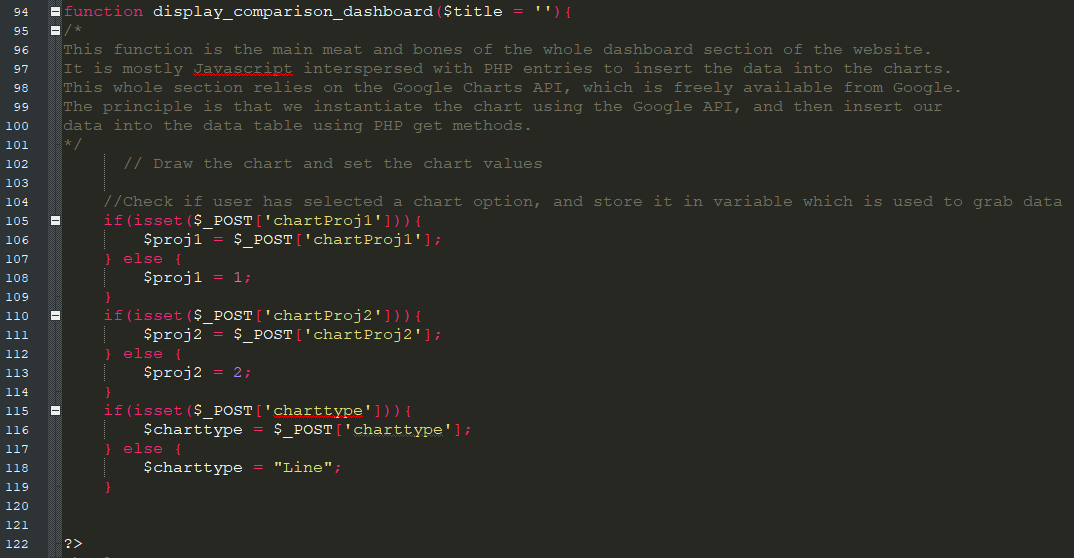


Figure 72 - Code to extract user choices in comparison dashboard

The code shown in figure 72 checks the $\_POST variables from the form to see if the form has been submitted. If values are held in these variables then they are stored into the PHP variables $proj1 (which is the ID for the user’s first chosen project), $proj2 (which is the ID for the user’s second chosen project) and $charttype, which is the user’s choice for the format to display the chart. All three of these variables are used later in the code to render the charts and tables according to the user’s chosen parameters.

#### Capability Gap-analysis Charts in the Comparison Dashboard

The capability gap-analysis charts featured in the comparison dashboard use the same functions as the results overview page to generate the gap analysis capability charts – get\_rm\_capability, get\_rm\_capability\_remainder, get\_rm\_capability\_label, get\_cm\_capability, get\_cm\_capability\_remainder, get\_cm\_capability\_label. These are discussed in figures 58-60, with the only difference this time being that we use the variables gathered from the user’s form entry described above as the parameters to identify the row in the database that we wish to return. There is an additional function required for the comparison capability gap analysis charts which is responsible for getting the project name of the user’s chosen projects, get\_project\_name, which is shown in figure 73 below.

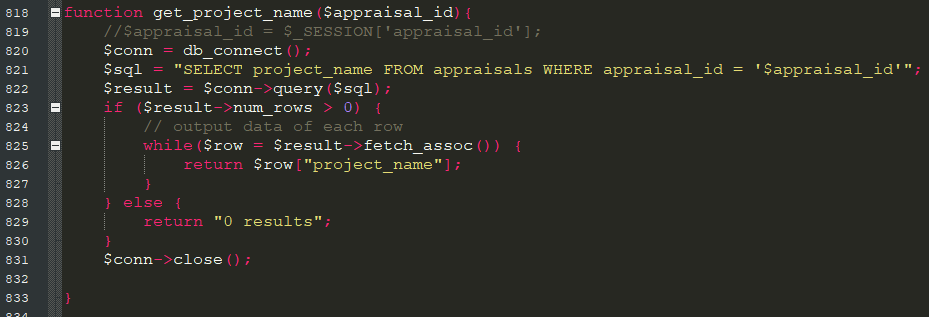


Figure 73 - get\_project\_name function

The function in 73 is a function designed to extract a project name of the user’s choice from the SQL database. The project name that is returned is based upon the $appraisal\_id parameter, which is used to identify the chosen row to return from the database.

#### Drawing the Custom Line/Bar Chart with the User’s Parameters

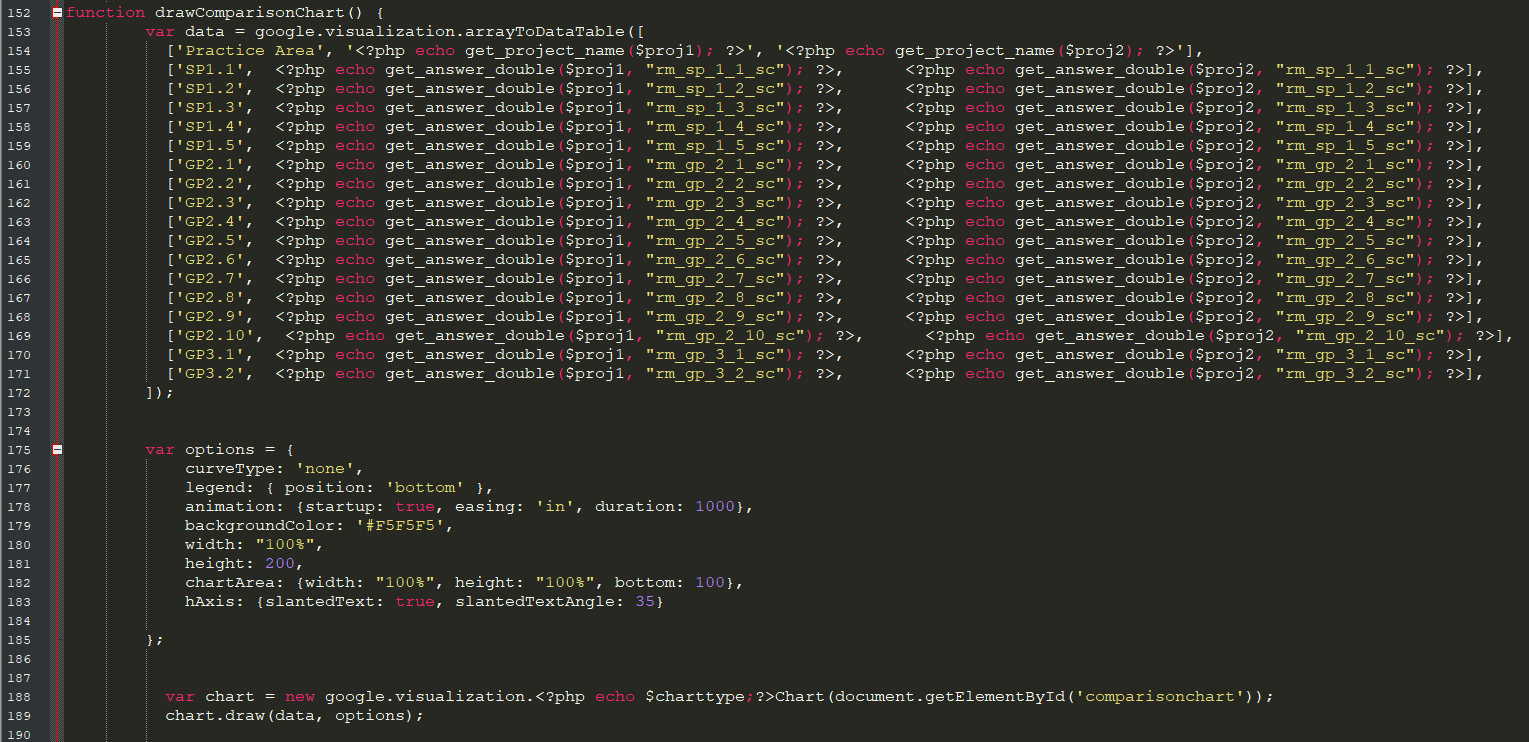


Figure 74 - Code to draw the custom line/bar chart

Figure 74 presents the code used to prepare the data and draw the chart for the requiements management version of the custom line/bar chart in the comparison dashboard. The two-dimensional data array has a row for each specific and generic practice related to requirements management. Within each row we have three pieces of information, the name of the practice, and the score of that practice for each of the user’s chosen projects. The get\_answer\_double function is used to retrieve the answers from the database – we have encountered this function before in figure 38. This time we place the variables $proj1 and $proj2 into the parameter list, because these hold the project IDs corresponding to the row in the database which we wish to extract, as discussed previously in this section.

Lines 188-189 in figure 74 show the code that instructs the system to show either a line or a bar chart depending on the user’s choice. This is achieved by echoing the $charttype variable into the command to draw the chart object. This variable will either hold “line” or “column”, which will trigger the two different chart types available when echoed.

#### Generating the Colour-coded Comparison Tables



Figure 75 - get\_comparison\_table function

Figure 75 shows the get\_comparison\_table function which is the function we call to display the table in the comparison dashboard for requirements management. There is a version of this function available for configuration management called get\_comparison\_table\_cm. This function gathers the information from the $\_POST variables, like we saw in figure 72, which are used to identify the data to be extracted from the database. The function then defines a table structure and calls the display\_comparison\_table\_row function for every row of data that it wishes to display – this function is shown in figure 76 below. It accepts 4 parameters: the title of the specific or generic practice to be displayed in the table, the appraisal IDs for the two projects that are to be compared, and the attribute from the database that we wish to extract.

Lines 179 to 188 in figure 76 are to establish the colour coding in the table, where the highest score of the two is shown as green and the lowest as red. If the two scores are identical, then no colours will be applied to that row. Lines 190-204 use the get\_answer function (described in figure 38) to display a tick or a cross in the box depending on whether or not that specific practice passed or failed. Lines 207-211 format the data defined earlier in the function and output it with integrated HTML to form a table row that can be echoed into a table.

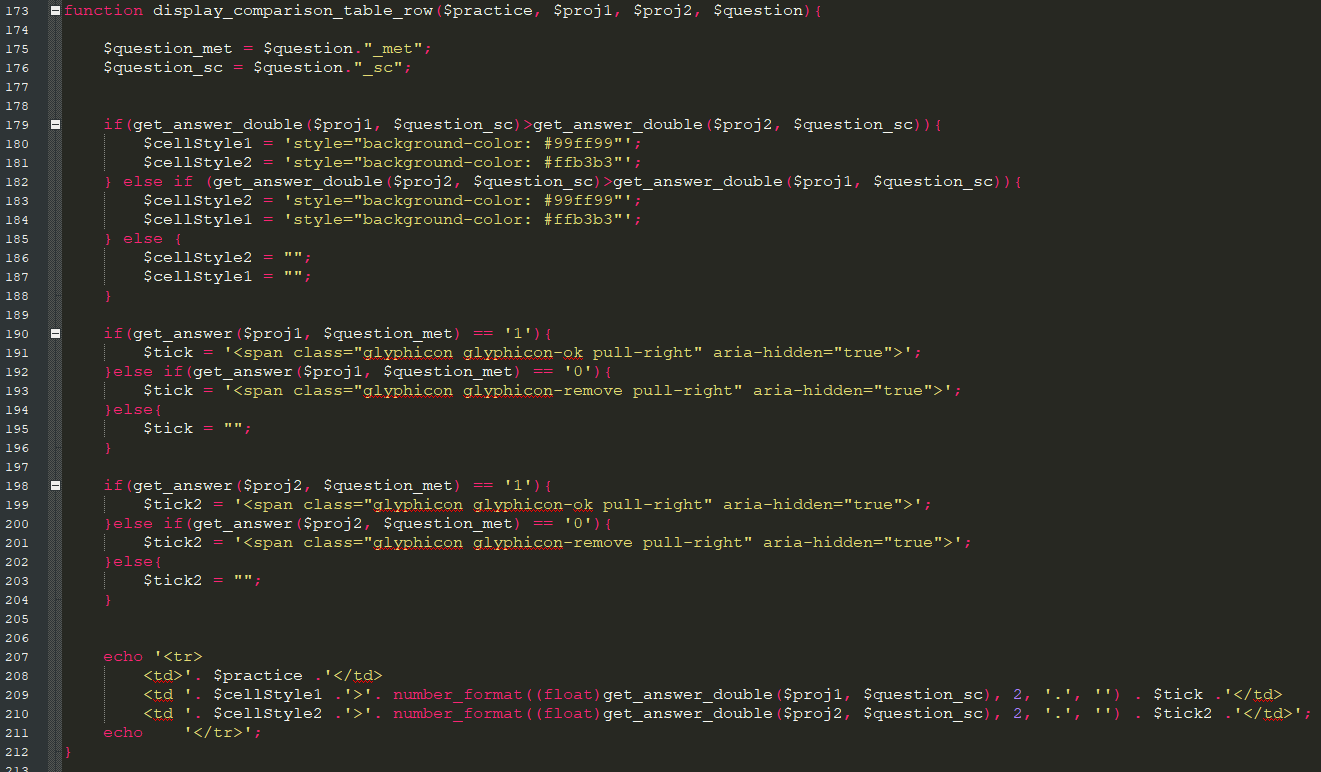


Figure 76 - display\_comparison\_table\_row function

## 5.6 Export Data Functionality

An export data function has been created to allow the administrator to export all data in the database to .CSV format for further analysis in excel or a BI tool if desired. The export\_data function, which is shown in figure 77, is the function responsible for achieving this. A SQL query is executed that selects all data from the database. SQL queries return in PHP without headers, so it was necessary to add extra code to the SQL query to select all of the headers, which are then appended to the top of the query using a UNION statement. The SQL query is then placed into a CSV file using the PHP fput command. The resulting file is downloaded via the user’s browser after the system places it there through the use of a PHP output stream.

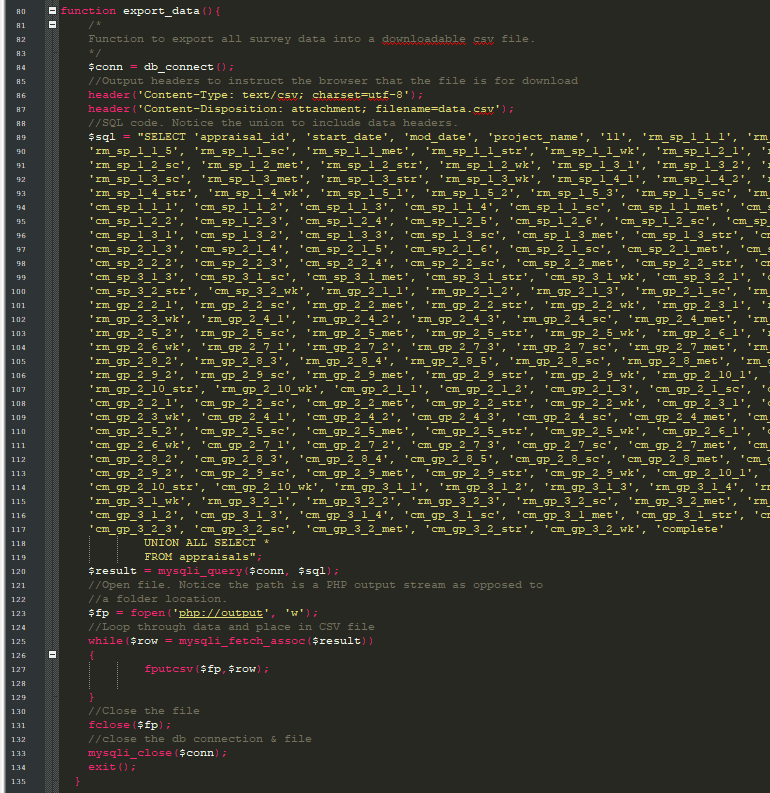


Figure 77 - export\_data function

## 5.7 Github & Atlassian

To mitigate against the risks of data loss and corruption, SourceForge’s Atlassian GitHub repository tool has been leveraged to manage the source code of the project and provide various points of backup should the code ever become corrupted. An example of Atlassian being used throughout the project is shown below in figure 78. There were no cases where a backup was required through development, but Atlassian provided extra security should that possibility have actually occurred.

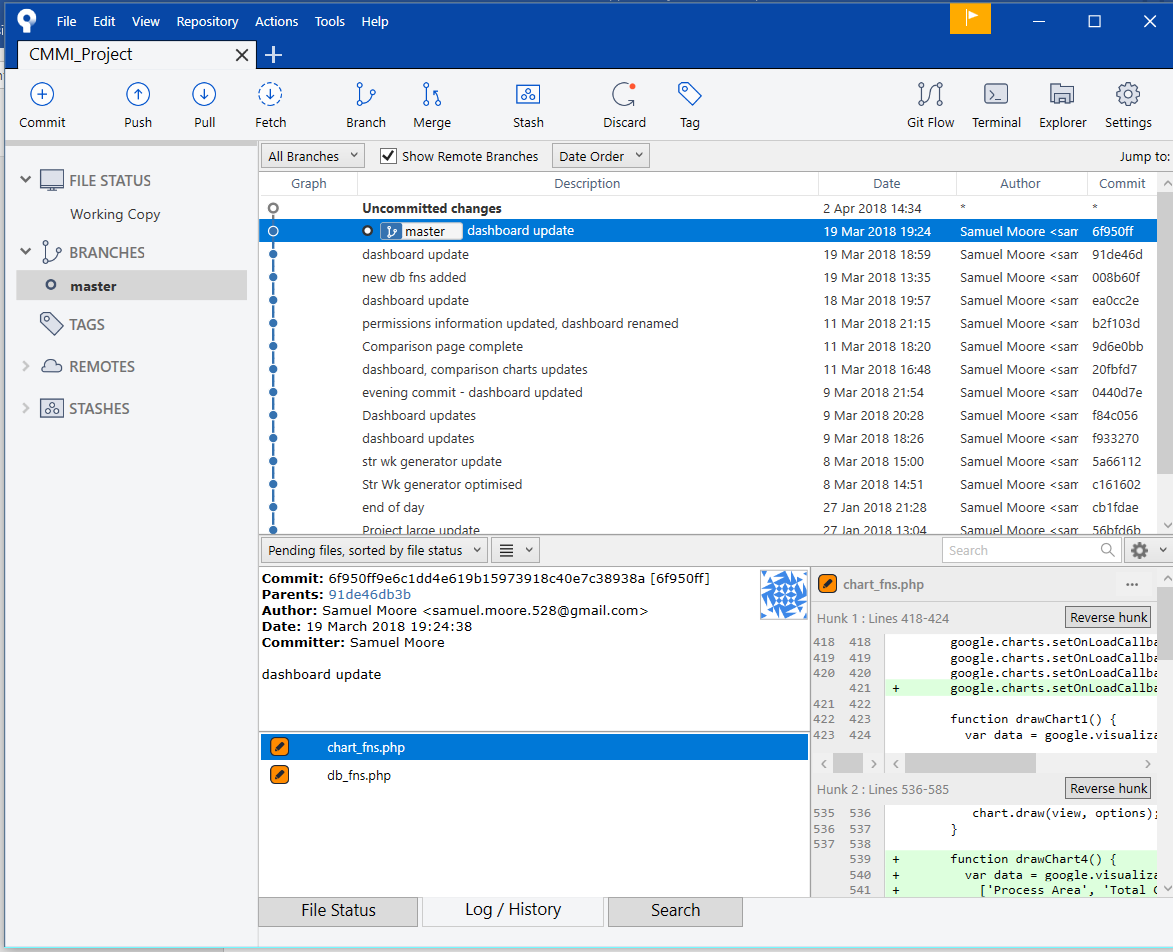


Figure 78 - Using Atlassian

# 6. Testing, Verification & Validation

This next chapter focuses on the testing, verification and validation aspects of the project. This chapter will outline a testing strategy, discuss the unit testing approach, discuss peer review of work products, requirements acceptance testing and end user feedback.

## 6.1 Testing Strategy

Testing is one of the key aspects involved in the release of any software product. Testing ensures that an application is robust, effective and conforms to expectations prior to it being released. Errors caught in testing can be resolved without ever having to reach public eye. In terms of software releases, it is much more effective to have a rigorous testing strategy than to have to release several bug-fixes after the product has already gone live. Given the revised waterfall methodology in use here, it would actually be possible to return to implementation after testing has been carried out, if required, in a prototyping fashion. However, this was not expected or required, largely due to time constraints in developing the application. There are three main strands of testing which form the overall test strategy for the application:

1. Unit Testing (Testing)
2. Peer Review (Verification)
3. Acceptance Testing of Requirements (Validation)

These three strands of testing have been carried out and are documented throughout the remainder of this chapter.

## 6.2 Unit Testing

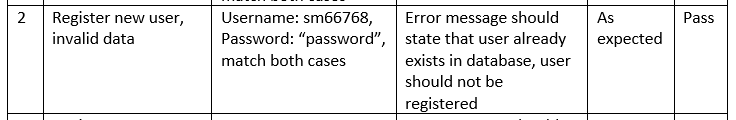
Unit testing has been used to ensure that functionality is performing as expected across the product. The tests were completed over two different devices, a Windows laptop and an LG Android smartphone. A sample unit test is provided below in figure 79, and the full list of unit tests can be seen in Appendix G.

Figure 79 - Sample unit test

## 6.3 Peer Reviews

Peer review is an important aspect of the validation process in ensuring that a product meets expectations in terms of it’s design, build and delivery. Given that the target audience of this project is Citi, there have been various peer reviews of some key work products throughout development.

#### Review of Functional Requirements

Stakeholder: Rosemary Teggart, Internal Audit - Technology

One of the stakeholders from Internal Audit, Rosemary Teggart, was able to review the requirements specification that was created for the tool. In general the feedback was positive. It was noted that the functional requirements lacked detail, if they were to be used in an industry-standard project, however it was agreed that they were likely sufficient for a student project, especially since there was no development team involved which eliminated the possibility of requirements ambiguity to development teams, or development teams misinterpreting requirements.

#### Review of Design Documents: Usecases, Activity, Sequence

Stakeholder: Michael Ervin, Internal Audit – Market Operations

Michael Ervin was an ideal candidate to review design documents, because in his capacity as a Market Operations internal auditor, he frequently interacts with and designs UML diagrams. Michael gave some constructive feedback on the use-case diagram regarding the “extends” notation – this feedback was acted upon and reflected in the use-case model. Michael’s other feedback was generally positive

#### Review Session of Final System

Stakeholders: Alison Neate (Internal Audit – Management Information), Christopher Clokey (Internal Audit – Technology), Rosemary Teggart (Internal Audit - Technology), David Albery (Compliance - Testing), Jonathan King (Compliance - Testing)

Several stakeholders across Citi were invited to view the tool in face-to-face sessions and give feedback on it. The main themes of the feedback have been consolidated and presented below:

|  |  |
| --- | --- |
| **Positives** | **Negatives** |
| * The responsive nature of the website is impressive | * CMMI could use a little more explanation |
| * The implementation of the Google Charts API was well received – users enjoyed the interactivity | * Charts can occasionally be slow to load in circumstances of poor internet |
| * Appraisal interface was clean and uncluttered |  |
| * People were impressed with the functionality in the comparison dashboard |  |
| * Graphically and aesthetically, people enjoyed the website |  |

## 6.4 Requirements Acceptance Criteria

An important activity involved in the verification of software is checking the software against the acceptance criteria laid out against the original requirements scope. This is an important traceability exercise that ensures that software is designed and executed as it was planned to be, and as the stakeholders expect it to be. Presented below are the acceptance criteria that were identified in the Requirements Capture & Analysis chapter and an update on the success or failure of these tests.

| No. | Requirement | Acceptance Test | Met? |
| --- | --- | --- | --- |
| FR1 | The system shall have a registration facility | A user is able to register an account on the system. This registration facility must prevent users from registering the same account twice. | Met |
| FR2 | The system shall have a log-in facility | A user is able to log in and out of the system using verified credentials gained from the registration process. Unauthenticated users should be denied access to the system | Met |
| FR3 | The system administrator shall be able to assign layered permissions to users on the system. | The administrator (and only the administrator) is able to delegate access to view dashboard results to a registered user | Met |
| FR4 | The system shall allow users to conduct the CMMI appraisal by filling in a questionnaire. | The user is able to conduct a full appraisal on the system | Met |
| FR5 | The system shall produce a strengths and weaknesses document upon the completion of the questionnaire | The user is able to view a strengths and weaknesses document upon completing the questionnaire | Met |
| FR6 | The system shall use a bespoke algorithmic scoring mechanism to establish the extent to which a process is enacted | An algorithmic score shall be displayed demonstrating the capability of a process once an appraisal is complete | Met |
| FR7 | The system shall be capable of displaying a gap analysis style of chart once the appraisal is complete. | The user can view a gap analysis chart for each completed appraisal | Met |
| FR8 | The system shall include a back-end database capable of storing appraisal information | The mySQL database can successfully store information entered from the front-end system | Met |
| FR9 | The system shall include a visual dashboard feature | The user is able to view results of appraisals in a dashboard format | Met |
| FR10 | The system shall be capable of displaying predesigned queries from the database as “key metrics”. | The user is able to view key metrics in the dashboard | Met |
| FR11 | The system shall feature a sand-box style “custom query builder” | The user is able to execute a custom query from within the system | Partially met |
| FR12 | The system shall feature a sand-box style “custom graph builder” | The user is able to design and view a custom graph from within the system | Met |
| FR13 | The system shall be web-hosted | The user can access the system using a web browser | Met |
| FR14 | The system shall be mobile and tablet compatible | The system can be used on smaller-screen devices | Met |

### 6.4.1 Requirements not Met, or Partially Met

There was one requirement, FR11, that was only partially met. It was deemed late in the development stage that this requirement would be too costly in terms of time to implement, and potentially cause risk to the integrity of the data in the system. This particular requirement was deemed as a “nice-to-have”, and therefore was not essential. As a halfway measure, the export data function was built instead. This export data function gives the user full access to the data in Microsoft Excel, which in theory allows them to manipulate and query the whole dataset as they need to, which is the purpose that the custom query mechanism had set out to achieve.

# 7. Appraisal Execution

In order to evaluate the quality of the software, it was deemed necessary to execute an appraisal on a real-life project in Citibank so that the software can exhibit its effectiveness or demonstrate areas for improvement. The Compliance Securities Surveillance (CSS) team in Citi were identified as a test case for the project. This team is responsible for building bespoke software platforms that are used across Citibank globally to monitor and control market securities, primarily to safeguard against fraud or money laundering. These software platforms are produced frequently and under tight time constraints coupled with high pressure. The software tool was used to analyse Requirements Management capability with respect to a recent project concluded in early 2018. Due to confidentiality requirements, it is not possible to disclose the detail, or name of this project.

The appraisal was conducted over a series of conference calls with members of the CSS team, wherein the software tool was used as the questioning and data capture mechanism. On the whole, the CSS team conduct their requirements management activities to a high standard, where documentation and dissemination of information is of high priority. However, there is still some room for improvement, which if implemented could strengthen the teams process capability for requirements management.

## 7.1 Observations & Actions

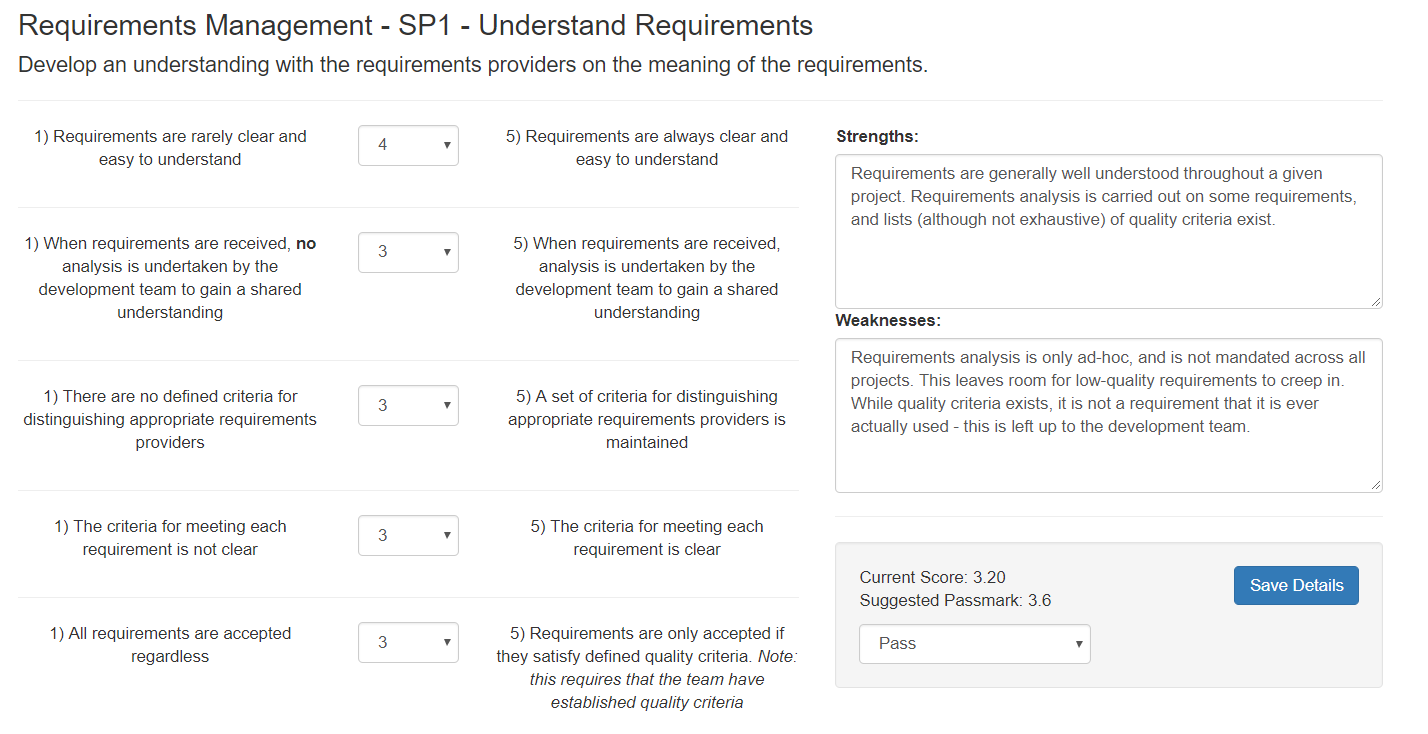
A set of observations, both good and bad, have been documented below as a result of the appraisal.

### 7.1.1 Observation 1: Requirements analysis activities should be mandatory

The first thing to be noticed when looking at figure 80 below is that a score of 4 has been given for the first construct of “requirements are clear and easy to understand”. The second thing of note, is that the scores relating to requirements analysis activities all are scored at 3. In general, requirements are very well understood across the project. There is rarely a case where the development team or project manager do not actually understand the requirements – this, of course, is a positive. However, what should be noted here is that this has been achieved without always performing requirements analysis activities to gain a shared understanding. This is not to say that analysis is never conducted on activities, but rather, it is not mandatory. So, in pressured time scenarios, this analysis activity is often seen as surplus to requirements and not conducted. On the surface we might suggest that if the requirements are generally well understood, then we don’t need to change anything. However, in this case, the requirements are only well understood as a result of the efforts of the people involved, rather than relying on the strength of an analysis process to gain understanding. The people involved in the project put extra effort in during the requirements process to gain an understanding from their stakeholders, normally through lengthy negotiations, rather than relying on analysis activities, such as passing the requirements through a quality criteria gate to ensure that the requirements meet the standards expected by the team. Ultimately, the requirements are understood well by the team – which is the crux of this specific practice. However, this understanding could be achieved more easily and with fewer instances of non-compliance by the simple implementation of a requirements quality checklist.

**Action:** Implement a requirements quality checklist as a mandatory and documented process.

Figure 80 - CSS Project - RM SP1



### 7.1.2 Observation 2: The process for obtaining commitment to requirements, and the process for changing requirements are both very strong

One of the key strengths in the CSS team’s requirements management process capability were in the areas of obtaining commitment to requirements, and managing requirements changes, both shown below in figures 81 and 82. Commitment to requirements is seen as an essential activity in Citibank and is documented and held in a Citibank-wide system called PTS (Portfolio Tracking System). It is not possible to proceed to the build phase of any project without submitting documented commitment for a requirement on PTS. The change request process is handled through JIRA, which is packed with features to help manage requirements changes, including; audit trails, impact assessments and the system will not allow users to proceed until the necessary documentation has been submitted.

**Action:** None

Figure 81 - CSS Project - RM SP2

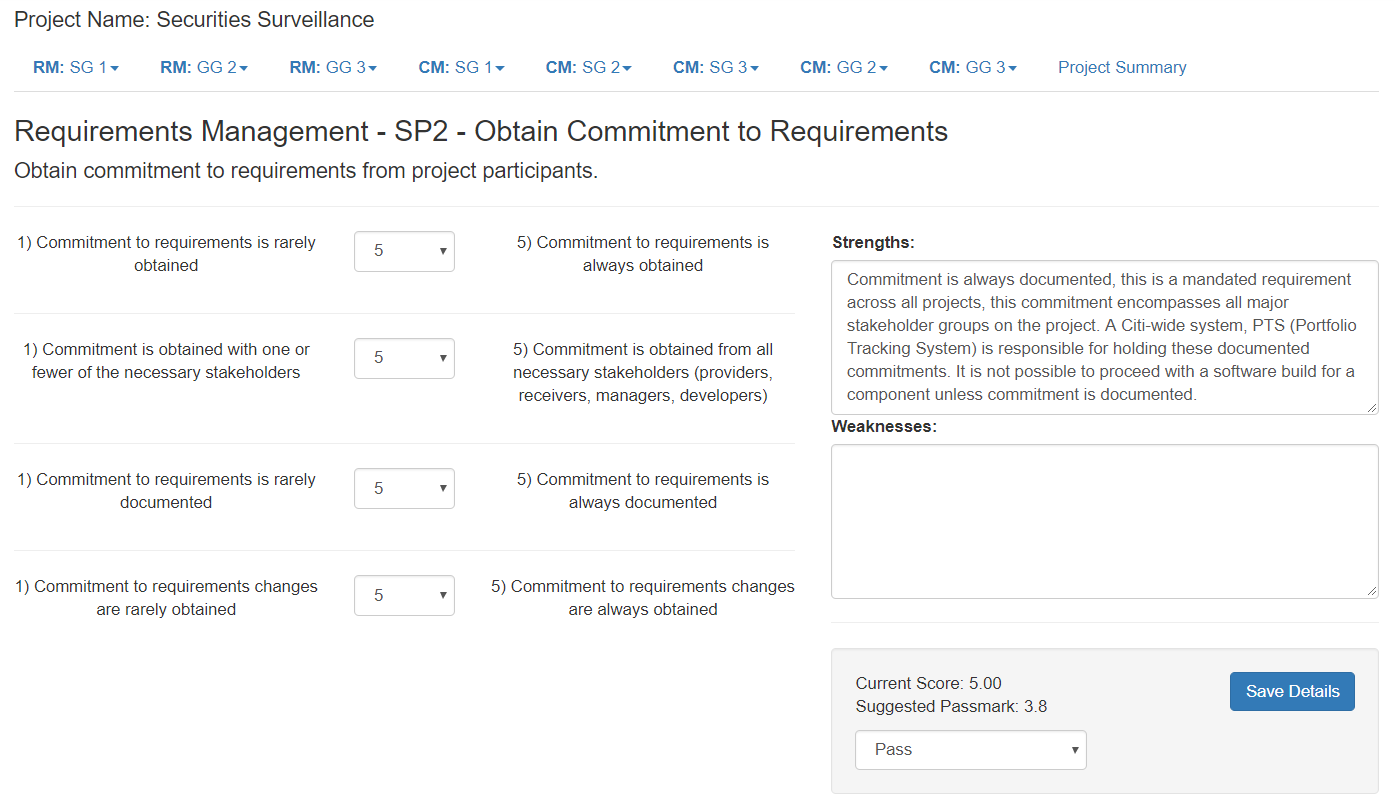
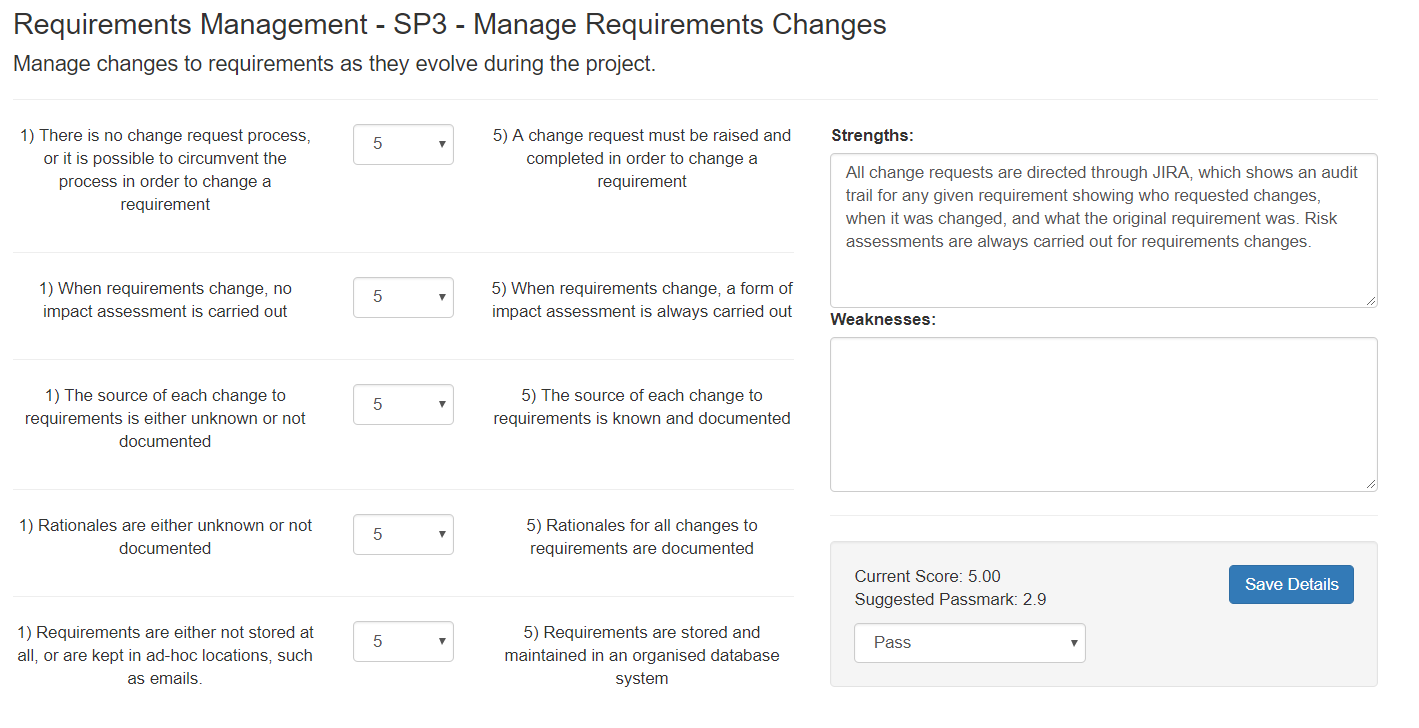


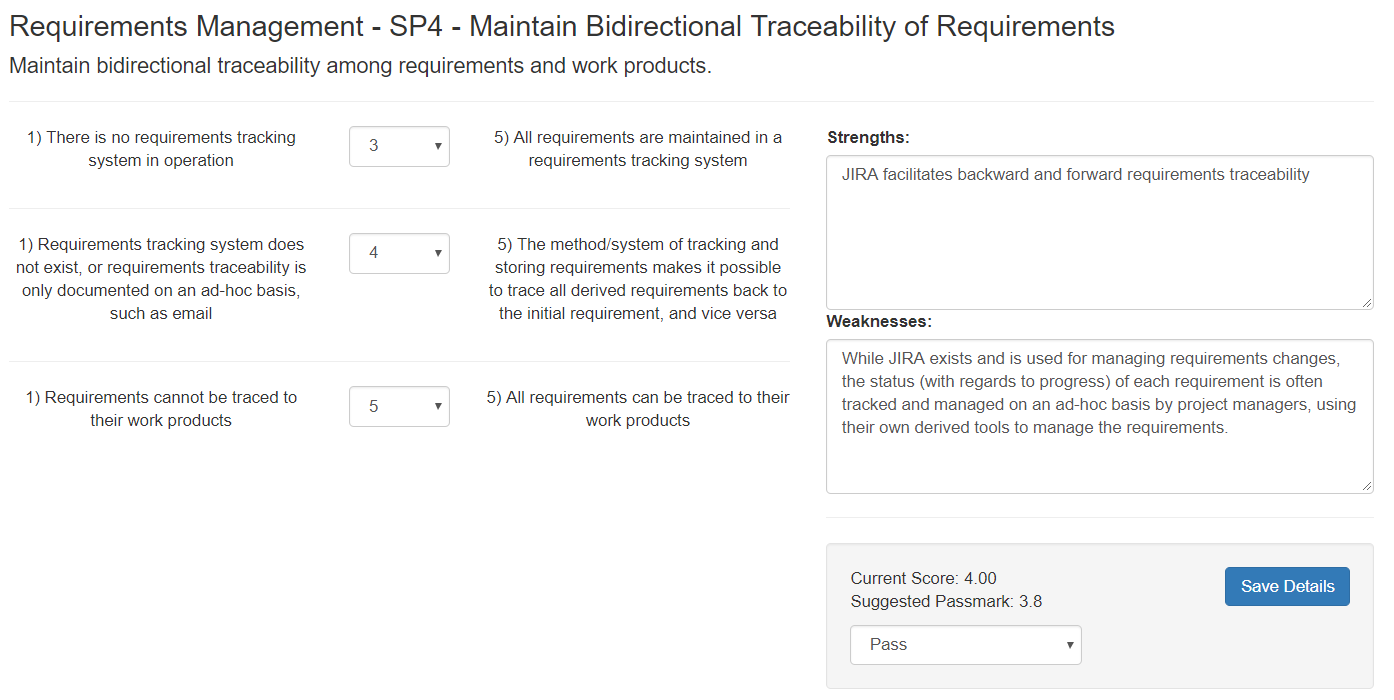
Figure 82 - CSS Project - RM SP3



### 7.1.3 Observation 3: Requirements progress should be tracked through JIRA

Requirements tracking activities do take place, and requirements traceability can easily be achieved through JIRA. However, one area for improvement would be that requirements are often tracked through ad-hoc systems developed and maintained by project managers, seen in figure 83 below. While this has been sufficient in achieving the needs of the project, it does carry an element of risk and relies on the ability of the project manager, rather than on the strength of the process itself. It is strongly recommended that requirements tracking activities be conducted through a formalised system to achieve consistency and stability in the process.

Figure 83 - CSS Project - RM SP4



## 7.1.4 Conclusion & Next Steps

This pilot exercise proves the validity of the software. The software is capable of achieving the objectives that were laid out at the onset of the project: diagnose weaknesses, diagnose strengths and suggest areas for improvement.

The next steps for the CSS team would be to implement the suggested improvements and undergo a full analysis cycle to ascertain if improvement has been achieved.

# 8. Evaluation

As a measure of success, the project aims and objectives have been assessed against the final result. This analysis should provide a picture of how effective the project has been in achieving its purpose.

## 8.1 Project Aim

“**The aim of the project is to create a CMMI appraisal tool which generates sub-practice-based questions to establish the extent to which a specific practice is enacted. The extent scale shall be based on a classification technique known as repertory grid. The tool will establish strengths and weaknesses based on the body of knowledge contained within the CMMI.”**

### 8.1.1 Justification

**Success.** This aim has been completed in full. The tool generates sub-practice based questions as required and allows the user to establish the extent to which a specific practice has been enacted, through the use of repertory grid scoring. The tool outputs strengths and weaknesses for each appraisal.

## 8.2 Project Objectives

1. Project Analysis & Planning

* This objective was met through the production of a detailed project plan which was submitted in the early stages of this assignment.

1. Requirements Gathering

* Extensive requirements gathering documents have been collated and are discussed in the Requirements Capture & Analysis section of this report. This objective was met successfully primarily through examining data in Citi, observations and stakeholder interviews.

1. Design

* This objective was met through the production of use-case, activity, sequence and data-flow diagrams, which are evidenced in the Design chapter of this report.

1. Implementation

* The implementation phase was completed successfully and is documented in the Implementation chapter of the report.

1. Testing

* Thorough testing has been conducted on the completed system and is documented in the Testing, Verification & Validation chapter of this report.

1. Pilot

* This was achieved when a pilot assessment was conducted with the CSS team, and the results of this are documented in the Appraisal Execution chapter

1. Deployment

* This objective has been achieved, and the final product is hosted on the university’s web server.

1. Documentation

* Project documentation has been completed.

1. Review & Evaluation

* Review and evaluation have been completed.

## 8.3 Areas of Strength

All in, it should be stated that the project was a success – evidenced not only by its timely delivery, but also in the strength of the products produced themselves. Below are some of the key areas in which this project succeeded without which the project would have lost much value.

### 8.3.1 Volume of Work

Volume should not always be used as an indication of the quality of a project. After all, what use is a system that can produce 1,000 tables if only one of those tables is actually useful? The project should actually be judged on the number of useful features or outputs that it creates. In the case of this project, the volume of outputs is even more of a strength considering the tight timescales that the project had to compete with, especially when combined with the developer’s full-time employment. The project included a very high volume of useful and quality outputs. For example, the decision to assess two process areas right up to capability level 3 was a huge undertaking. Going to capability level 3 on one process area represents a large amount of work, especially considering that in order to assess CL2, 10 individual generic practices would need to be assessed, on top of the specific practices assessed at CL1. CL3 then requires the assessment of two more generic practices. This volume was then effectively doubled by adding a second process area for assessment. This is evidenced by there being more than 250 attributes per row in the database just to hold the appraisal information alone. Assessing these process areas was not a straight-forward or linear approach either: the CMMI does not offer a simple check-list that could be lifted and placed into the software tool without further thought. Instead, weeks of detailed research were carried out into both of the process areas and repertory grid technique. This detailed research then produced a novel and well-informed approach to assessing process capability.

### 8.3.2 Quality of Research

Often an under-appreciated facet of any project, the literature review section of this report demonstrates the important role that detailed research played in the development of this project. In order to gain the most useful insight and information, many journals, books, websites and conference papers were read in connection with the CMMI, repertory grid, personal construct theory and system design methodologies in order to produce a full picture of research.

### 8.3.3 Novel Approach

Assessing a company’s CMMI process capability is by no means novel, given that the CMMI and repertory grid are not novel techniques by themselves. However, what was novel about this project was the combination of these two techniques to assess process capability, coupled with the system that was developed to collect the information and visualise the results – this unique combination cannot currently be found on the market, which is testament to the creative nature of the project. This project opens the door to a new approach to leveraging the CMMI.

### 8.3.4 Project Management

With the volume of work undertaken, and the new territory being explored with the implementation of a novel approach, project planning would take on a vital role in ensuring the success of the project. None of the major deadlines were missed with regards to the project, and the system was fully implemented on-schedule.

### 8.3.5 Implementation of Google Charts API

The use of the Google charts API, shown in figure 84, is a significant strength in the system. Care was taken to design visualisations that were informative to the user and added depth to the insight that can be gained from the CMMI. In particular, the process comparison dashboard allows the user to visually compare two projects at a glance, without having to painstakingly examine excel spreadsheets or tables. This creates an ease-of-use that is utterly invaluable to any end-user. The results dashboard, shown in figure 85, also demonstrates an effective use of this API by allowing the end-user to examine all completed appraisals from a “birds-eye” view. This aggregated approach to reporting allows key business leaders to understand evolving trends in relation to process capability, or to hone into particular areas of strength or weakness throughout the business. These tangible benefits found in both the results dashboard and the comparison dashboard transform the system into a deeply practical tool that truly enhances user knowledge in relation to process capability.

Figure 84 - Use of Google Charts API

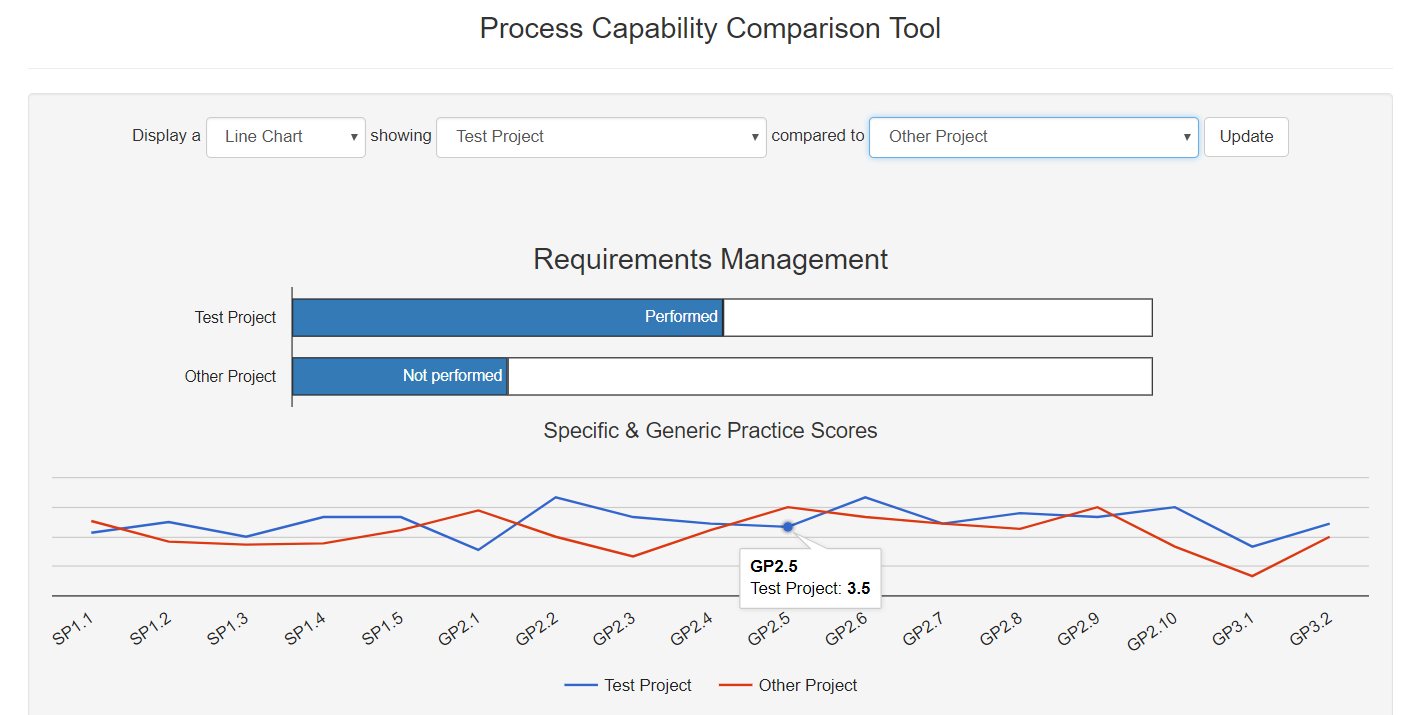
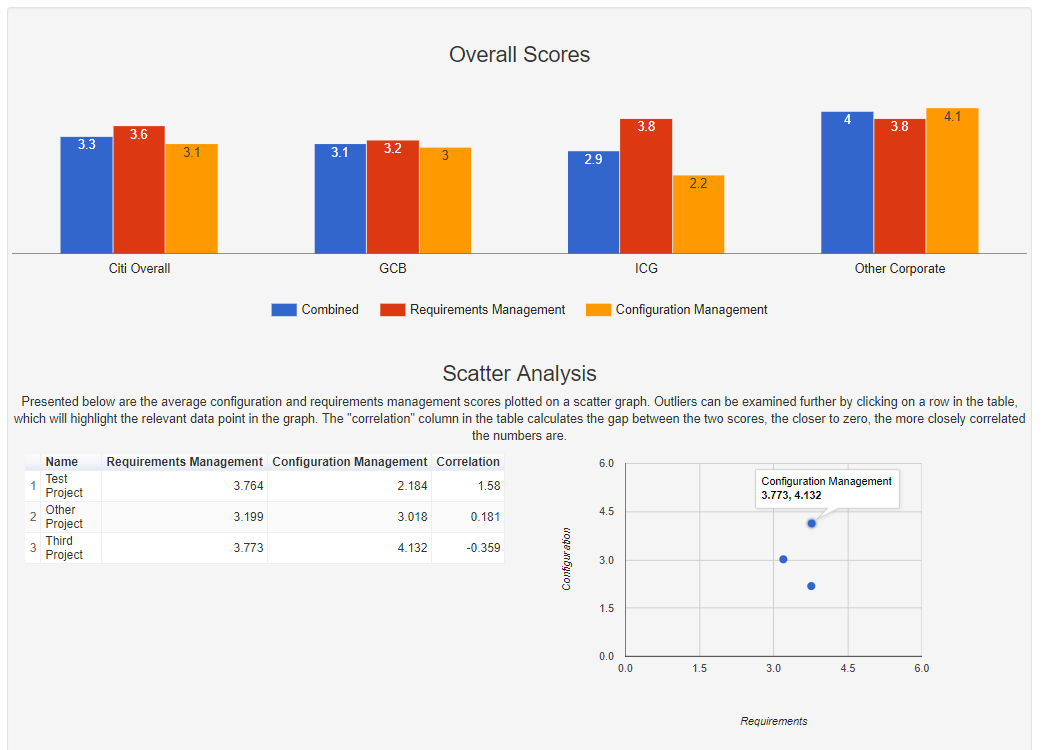


Figure 85 - Google Charts API in Results Dashboard



## 8.4 Areas for Improvement

The development of the system was a success, as shown through the justification of the completed objectives. However, that does not mean that a perfect system was created. In many ways, the resultant system was limited by time constraints and there being no budget to work with. The system was a success within these constraints, but if the constraints were removed, then a higher quality system could be developed. In essence, this project represents two projects in one. There is a project to develop a piece of software, and a project to use this software to produce analytical insight into process capability. In some ways, the project had to operate with a dual focus. Ultimately, if the software had been the sole focus, then the software may have been richer in terms of features. Or, if the appraisal itself had been the main focus, then more process areas could have been assessed at Citibank. It became clear when building the software that implementing two process areas represented a huge undertaking if the desired level of detail were to be included. The SQL database holds more than 250 distinct columns of data, all of which relate to individual scores to assess process capability – this exhibits the volume challenge involved in building the software for two process areas. However, despite this dual-focus, this project does lay a very firm foundation for process improvement and can be used in a vast array of business areas within Citi. The ground-work has been laid should Citibank wish to develop this product further.

## 8.5 Future Development

#### Include All Process Areas From the CMMI

An obvious improvement would be to expand the tool to include all process areas. This would theoretically, if implemented correctly, allow Citibank to use the tool to assist in conducting a Citi-wide staged approach assessment using the CMMI.

#### More Extensive Dashboard & Querying Power

The system would benefit from a more extensive dashboard feature. Ultimately, this would best be achieved by implementing a business intelligence platform that uses the tool as a data source. BI platforms offer much greater flexibility and detail in producing dashboard analysis platforms.

#### The Ability to Add/Remove Repertory Grid Constructs

The repertory grid constructs built into the tool are static and can only be edited through re-developing the system. If Citibank were able to invest the time and money, then it would be possible to change the database structure and website interface to allow the user to dynamically add or remove repertory grid constructs for any given specific or generic practice. This would be a valuable addition which would really establish the tool as a market leader, but sadly was not possible in the context of a student project.

# 9. Conclusion

This project has represented an enormous undertaking for me as a student, and as an employee in Citibank. The learnings gained through the completion of this project will stay with me throughout the rest of my career. Probably the largest learning point relates to the area of time-management. Tackling this project on a part-time basis forced me to create a balance between my professional career at Citibank, and my academic career at Ulster University. This balance was not always struck cleanly; at times my responsibilities in Citibank had to take precedence over this project, which was always a threat to the timely delivery of the project. The experience of adapting to and mitigating this risk was an excellent learning experience for me. This project also significantly widened my network within Citibank, allowing me to connect and have meaningful conversations with people from many different teams across the bank. A further learning was to be found in the technical expertise required to implement a system with this level of complexity – the skills developed here will no doubt stand by me as I progress through my career.

This project was completed as a part of a 5-year part-time course at the university. Of all of those years, this project stands out as the true highlight and the pinnacle of my learning. At the end of this project I can look back with pride on what has been a truly enriching period of time in my student career at Ulster University.

# References

|  |  |
| --- | --- |
| [1] | Software Engineering Institue (SEI), CMMI® for Development, Version 1.3, Carnegie Mellon, 2010. |
| [2] | Isabel Lopes Margarido, Raul Moreira Vidal, Marco Vieira, “Lessons Learnt in the Implementation of CMMI Maturity Level 5,” in *2012 Eigth International Conference on the Quality of Information and Communications Technology*, 2012. |
| [3] | S. I Made Sugi Ardana, “Software Development Evaluation Process Using CMMI-Dev on Limited Resources Company,” in *2017 3rd International Conference on Science in Information Technology (ICSITech)*, 2017. |
| [4] | Nan Niu, Steve Easterbrook, “So, You Think You Know Others' Goals? A Repertory Grid Study,” IEEE, Toronto, 2007. |
| [5] | G. A. Kelly, The Psychology of Personal Constructs, Norton, 1955. |
| [6] | Enrique Diaz de Leon, Paul Guild, “Assessing Intangibles in Lean Start-Ups Using Repertory Grids,” in *PICMET'17 Technology Management for Interconnected World*, 2017. |
| [7] | Mark Heckmann, Lukas Burk, “Gridsampler – A Simulation Tool to Determine the Required Sample Size for Repertory Grid Studies,” *Journal of Open Research Software,* vol. 2, p. 2, 2017. |
| [8] | B. McFeeley, IDEAL: A User's Guide for Software Process Improvement, Pittsburgh: Carnegie Mellon, 1996. |
| [9] | ISTQB Exam Certification, “What are the Software Development Models?,” 2017. [Online]. Available: http://istqbexamcertification.com/what-are-the-software-development-models/. [Accessed October 2017]. |
| [10] | ISTQB Exam Certification, “What is Waterfall model- advantages, disadvantages and when to use it?,” 2017. [Online]. Available: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/. [Accessed October 2017]. |
| [11] | ISTQB Exam Certification, “What is Iterative model- advantages, disadvantages and when to use it?,” 2017. [Online]. Available: http://istqbexamcertification.com/what-is-iterative-model-advantages-disadvantages-and-when-to-use-it/. [Accessed October 2017]. |
| [12] | ISTQB Exam Certification, “What is Spiral model- advantages, disadvantages and when to use it?,” 2017. [Online]. Available: http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/. [Accessed October 2017]. |
| [13] | E. D. Wilie, “The Spiral Model,” 2014. [Online]. Available: http://www.the-software-experts.com/e\_dta-sw-process-model-spiral.php. [Accessed October 2017]. |
| [14] | ISTQB Exam Certification, “What is Agile model – advantages, disadvantages and when to use it?,” 2017. [Online]. Available: http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/. [Accessed October 2017]. |
| [15] | D. Firesmith, “Quality Requirements Checklist,” *Journal of Object Technology,* vol. 4, no. 9, p. 1, 2005. |
| [16] | Suzanne Robertson, James Robertson, “Ten Tests for Requirements,” Volere, [Online]. Available: http://www.volere.co.uk/ten-tests-for-requirements. [Accessed 26 February 2018]. |
| [17] | S. Sehlhorst, “Prioritizing Software Requirements with Kano Analysis,” Pragmatic Marketing, 10 August 2007. [Online]. Available: https://pragmaticmarketing.com/resources/articles/prioritizing-software-requirements-with-kano-analysis. [Accessed 26 February 2018]. |
| [18] | K. E. Wiegers, “First Things First: Prioritizing Requirements,” Process Impact, [Online]. Available: http://www.processimpact.com/articles/prioritizing.html. [Accessed 26 February 2018]. |
| [19] | Google, “Google Charts,” Google, [Online]. Available: https://google-developers.appspot.com/chart/interactive/docs/gallery/piechart. [Accessed March 2018]. |
| [20] | CMMI Product Team, CMMI® for Development, Version 1.3, Carnegie Mellon, 2010. |
| [21] | Simone Stumpt, Janet McDonnell, “Using Repertory Grid to Test Data Quality and Expert's Hunches,” in *14th International Workshop on Database and Expert Systems Applications (DEXA'03)*, 2003. |
| [22] | One Stop Testing, “Iterative Model,” [Online]. Available: http://www.onestoptesting.com/sdlc-models/iterative-model.asp. [Accessed October 2017]. |
| [23] | Tutorials Point, “https://www.tutorialspoint.com/mvc\_framework/mvc\_framework\_introduction.htm,” Tutorials Point, [Online]. Available: https://www.tutorialspoint.com/mvc\_framework/mvc\_framework\_introduction.htm. [Accessed March 2018]. |

Appendices

**Appendix A – Glossary of Terms**

|  |  |
| --- | --- |
| **Appraisal** | “An examination of one or more processes by a trained team of professionals using an appraisal reference model as the basis for determining, at a minimum, strengths and weaknesses.” [20] |
| **Capability Level** | “Achievement of process improvement within an individual process area.” (See also “generic goal,” “specific goal,” “maturity level,” and “process area.”) [20] |
| **Configuration Management** | “A discipline applying technical and administrative direction and surveillance to (1) identify and document the functional and physical characteristics of a configuration item, (2) control changes to those characteristics, (3) record and report change processing and implementation status, and (4) verify compliance with specified requirements.” [20] |
| **Continuous Representation** | “A capability maturity model structure wherein capability levels provide a recommended order for approaching process improvement within each specified process area.” (See also “capability level,” “process area,” and “staged representation.”) [20] |
| **Expected CMMI Components** | “CMMI components that describe the activities that are important in achieving a required CMMI component.” [20] |
| **Generic Goal** | “A required model component that describes characteristics that must be present to institutionalize processes that implement a process area.” [20] |
| **Generic Practice** | “An expected model component that is considered important in achieving the associated generic goal.  The generic practices associated with a generic goal describe the activities that are expected to result in achievement of the generic goal and contribute to the institutionalization of the processes associated with a process area.” [20] |
| **Informative CMMI Components** | “CMMI components that help model users understand the required and expected components of a model.  These components can be examples, detailed explanations, or other helpful information. Subpractices, notes, references, goal titles, practice titles, sources, example work products, and generic practice elaborations are informative model components.” [20] |
| **Institutionalisation** | “The ingrained way of doing business that an organization follows routinely as part of its corporate culture.” [20] |
| **Managed Process** | “A performed process that is planned and executed in accordance with policy; employs skilled people having adequate resources to produce controlled outputs; involves relevant stakeholders; is monitored, controlled, and reviewed; and is evaluated for adherence to its process description.” (See also “performed process.”) [20] |
| **Maturity Level** | “Degree of process improvement across a predefined set of process areas in which all goals in the set are attained.” (See also “capability level” and “process area.”) [20] |
| **Performed Process** | “A process that accomplishes the needed work to produce work products; the specific goals of the process area are satisfied.” [20] |
| **Process** | “A set of interrelated activities, which transform inputs into outputs, to achieve a given purpose.” [20] |
| **Process Area** | “A cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making improvement in that area.” [20] |
| **Process Capability** | “The range of expected results that can be achieved by following a process.” [20] |
| **Process Improvement** | “A program of activities designed to improve the process performance and maturity of the organization’s processes, and the results of such a program.” [20] |
| **Required CMMI Components** | “CMMI components that are essential to achieving process improvement in a given process area.  Specific goals and generic goals are required model components. Goal satisfaction is used in appraisals as the basis for deciding whether a process area has been satisfied.” [20] |
| **Requirements Management** | “The management of all requirements received by or generated by the project or work group, including both technical and nontechnical requirements as well as those requirements levied on the project or work group by the organization.” [20] |
| **Specific Goal** | “A required model component that describes the unique characteristics that must be present to satisfy the process area.” [20] |
| **Specific Practice** | “An expected model component that is considered important in achieving the associated specific goal.  The specific practices describe the activities expected to result in achievement of the specific goals of a process area.” [20] |
| **Staged Representation** | “A model structure wherein attaining the goals of a set of process areas establishes a maturity level; each level builds a foundation for subsequent levels.” (See also “maturity level” and “process area.”) [20] |
| **Sub-practice** | “An informative model component that provides guidance for interpreting and implementing specific or generic practices.  Subpractices may be worded as if prescriptive, but they are actually meant only to provide ideas that can be useful for process improvement.” [20] |

**Appendix B – Listing of Volere Shells**

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 1 |  | **Event/Use Case #: 1** | |
| **Description:** The system shall have a registration facility | | | |
| **Rationale:** To allow the system administrator to create an account, so that users may log into and use the system. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** A form shall collect the user’s data, including a name, email address and a password. The password should be hashed before storage in the database to secure the data. Given that the appraisal data will be confidential, the system administrator will be the only person able to register users. | | | |
| **Customer Satisfaction:**1 | **Customer Dissatisfaction:**5 | |  |
| **Dependencies:** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 2 |  | **Event/Use Case #:  1** | |
| **Description:** The system shall have a log-in facility | | | |
| **Rationale:** To allow pre-registered users to log into the system, and prevent non-approved users from gaining access to the system. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** A form will allow registered users to enter their email and password to gain access to the system. | | | |
| **Customer Satisfaction:**1 | **Customer Dissatisfaction:**5 | |  |
| **Dependencies:** #1 | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 3 |  | **Event/Use Case #:   1** | |
| **Description:** The system administrator shall be able to assign layered permissions to users on the system. | | | |
| **Rationale:** To allow the creation and management of user accounts for limited purposes. For example, some users will be able to conduct a CMMI appraisal and view the dashboard, while other users will only have access to the results dashboard. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled in the form of a ‘user settings’ screen that can only be accessed by the administrator. | | | |
| **Customer Satisfaction:**1 | **Customer Dissatisfaction:**2 | |  |
| **Dependencies:** #1 | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 4 |  | **Event/Use Case #:  1, 2** | |
| **Description:** The system shall allow users to conduct the CMMI appraisal by filling in a questionnaire. | | | |
| **Rationale:** This is the primary purpose of the system | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled in the form of a questionnaire that will present questions based upon a repertory grid style of questioning | | | |
| **Customer Satisfaction:  3** | **Customer Dissatisfaction:  5** | |  |
| **Dependencies: #8** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 5 |  | **Event/Use Case #:   2** | |
| **Description:** The system shall produce a strengths and weaknesses document upon the completion of the questionnaire | | | |
| **Rationale:** To allow the user to view areas for improvement in relation to the appraised process areas | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled in the form of a printable page generated by the software that will | | | |
| **Customer Satisfaction:  3** | **Customer Dissatisfaction:  5** | |  |
| **Dependencies: #4, #6** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 6 |  | **Event/Use Case #:   2** | |
| **Description:** The system shall use a bespoke algorithmic scoring mechanism to establish the extent to which a process is enacted | | | |
| **Rationale:** To allow the establishment of capability levels in each process area, and to assign an overall capability level. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled by designing an algorithm which will feed off responses indicated in the repertory grid questions. This algorithm shall be incorporated into the questionnaire process and display the final score once the questionnaire is complete. The number shall then be stored in the database. | | | |
| **Customer Satisfaction:  4** | **Customer Dissatisfaction:  4** | |  |
| **Dependencies: #4, #8** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 7 |  | **Event/Use Case #: 2** | |
| **Description:** The system shall be capable of displaying a gap analysis style of chart once the appraisal is complete. | | | |
| **Rationale:** To allow the user to visually see the capability level of each assessed process area, and how far it can yet be improved upon. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled by using the bespoke algorithm to first calculate the capability in each process area, and then representing this data in a horizontal bar chart, with the process areas as the Y axis, and the capability level as the X axis. This chart shall be presented to the user upon completion of the questionnaire. | | | |
| **Customer Satisfaction:  4** | **Customer Dissatisfaction:  2** | |  |
| **Dependencies: #4, #6** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 8 |  | **Event/Use Case #:  2** | |
| **Description:** The system shall include a back-end database capable of storing appraisal information | | | |
| **Rationale:** To allow the user to store CMMI appraisals and query the database to analyse a wider set of results. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled by incorporating a mySQL database into the website which will allow queries to be executed upon the appraisal data. | | | |
| **Customer Satisfaction:  4** | **Customer Dissatisfaction:  2** | |  |
| **Dependencies: #13** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 9 |  | **Event/Use Case #:  3** | |
| **Description:** The system shall include a visual dashboard feature | | | |
| **Rationale:** To allow the user to visually analyze the wider set of CMMI appraisal results. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled by incorporating dashboard section into the website, the dashboard shall be powered by feeding queries from the mySQL database into the Google charts API. | | | |
| **Customer Satisfaction:  5** | **Customer Dissatisfaction:  2** | |  |
| **Dependencies: #4, #6, #8** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 10 |  | **Event/Use Case #: 3** | |
| **Description:** The system shall be capable of displaying predesigned queries from the database as “key metrics”. | | | |
| **Rationale:** To allow the creation of KPIs for the appraisals. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled by writing queries into the website’s PHP to draw out relevant metrics from the database. For example, “Mean Appraisal Score” could be used as a key performance indicator. | | | |
| **Customer Satisfaction:  3** | **Customer Dissatisfaction:  1** | |  |
| **Dependencies: #8** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

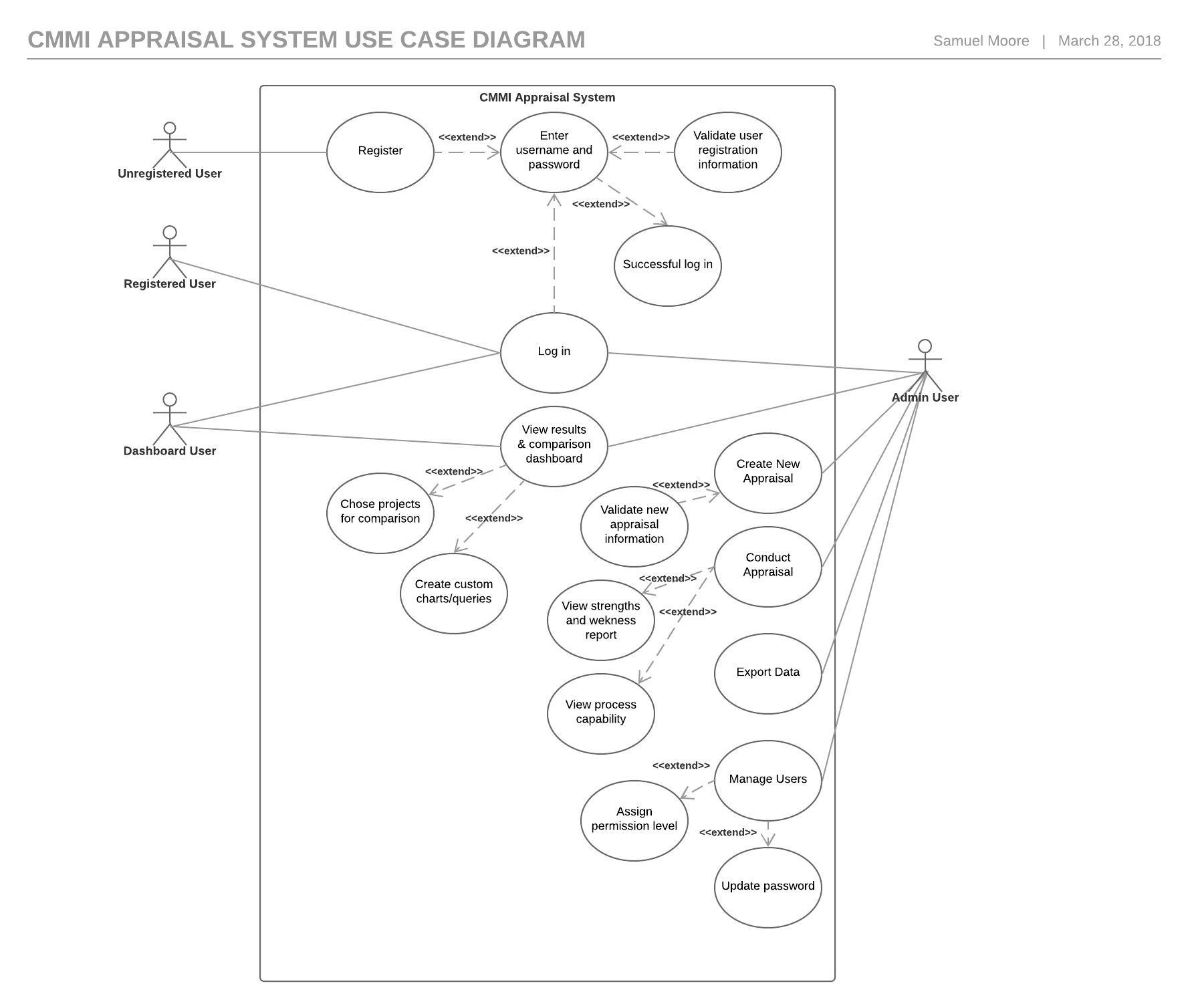
|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 11 |  | **Event/Use Case #:   3** | |
| **Description:** The system shall feature a sand-box style “custom query builder” | | | |
| **Rationale:** To allow users to design and execute their own queries against the database. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled by including a page on the website which will allow the user to select various options in order to design their own query to be ran against the database. | | | |
| **Customer Satisfaction:  5** | **Customer Dissatisfaction:  1** | |  |
| **Dependencies: #8** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 12 |  | **Event/Use Case #:    3** | |
| **Description:** The system shall feature a sand-box style “custom graph builder” | | | |
| **Rationale:** To allow users to design and view their own graphs using data from the database. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled by including a page on the website which will allow the user to select various options in order to design their own graph to analyze appraisal results | | | |
| **Customer Satisfaction:  5** | **Customer Dissatisfaction:  1** | |  |
| **Dependencies: #8** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

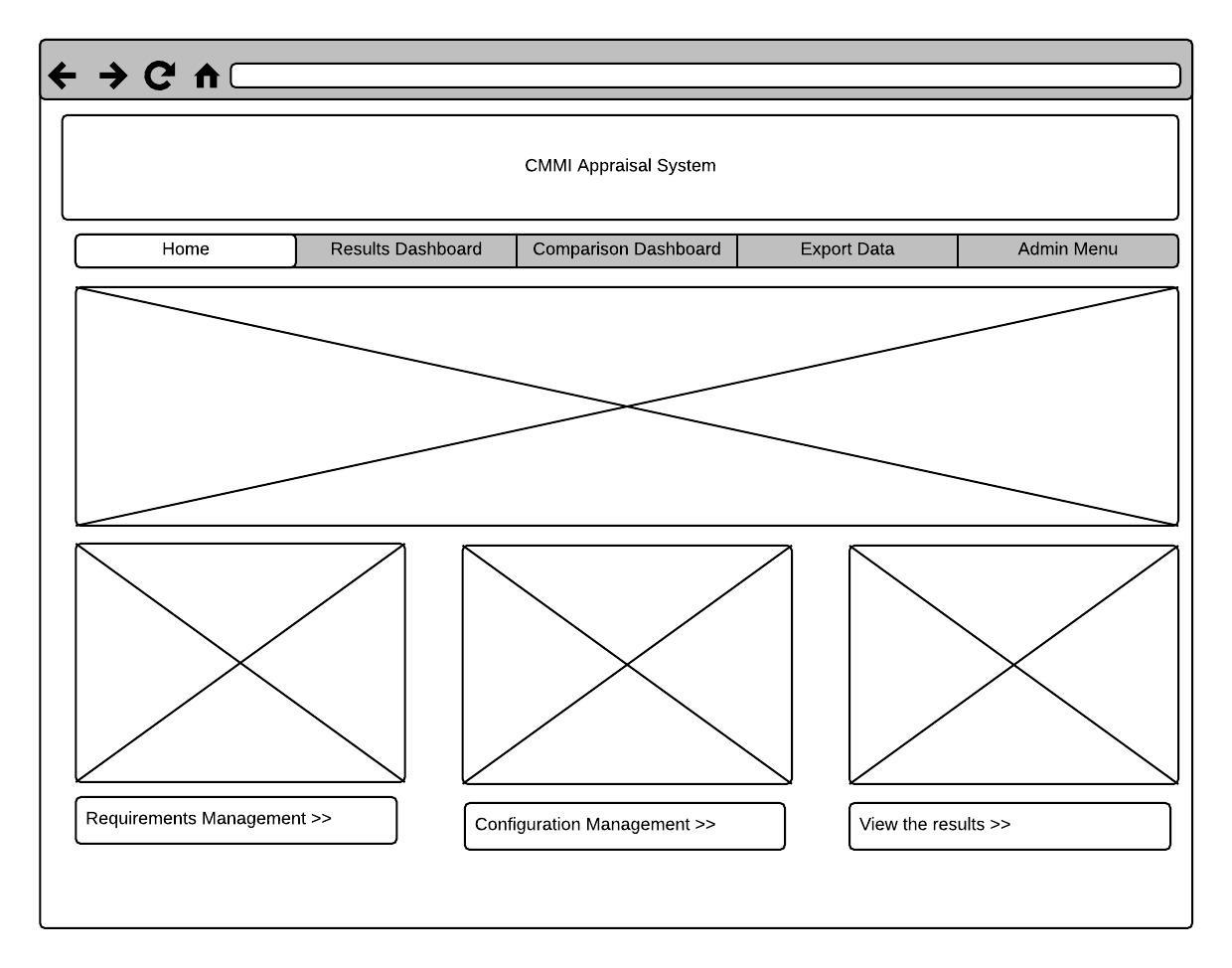
|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 13 |  | **Event/Use Case #:  n/a** | |
| **Description:** The system shall be web-hosted | | | |
| **Rationale:** To allow users to access the system online, thus allowing the integration of the online database and Google charts API | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled by the website being hosted on Ulster Universities web server and the user being able to access the software by navigating to the site’s URL in their web browser. | | | |
| **Customer Satisfaction:  1** | **Customer Dissatisfaction:  5** | |  |
| **Dependencies:** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement #:** 14 |  | **Event/Use Case #: n/a** | |
| **Description:** The system shall be mobile and tablet compatible | | | |
| **Rationale:** To allow preserve the aesthetics and function of the system should the user choose to access the tool using a phone or tablet. | | | |
| **Source:** Samuel Moore – Developer | | | |
| **Fit Criterion:** This requirement shall be fulfilled by the use of HTML and CSS to design a mobile-specific menu that is triggered once a tablet-or-smaller screen size is detected. | | | |
| **Customer Satisfaction:  3** | **Customer Dissatisfaction:  4** | |  |
| **Dependencies: #13** | | | |
| **Supporting Materials:** | | | |
| **History:** Created November 5th 2017 | | | |

**Appendix C – Use Case Diagram**

****

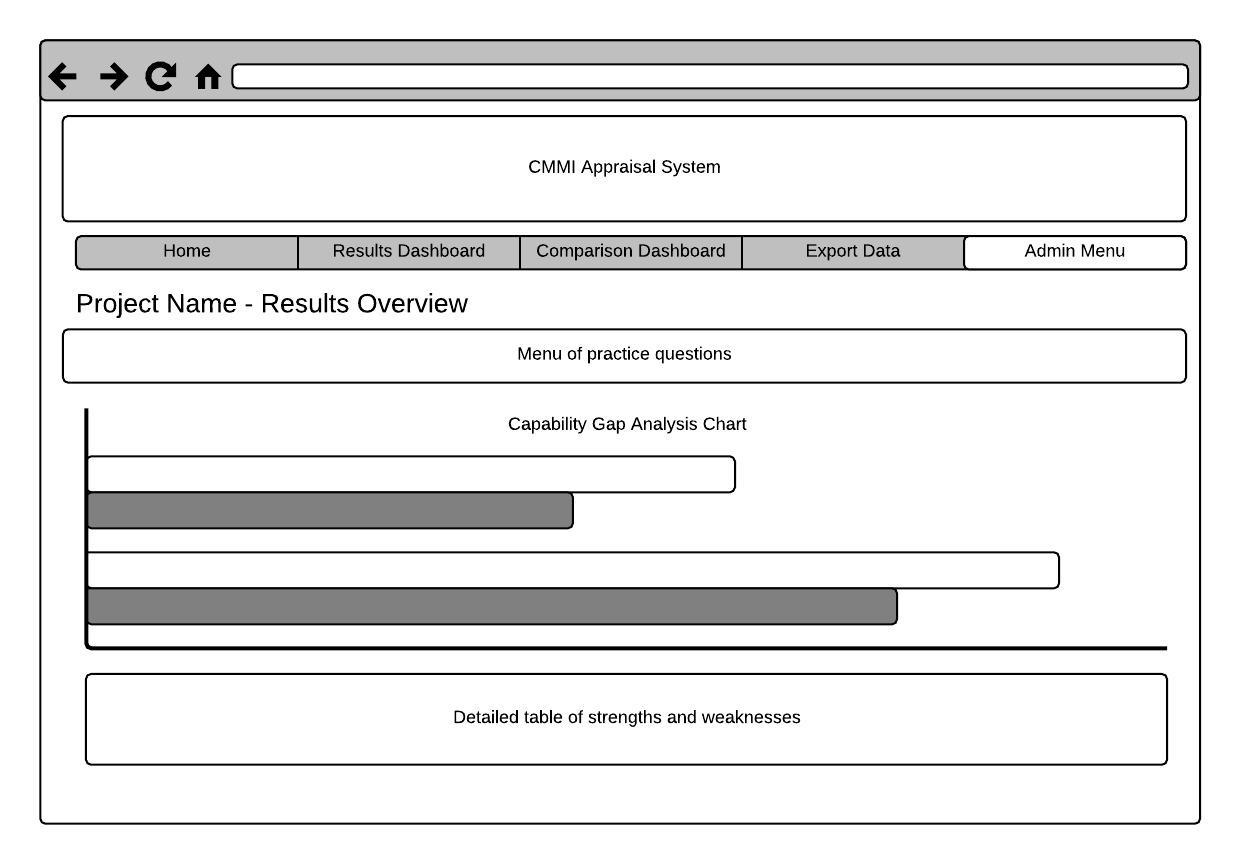
**Appendix D1 – Home Page Wireframe**

****

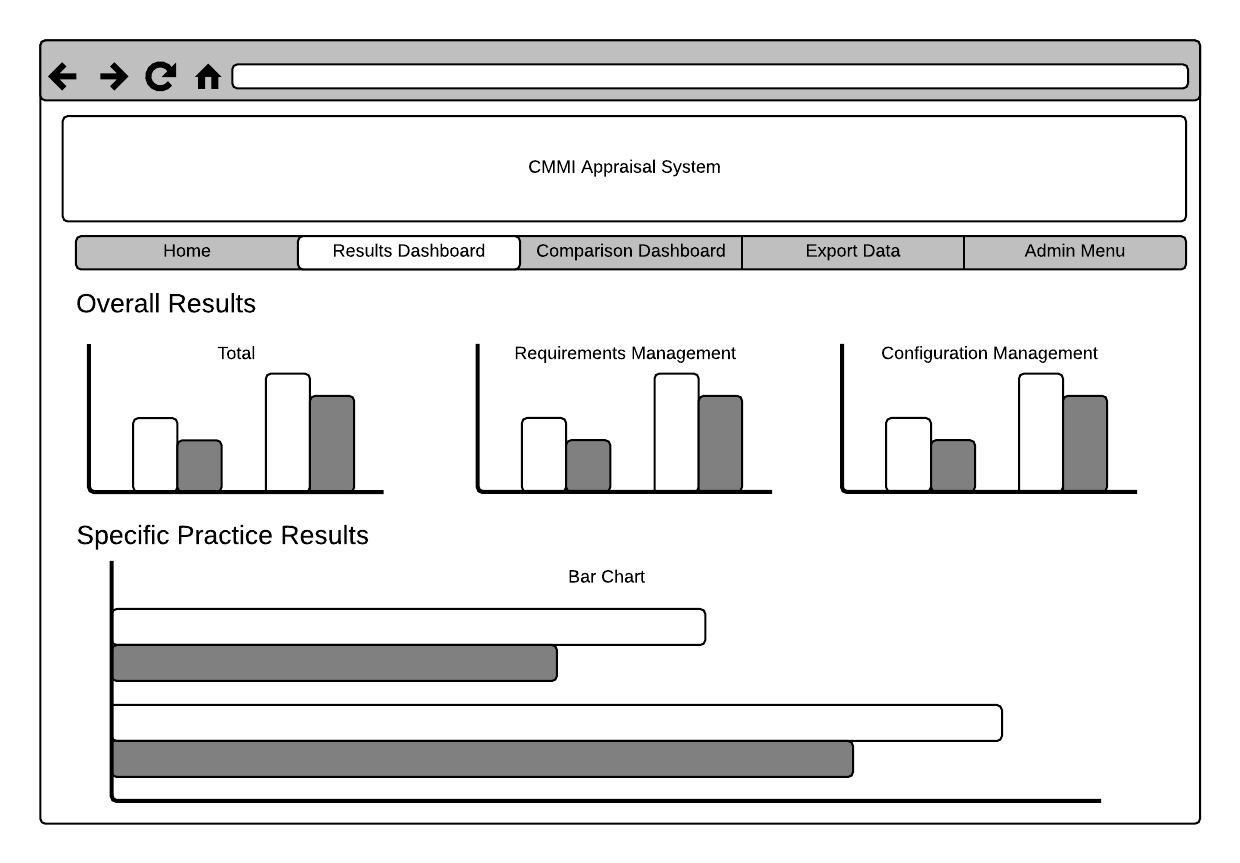
**Appendix D2 – Conduct Appraisal**

****

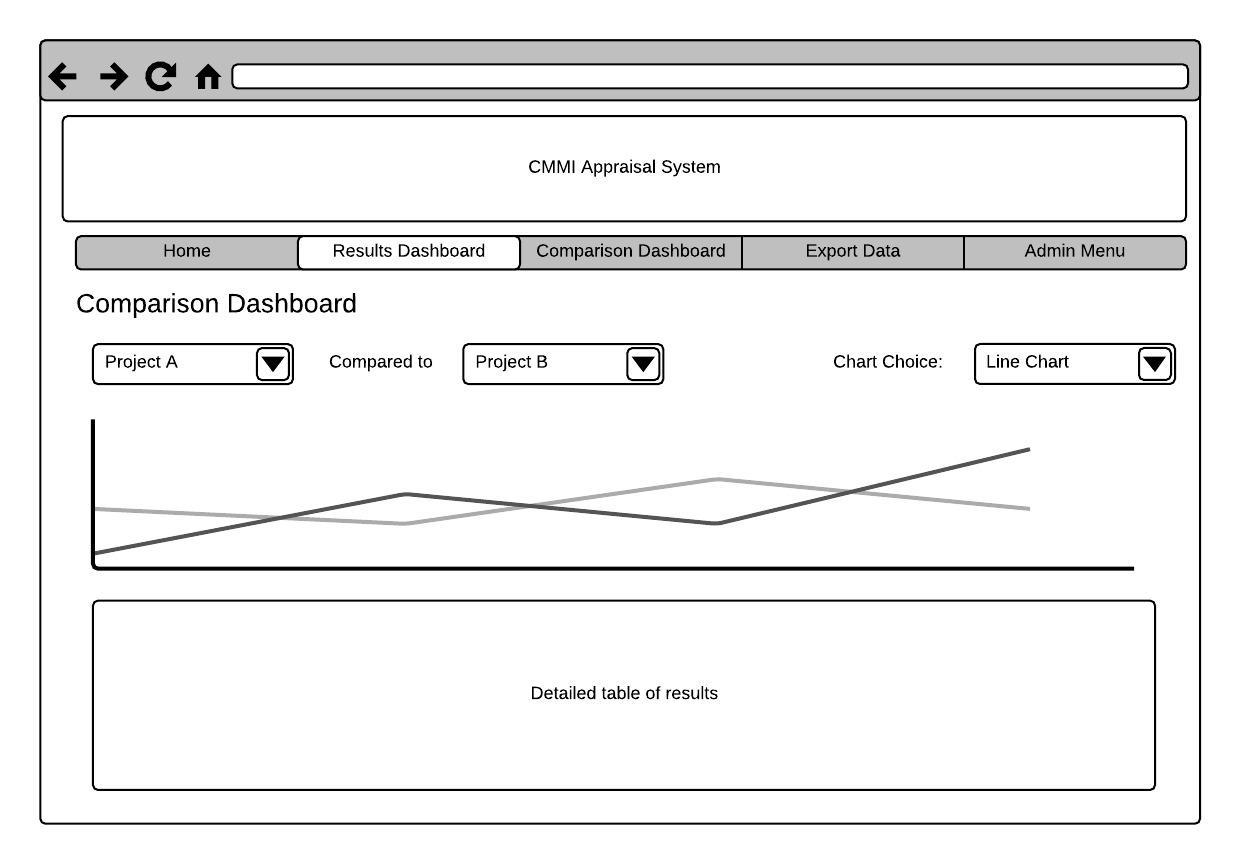
**Appendix D3 – Appraisal Results**

****

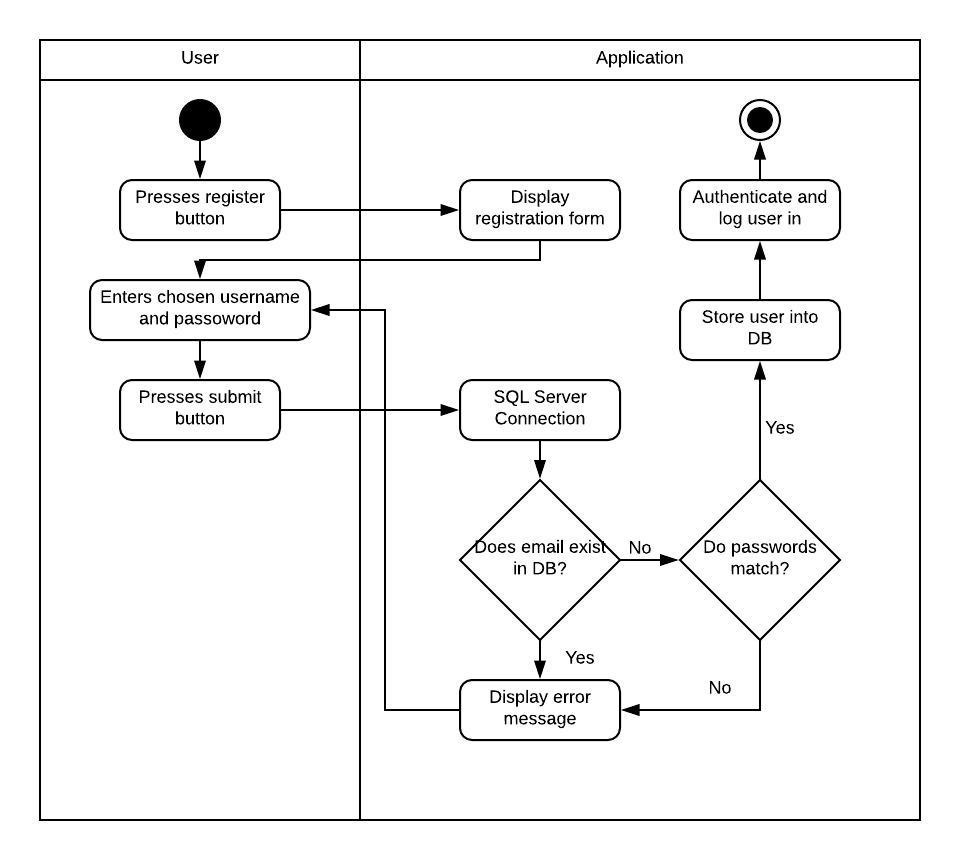
**Appendix D4 – Results Dashboard**

****

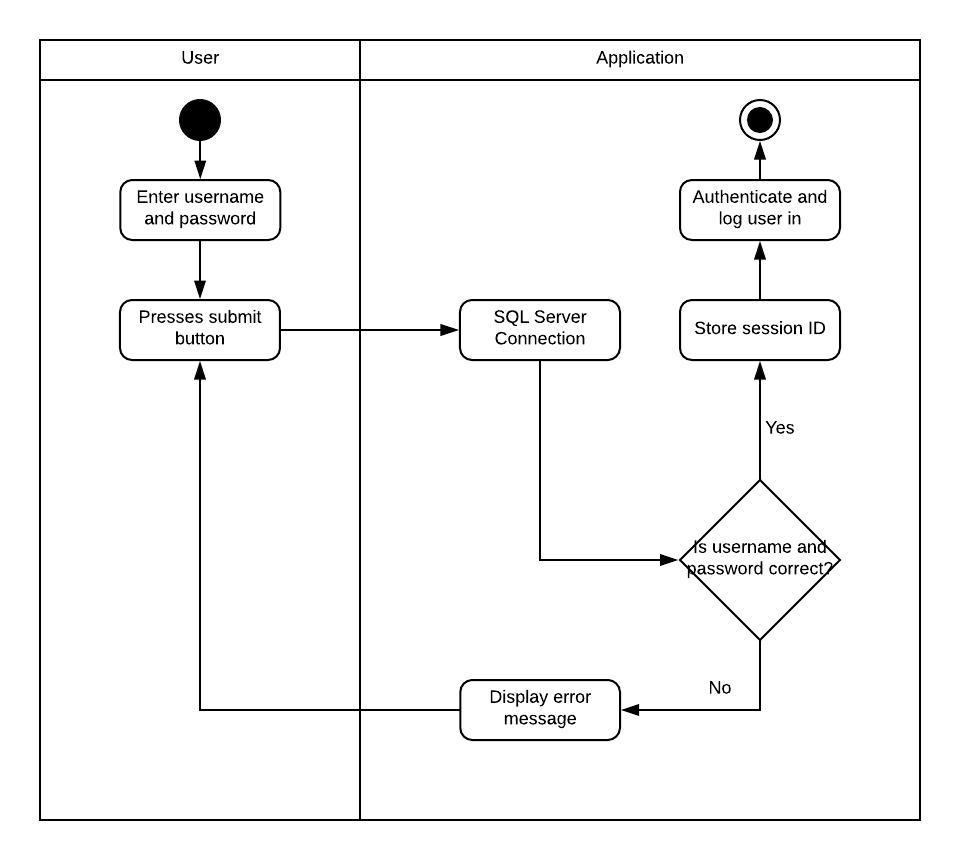
**Appendix D5 – Comparison Dashboard**

****

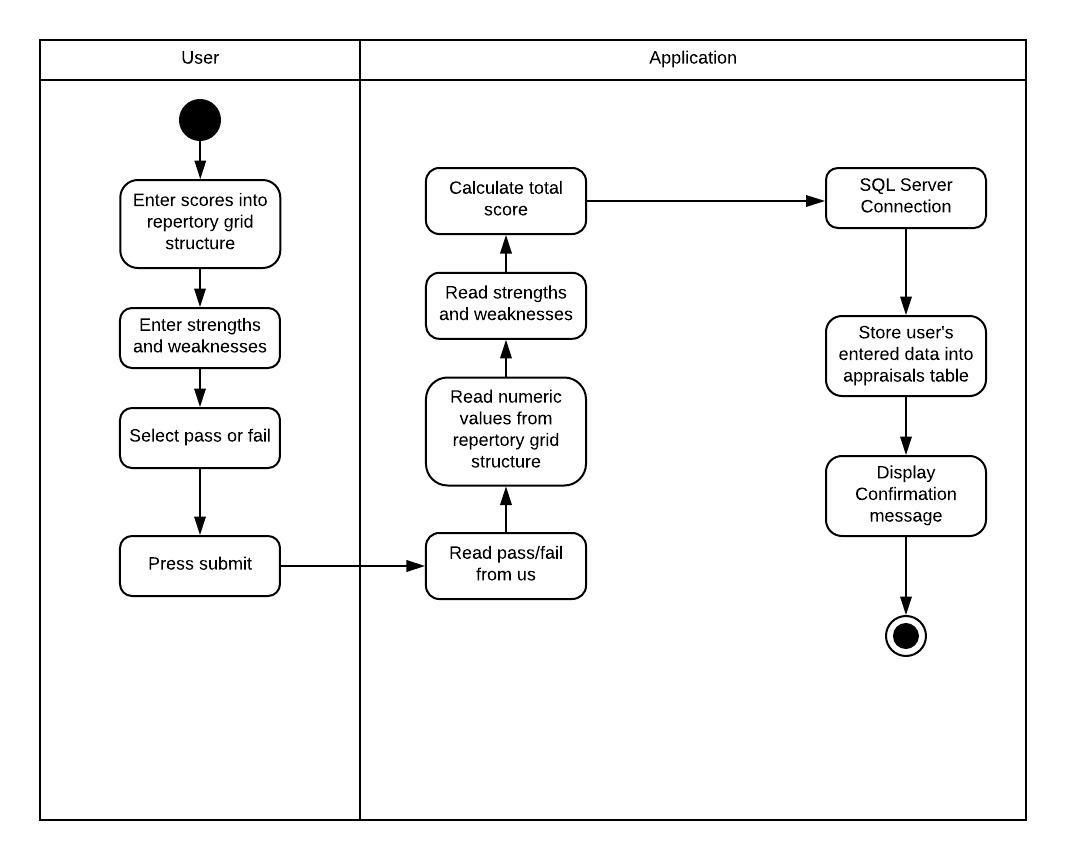
**Appendix E1 – Register User**

****

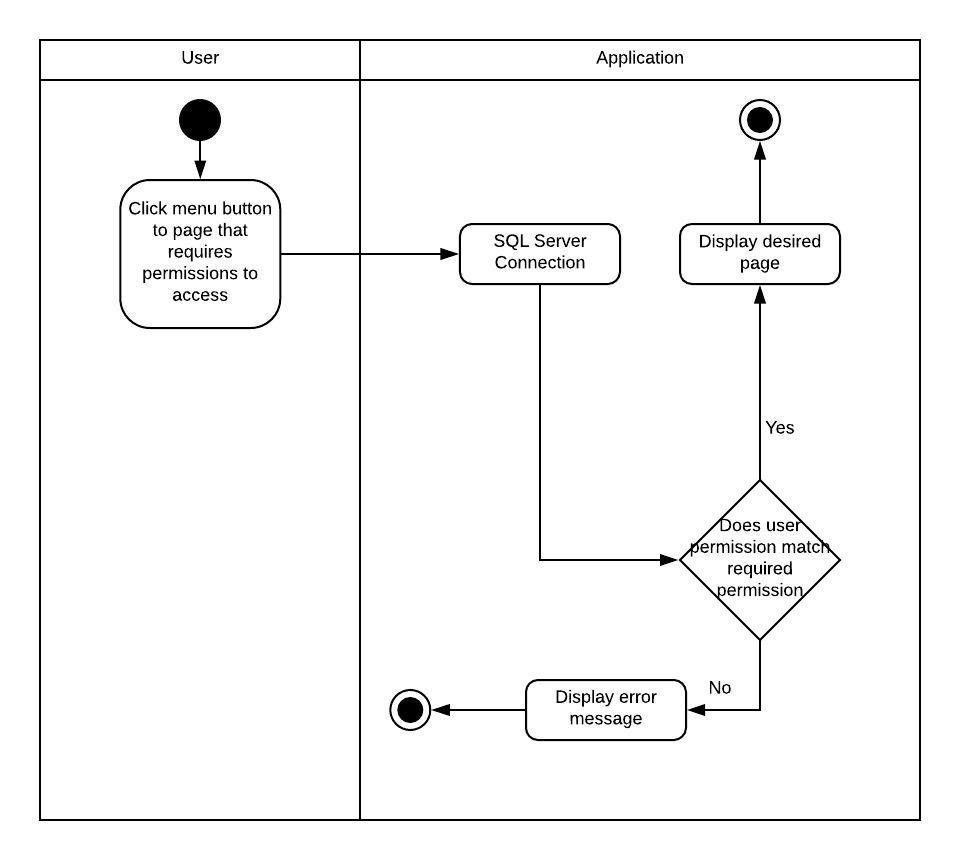
**Appendix E2 – Log in**

****

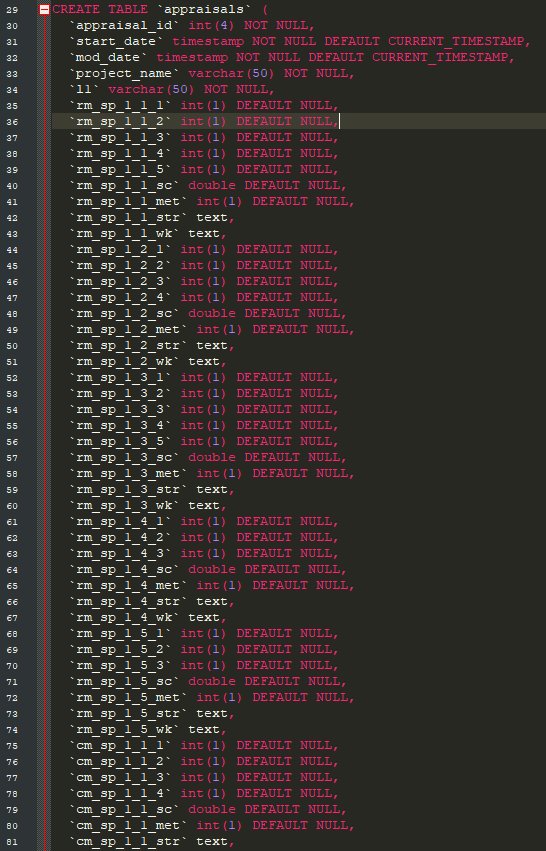
**Appendix E3 – Store data**

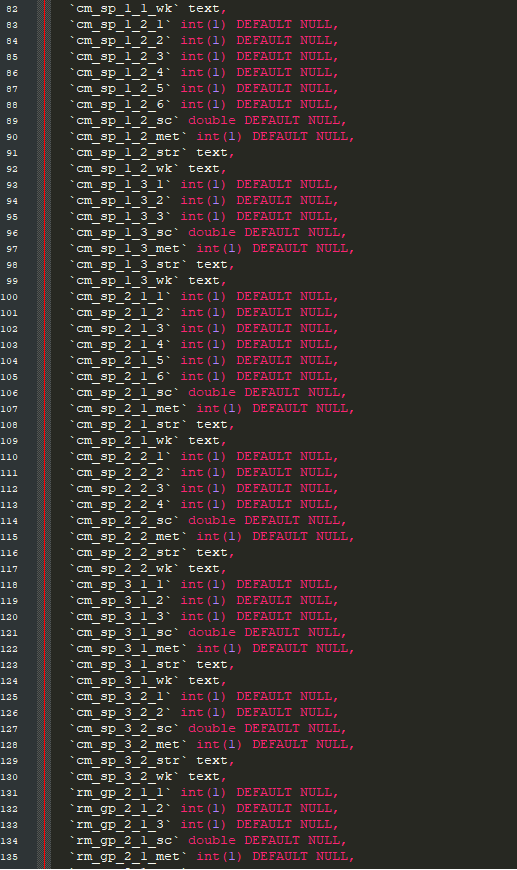
****

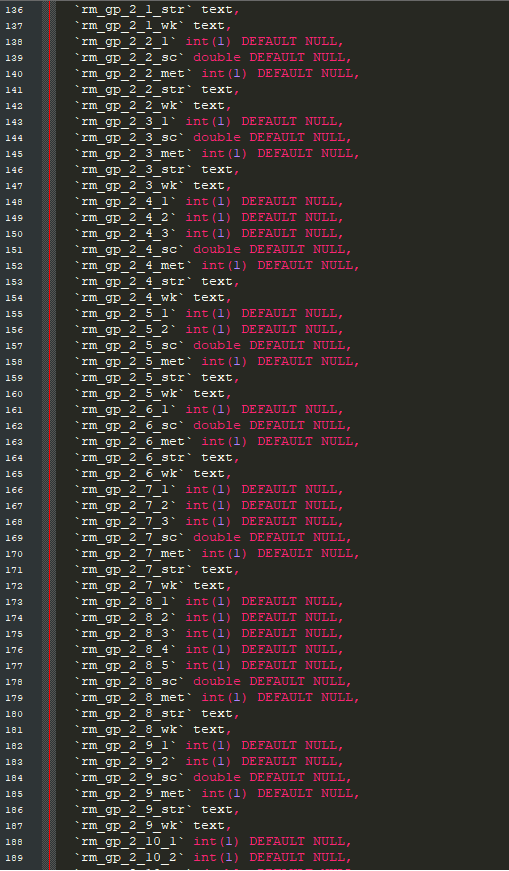
**Appendix E4 – Permissioned Access**

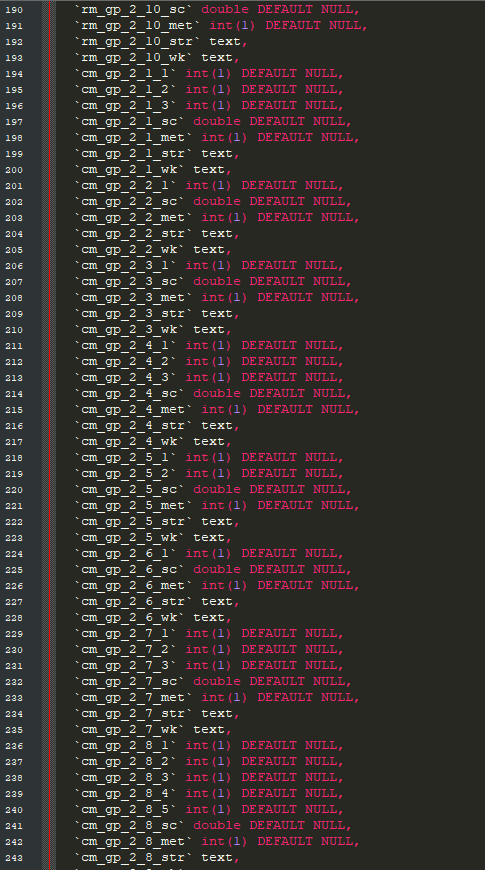
****

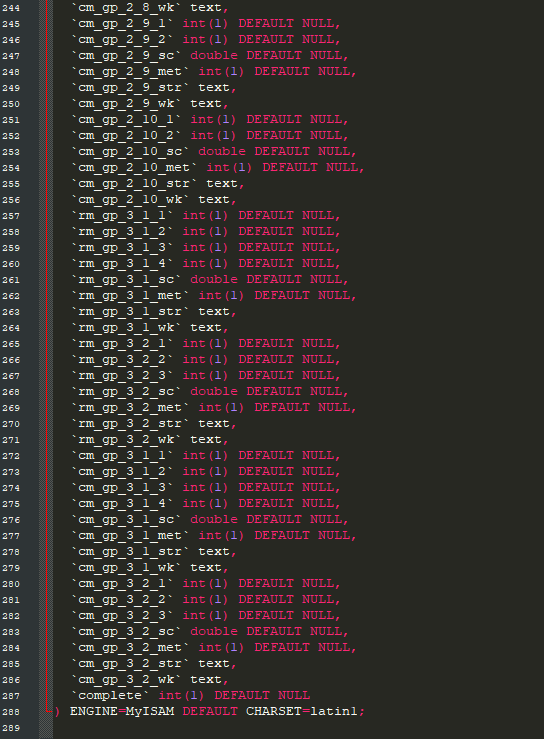
**Appendix F1 – Code to create appraisal table in SQL Database**



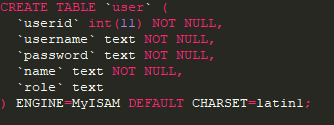




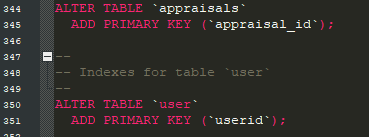




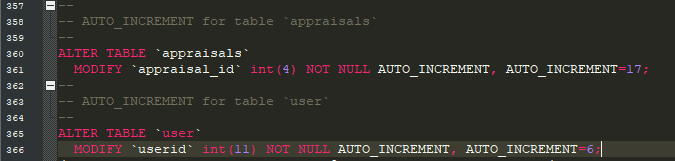
**Appendix F2 – SQL Code to create User table**



**Appendix F3 – Adding primary keys to SQL tables**



**Appendix F4 – Adding auto increment to primary keys**



**Appendix G – Unit Test Plan**

| **#** | **Test** | **Data** | **Expected Result** | **Actual Result** | **Pass / Fail** |
| --- | --- | --- | --- | --- | --- |
| 1 | Register new user successfully | Username: sm66768, Password: “password”, match both cases | User registered successfully | As expected | Pass |
| 2 | Register new user, invalid data | Username: sm66768, Password: “password”, match both cases | Error message should state that user already exists in database, user should not be registered | As expected | Pass |
| 3 | Register new user, invalid data | Username: sm66770, Password: “password”, does not match both cases | Error message should state that passwords do not match, user should not be registered | As expected | Pass |
| 4 | Log in, valid data | Username: sm66769, password: password | Log in successfully, be taken to home page | As expected | Pass |
| 5 | Log in, invalid data | Username: sm66769, password: 1234 | Error message presented, user not logged in | As expected | Pass |
| 6 | Responsive test, all pages | N/A | Page can be viewed successfully and fully on mobile device and laptop | As expected | Pass |
| 7 | Begin appraisal link, home page | Click begin appraisal link on home page | User is taken to begin appraisal page | As expected | Pass |
| 8 | Requirements management link, home page | Click requirements management link on home page | User is taken to external website relating to RM | As expected | Pass |
| 9 | Configuration Management link, home page | Click configuration management link on home page | User is taken to external website relating to configuration management | As expected | Pass |
| 10 | Results link, home page | User clicks on results link on home page | User is taken to results dashboard | As expected | Pass |
| 11 | Admin menu, logged in as admin | Logged in as admin | The admin menu shows in the nav bar when user is logged in with admin permissions | As expected | Pass |
| 12 | Admin menu, not logged in as admin | Logged in as user | The admin menu does not appear in the nav bar when user is not logged in with admin permissions | As expected | Pass |
| 13 | Create New Appraisal | Project Tesla, ICG | New appraisal is created | As expected | Pass |
| 13 | Create new Appraisal, with existing name in DB | Test Project, ICG | Appraisal is not created, error message shown | As expected | Pass |
| 14 | Click edit on manage appraisals | Click edit button beside project on manage appraisals | User is taken to that appraisal’s data entry screen | As expected | Pass |
| 15 | Manage users, edit button | Click edit button associated with user | User is taken to manage user page for that user | As expected | Pass |
| 16 | Manage users, change password | Change password for user to “123”, then log that user in successfully | Password is changed and user can be logged in | As expected | Pass |
| 17 | Manage users, change permission level to results | Change permission level to “results” for a given user | Permission level is changed to results | As expected | Pass |
| 18 | Manage users, change permission level to admin | Change permission level to “admin” for a given user | Permission level is changed to admin | As expected | Pass |
| 19 | Save appraisal data for RM\_SP1\_1 (then repeat for 35 other practices) | 1-5 responses for rep grid, text entered in strength and weaknesses, pass/fail entered | Data is saved successfully in all cases | As expected | Pass |
| 20 | Check current score for all specific practices | Check current score against responses given in rep grid, numbers between 1-5 | Data matches in all cases | As expected | Pass |
| 21 | Results Dashboard | Check that chart numbers showing in all charts match the numbers stored in the database | Data matches in all cases | As expected | Pass |
| 22 | Complete a new appraisal, check it appears in dashboard | Complete new appraisal, new appraisal should appear in results dashboard | New appraisal appears in dashboard | As expected | Pass |
| 23 | Comparison dashboard, colour coding | Check colour coding in all rows for Test Project | Colour coding should appear correctly | As expected | Pass |
| 24 | Comparison dashboard, switch chart function | Fill in form and select chart type | Chart should change based on the users selections on the form | As expected | Pass |
| 25 | Access results dashboard, correct permission | Access the results dashboard from a user account with results or admin access | Access successful | As expected | Pass |
| 26 | Access results dashboard, insufficient permission | Access the results dashboard from a user account with insufficient permission level | Access not granted | As expected | Pass |
| 27 | Access comparison dashboard, correct permission | Access comparison dashboard from a user account with results or admin access | Access granted | As expected | Pass |
| 28 | Access comparison dashboard, insufficient permission | Access comparison dashboard from a user account with insufficient permission | Access not granted | As expected | Pass |