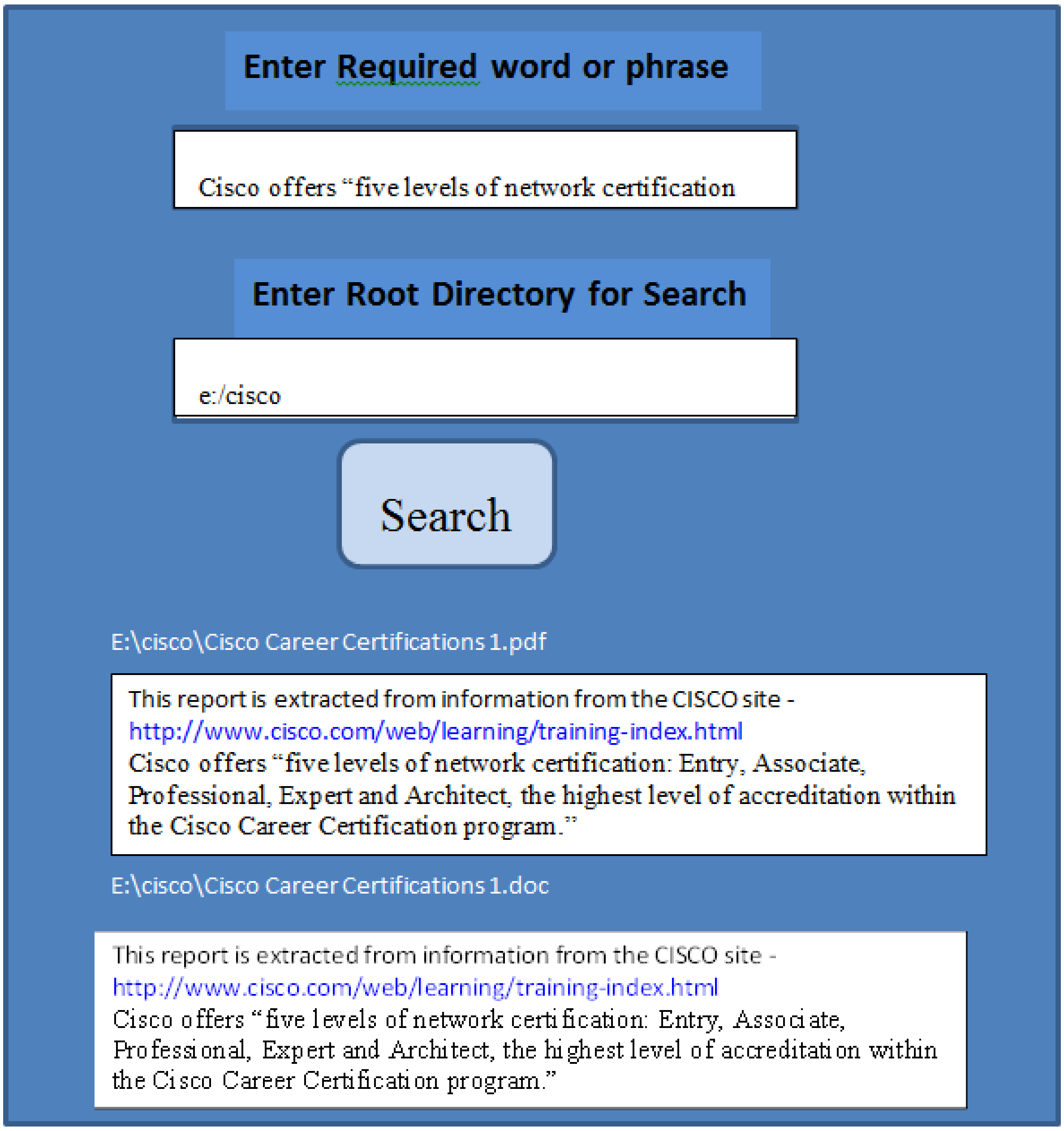
Aardvark Pty. Ltd. Often needs to retrieve documents on its hard drive based on those documents containing words or phrases. It thus needs a system that allows the user to enter the required phrase or word and to then find the document/s containing that phrase and to print them to screen. The type of documents that need to be accessed are

1. Text
2. Word (.doc or .docx)
3. Pdf

To shorten the search path, the user would also enter the root directory where the search begins. An example of the Ruby Rails implementation is shown below


Enter Required word or phrase
Cisco offers "five levels of network certification
Enter Root Directory for Search
e:/cisco
Search

E:\cisco\Cisco Career Certifications 1.pdf

This report is extracted from information from the CISCO site - http://www.cisco.com/web/learning/training-index.html
Cisco offers "five levels of network certification: Entry, Associate, Professional, Expert and Architect, the highest level of accreditation within the Cisco Career Certification program."

E:\cisco\Cisco Career Certifications 1.doc

This report is extracted from information from the CISCO site - http://www.cisco.com/web/learning/training-index.html
Cisco offers "five levels of network certification: Entry, Associate, Professional, Expert and Architect, the highest level of accreditation within the Cisco Career Certification program."

The project is to be delivered in several phases

Phase 1 (5 marks due week 7)

In phase an initial prototype is required. This prototype will

1. Request a term
2. A directory

It will then search for text documents containing the term/word in only the specified directory. No user interface is required and it can be implemented entirely in Ruby without rails.

Phase 2 (5 marks due week 8)

This phase extends phase 1 by adding the capacity to search word and pdf documents as well as text documents

Phase 3 (10 marks due week 9)

This phase extends phase 2 by extending the search to subdirectories. That is, given a starting directory, the application will search the entire tree from the starting directory for the required word/term. Hint- this will require the use of recursion (we will cover this later)

Phase 4 (15 marks due week 11)

This prototype will then be transported to a rails framework with a user interface similar to the example below. Here for every match found, the filename including the full path will be printed to the browser and the text of the document will be printed into a text-area (these will be added dynamically according to the number of matches found).

As well, the implementation will connect to a database which will give the user an option to store/retrieve the search term and filenames and paths. Thus, if a term has been used and stored, the search can be considerably shortened.

Marking Scheme for Phases 1,2 and 3

Marking Scheme	Approx % of grade	Excellent	Good	Reasonable	Poor
Program specifications/ Correctness	52%	No errors, program always works correctly and meets the specification(s).	Minor details of the program specification are violated, program functions incorrectly for some inputs.	Significant details of the specification are violated, program often exhibits incorrect behaviour.	Program only functions correctly in very limited cases or not at all.
Documentation, style	16%	code is well-commented and styling and naming conventions are well implemented	comments are generally used when required and naming and styling conventions are usually followed	some comments are missing or naming and styling conventions are not always followed	Hardly any comments or adherence to naming and styling conventions
Design Principles and efficiency	16%	The program always follows good design principles such as loose coupling and high cohesion and is efficient	The program usually follows good design principles such as loose coupling and high cohesion and is generally efficient	The program sometimes follows good design principles such as loose coupling and high cohesion. There are some flaws in its efficiency .	The program is badly designed and inefficient
Robustness	16%	The program catches all exceptions that can occur and provides all necessary information to the user when they occur	The program catches most exceptions that can occur and provides reasonable information to the user when they occur	The program catches some exceptions that can occur and provides some information to the user when they occur	The program does not catch exceptions

Marking Scheme for Phase 4

Marking Scheme	Approx % of grade	Excellent	Good	Reasonable	Poor
Program specifications/ Correctness	52%	No errors, program always works correctly and meets the specification(s).	Minor details of the program specification are violated, program functions incorrectly for some inputs.	Significant details of the specification are violated, program often exhibits incorrect behaviour.	Program only functions correctly in very limited cases or not at all.
Documentation, style	12%	code is well-commented and styling and naming conventions are well implemented	comments are generally used when required and naming and styling conventions are usually followed	some comments are missing or naming and styling conventions are not always followed	Hardly any comments or adherence to naming and styling conventions
Design Principles and efficiency	12%	The program always follows good design principles such as loose coupling and high cohesion and is efficient	The program usually follows good design principles such as loose coupling and high cohesion and is generally efficient	The program sometimes follows good design principles such as loose coupling and high cohesion. There are some flaws in its efficiency .	The program is badly designed and inefficient
Robustness	12%	The program catches all exceptions that can occur and provides all necessary information to the user when they occur	The program catches most exceptions that can occur and provides reasonable information to the user when they occur	The program catches some exceptions that can occur and provides some information to the user when they occur	The program does not catch exceptions
User Interface	12%	Interface is neat and well organized and uses rails capabilities extensively	Interface is basically neat and well organized but does not make full use of rails capabilities	Interface is reasonable and makes some use of rails capabilities	Interface is untidy and not organized and does not use rails capabilities