1. **Outcome****(b)**: An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
2. **Interpretation**: This outcome states that students who have gone through the program should have the ability to analyze the requirements of a computing product. The result of the requirement analysis should be documented following the IEEE software engineering standards.
3. **Student Work Sampled**: The requirement analysis documents submitted in CMSC 4513 – Software Design and Development are sampled. The requirement analysis documents should be completed based on the following template, which is a simplifed version of the IEEE software engineering standards (a detailed version of the template is given at the end of this document).

**Table of contents**

1. Revision history

2. Introduction

2.1. Purpose and scope

2.2. Definitions, acronyms, and abbreviations

2.3. References

2.4. Overview

3. Overall description

3.1. Product functions

3.2. User characteristics

3.3. Constraints

3.4. Assumptions and dependencies

4. Specific requirements

4.1. External interface requirements

4.1.1. User interfaces

4.1.2. Hardware interfaces

4.1.3. Software interfaces

4.1.4. Communication interfaces

4.2. Functional requirements

4.3. Performance requirements

4.4. Design constraints

4.5. Software system attributes

1. **Evaluation Procedure**: Student work will be evaluated by at least two faculty members, one of whom must be the instructor of CMSC 4513 – Software Design and Development. The evaluation instrument is given at the end of the document.

The evaluation instrument is designed based on three categories of competency as reflected in the requirement analysis document.

The evaluation instrument contains a list of yes-no questions corresponding to the above three categories of competency. The instrument is used for every requirement analysis document. The percentage of yes answers to every question is evaluated using the rubric given in Section 5. Questions receving Unsatisfactory or Developing under the rubric are used to indentify opportunities for improvement based on the category of competency that the questions reflect.

1. **Evaluation Metrics**: Consider the rubric given where is the percentage of yes answers to a particular question. If fewer than 25% of the requirement analysis documents receive a yes answer to a particular question, then the result is unsatisfactory. A close examination of the question and its corresponding competency category is needed to determine if some improvement can be made. To a lesser degree, if between 25% and 50% of the requirement analysis documents receive a yes answer to a particular question, the question and its competency category need to be investigated. If more than 50% of the requirement analysis documents receive a yes answer to a particular question, the rubric may be updated to reflect higher expectations or a conclusion can be drawn that the goal corresponding to that particular question has been achieved.

|  |  |  |  |
| --- | --- | --- | --- |
| **Unsatisfactory** | **Developing** | **Satisfactory** | **Excellent** |
|  |  |  |  |

1. **Course Cross Reference**: The courses that cover the topics relavant to this outcome are CMSC 3303 System Analysis and Design, CMSC 4283 Software Engineering, and CMSC 4513 Software Design and Development.

**Template of the Requirement Analysis Document**

1. Revision history

*<Show the time, version and who write the sections or subsections of this document.>*

2. Introduction

2.1. Purpose and scope

*<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the system that is covered by this product (what does it do and what does it not do), particularly if this document describes only part of the system or a single subsystem.>*

2.2. Definitions, acronyms, and abbreviations

*<Describe definitions of all the terms, acronyms and abbreviations used in the document. Special attention should be paid to the clarification of terms and concepts from the domain of application.>*

2.3. References

*<List any other documents or Web addresses to which this document refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>*

2.4. Overview

*<Contain an outline of the remainder of the document.>*

3. Overall description

3.1. Product functions

*<Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Details will be provided in the next section, so only a high level summary is needed here. Organize the functions to make them understandable to any reader of the document. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a use case diagram, is often effective.>*

3.2. User characteristics

*<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.>*

3.3. Constraints

*<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>*

3.4. Assumptions and dependencies

*<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>*

4. Specific requirements

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

4.1. External Interface Requirements

4.1.1. User Interfaces

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

4.1.2. Hardware Interfaces

*<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>*

4.1.3. Software Interfaces

*<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>*

4.1.4. Communication Interfaces

*<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>*

4.2. Functional requirements

*<Describe how the transformation of inputs to outputs is achieved. The description is given for each class of functions, and sometimes for each individual function. To a certain extent, this description can be seen as a solution to the user. This component of the requirement specification is the main starting point for the design phase.>*

4.3. Performance requirements

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

4.4. Design constraints

*<Design constraints may result from such things as the prescribed use of certain standards or hardware.>*

4.5. Software system attributes

*<Particular attention is paid to quality aspects, such as availability, security, and maitainability. These requirements must be measurable and verifiable. They must be stated in objective terms.>*

**Assessment Instrument for Outcome (b)**

* **Outcome(b)**: An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
* **Student Work:** Requirement analysis documents completed by students in CMSC 4513 Software Design and Development.
* **Evaluation Questions:**
  1. Section 3.2 (User characteristics) documents various classes of users who will use the system. Yes No
  2. Is the scope of the product (what does it do and what does it not do) correctly specified in Section 2.1? Yes No
  3. Are the logical characteristics of each interface between the product and the user correctly described in Section 4.1.1? *These may include sample screen images, GUI standards or product style guidelines, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, and error message display standards.* Yes No
  4. Are the logical and physical characteristics of each interface between the software and the hardware components of the system correctly described in Section 4.1.2? *These may include the supported device types, the nature of the data and control interactions between the software and the hardware components, or the communication protocols to be used.* Yes No
  5. Are the connections between this product and other specific software components (name and version) correctly described in Section 4.1.3? *Other software components include databases, operating systems, tools, libraries, and integrated commercial components. The description should identify the data items or messages sent between the product and the other components and their purpose.* Yes No
  6. Are communication functions or protocols required by the sytem correctly described in Section 4.1.4? *Communication functions include e-mail, Web browser, network server communications protocols, and electronic forms. Communication prototols include standards such as FTP or HTTP.* Yes No
  7. Are functional requirements correctly described in Section 4.2? *Functional requirements describe how system inputs are transformed into outputs. This part is the basis for the design phase. For each specific funtional requirement, the document should discuss the purpose of the function, the inputs, processing, and outputs.*
  8. Are performance requirements correctly described in Section 4.3? *Performance requirements help the developers understand the intent and make suitable design choices. Such requirements need to be as specific as possible. Performance requirements for individual functional requirements or features need to be specified. Performance requirements can be static or dynamic requirements. Static requirements concern the number of terminals to be connected or the number of users that can be handled concurrently. Dynamic requirements concern the operational performance of the system, such as how often certain functions are called and how fast should the system’s reaction be.* Yes No
  9. Are the funtional and performance requirements specified in testable/measurable terms? Yes No
  10. Are only external system behaviors (black box) specified? *No low-level implementation details should appear in the document.* Yes No