# Milestone 2: Requirements Analysis Phase

**Objective:** In the Requirements Analysis Phase the requirements gathered in the planning phase are used to address components such as report definitions and layouts, screen definitions and layouts, data element definitions, workflow diagrams, and security matrices. The Requirements Document (aka Requirements Specification) like the Project Proposal plan and for consistency should contain a cover page, containing at a minimum a project title, author, course number, instructor name, document revision number, and date along with a 3-5 sentence abstract that provides an overview of the project and introduces the reader to the document.

**Deliverable:** Submission of the [Projects Requirements Document](#_Projects_Requirements_Document)

Refer to the information below for additional details on each section of the Projects Requirements Document. A Project Requirements template can be found in the class resources.

**Special Note:** Depending on the project, portions of this milestone may not be applicable. In cases like these, work with the instructor to determine the components of the project that may replace this step. The instructor will determine what documentation is needed and how this documentation will be assessed.

Use Cases/User Stories Functional Requirements

A use case describes a sequence of actions a project performs thatroject s an a system technoligies ative text of tuse cases tells the store of how a user and the system interaacte in aa simple,shows provide an observable result of a value to a particular actor. The narrative text of a use case tells the story of how a user and the project interact in a simple, structured, and complete way. If necessary, the use cases can be documented in an external Excel spreadsheet with a reference to the document provided in the Requirements Document.

Use cases may be provided as diagrams or text narratives using agile scrum user story format. A generic example follows:

Diagram

Description automatically generated

Use Cases/User Stories Non-Functional Requirements

A use cases should also be defined for non-functional requirements (NFR’s). A list common NFR’s can be found at <https://en.wikipedia.org/wiki/Non-functional_requirement> and this should be reviewed with the appropriate NFR’s applicable to the project defined as use cases or as a narrative using user stories. If necessary, the use cases can be documented in an external Excel spreadsheet with a reference to the document provided in the Requirements Document.

System Design (Top-Down Approach)

The top-down design approach begins with a listing of the primary functions of the system. For example, if the software has three primary functions, then each function is treated as a major task. The major tasks are then analyzed and further decomposed into subtasks as needed.

Use the top-down approach in the analysis of the system requirements. This will then be used in the design of the system. Represent the top-down design with a diagram similar to a flow chart. A generic example follows for a system that calculates a utility bill:

There are many more examples online. Try to use a verb as the first word in the task description.

Technical Requirements

Provide a listing of the technical requirements that the system must fulfill. List these requirements and include a brief description. Follow the listing of technical requirements with a listing and explanation of the software and/or hardware that is necessary to meet the requirement.

System Logical Model

Provide a diagram of the logical architecture of the system. This diagram should reflect the business rules of the application, if applicable, and illustrate the flow of information through the system. The logical design should be abstract; that is, it should not include any implementation details. The purpose of the logical model is to map data from its source to its appropriate destination, following the system decision structure. The logical model describes how the system works, in terms of function and logical information. An example of a logical architecture diagram of the utility bill system follows:

Diagram

Description automatically generated

Reports

Provide a listing of the reports the system will provide, if applicable. For each report, include details such as:

1. Title and purpose of each report
2. Data included in each report
3. Formatting requirements
4. Frequency

Example:

|  |  |  |
| --- | --- | --- |
| <Report Title> | | |
| Display Value | Variable or Field | Format Notes |
| Customer ID | Customer.ID | "xx-xxxxxxx" |
| Customer Plan | Customer.planID | Premier, Standard or Basic |
| Customer Data Usage | Customer.dataMB | Number, 2 decimal places |
| Customer Bill | Customer.amtDue | Number, 2 decimal places |

Depending on the project, if a list of reports in not applicable, consider providing documentation similar to the following examples:

1. An analysis of an algorithm and proof of its computational complexity
2. A visualization of a process

Screen Definitions and Layouts

Provide a draft of each user interface screen required by the system. These interface "sketches" will be refined in the design phase. Diagrams should include components such as, but not limited to:

1. Title
2. Toolbars
3. Date entry fields
4. Content, such as text, charts, tables, etc.

Example  
Table

Description automatically generated

Depending on the project, if screen definitions and layouts are not applicable, work with the instructor to define the components of the project that may replace this step. The project may be theoretical, practical, software development, or a mobile application. Flowcharts, mathematical proofs, and technical diagrams are suitable substitutes for project that do not involve creating complete GUI-based applications.

Security

Discuss the security issues relevant to the system. If there are no security issues, explain why. Provide a matrix that illustrates the permissions/access rights of different types of users on the system.

Helpful Hint: If there are no immediately evident security issues, it does not mean there are not any. An often-overlooked security issue is badly designed software that is exploited by hackers to penetrate a system even if the hacking objective is a different component of that system.

Ensure that you are properly addressing the security issues or the absence of security issues. Overall, explain the nature of security issues that may impact the project or why there will be no such impact.

Example:

Graphical user interface, application, table

Description automatically generated

Projects Requirements Document Submission

Look back on what has been done so far in the project and make changes or modify what was done in the past. It is important to continually review the project plan to address issues and oversights that are discovered.