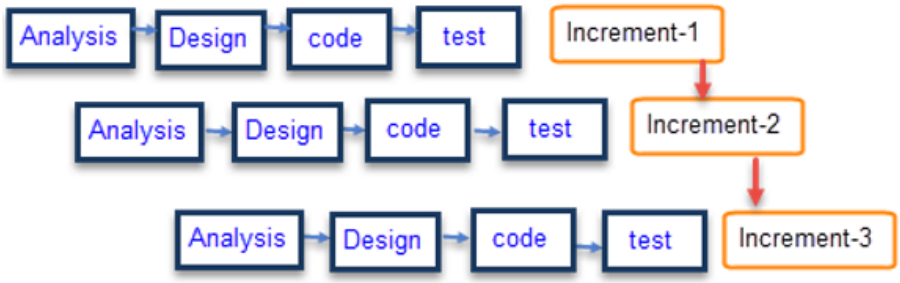
Workshop- Software specification

Software specification is the process of establishing what services are required and the constraints on the system's operation and development.

In this workshop, you and your team will execute tasks to produce the initial documentation for your system (final assignment). **The documents you will produce must be stored in the GitHub repository of your project by the end of the workshop.**

Part 1 – Using the Incremental Model

For this part of the workshop, assume that you and your team are going to use the Incremental Model to develop your software project.
(*) Please bear in mind that you will use Agil and Scrum in your project, not Incremental Model.



Incremental Model

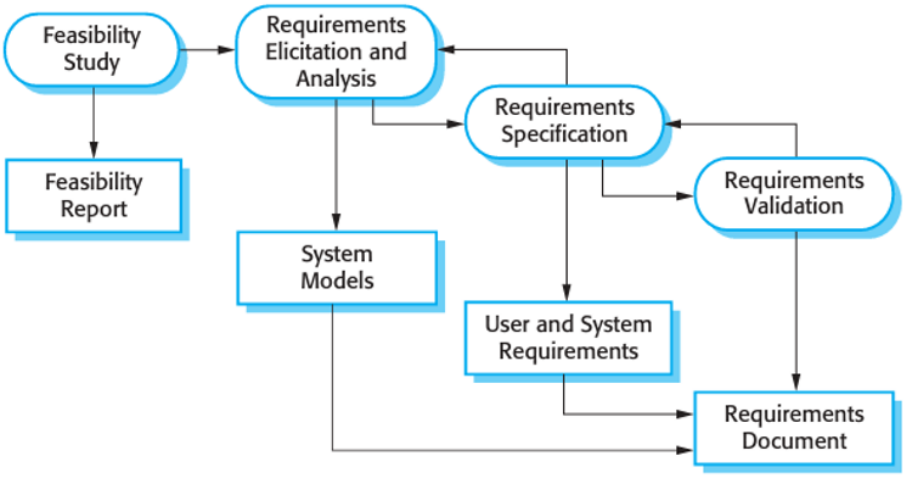
Task 1: How would you break down your project in increments? List the increments.

1	
2	
3	
4	
5	
6	
7	

Task 2: How would your plan your project's timeline?

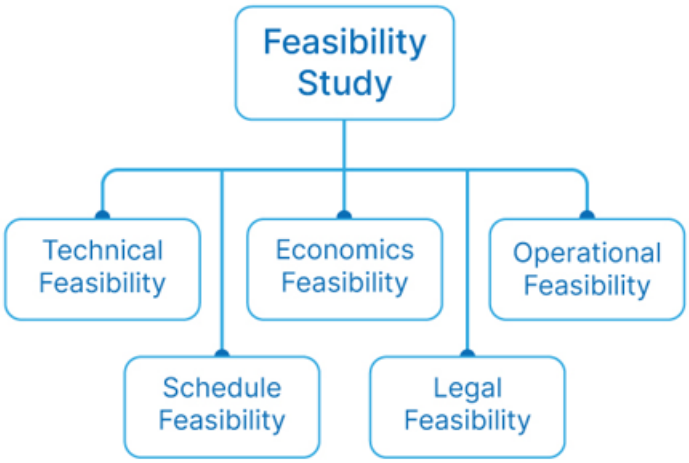
FEB				MAR				APR				MAY				JUN			
1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4

Part 2 – Software Specification



Task 1: Feasibility Study to produce a Feasibility Report (or Study)

Write a document containing 5 sections: Technical Feasibility, Schedule Feasibility, Economics Feasibility, Legal Feasibility, Operational Feasibility.



Technical feasibility

Technical Feasibility analyzes/evaluates current resources for hardware, software, and technology needed to develop the project. This technical feasibility study provides information on whether the appropriate resources and technology required for use in project development are in place. In addition, the feasibility study also analyzes the technical strength and capabilities of the technical team, whether existing technology can be used, and whether the selected technology is easy to maintain and upgrade.

- Analyze the technical skills and capabilities of software development team members.
- Determine if the relevant technology is stable and established.
- Assess that the technologies chosen for software development will have many users so that they can be consulted if they encounter problems or need improvement.

Operational Feasibility

Operational feasibility analyzes the level of service delivery according to requirements and the ease of operating and maintaining the product after deployment. Along with these other operational areas, it determines the product's usability, whether the software development team's decisions for the proposed solution are acceptable, and so on.

- Determine if the expected issue in the user request is a high priority.
- Determine if the organization is satisfied with alternative solutions proposed by the software development team.
- Determine if the solution proposed by the software development team is acceptable.
- Analyze whether users are comfortable with new software.

Economic feasibility

Project costs and benefits are analyzed in a profitability study. This means that as part of this feasibility study, a detailed analysis of the costs of the development project will be made. This includes all costs necessary for the final development, such as hardware and software resources required, design and development costs, operating costs, etc. It is then analyzed whether the project is financially beneficial to the organization.

- The costs incurred in software development generate long-term benefits for an organization.
- Costs required to conduct a complete software study (e.g., requirements extraction and requirements analysis).
- Hardware, software, development team, and training costs.

Legal Feasibility

In a legal feasibility study, the project is analyzed from the view of legality. This includes analysis of obstacles in the legal implementation of the project, data protection or social media laws, project certificates, licenses, copyrights, etc. Overall, a legal feasibility study is a study to determine whether a proposed project meets legal and ethical requirements.

Schedule Feasibility

A schedule feasibility study mainly analyzes the proposed project deadlines/deadlines, including the time it will take the team to complete the final project. This has a significant impact on the organization as the project's purpose may fail if it is not completed on time.

Step 2: Requirements Specification to produce a User and System Requirements

Fill in the Software requirement specification (SRS) document template provided on Blackboard. Here are more details of what to write in each section of the document:

1. Introduction

An SRS introduction is exactly what you expect—it's a 10,000-foot view of the overall project. When writing your introduction, describe the purpose of the product, the intended audience, and how the audience will use it. In your introduction, make sure to include:

- Product scope: The scope should relate to the overall business goals of the product, which is especially important if multiple teams or contractors will have access to the document. List the benefits, objectives, and goals intended for the product.
- Product value: Why is your product important? How will it help your intended audience? What function will it serve, or what problem will it solve? Ask yourself how your audience will find value in the product.
- Intended audience: Describe your ideal audience. They will dictate the look and feel of your product and how you market it.
- Intended use: Imagine how your audience will use your product. List the functions you provide and all the possible ways your audience can use your product depending on their role. It's also good practice to include use cases to illustrate your vision.
- Definitions and acronyms: Every industry or business has its own unique acronyms or jargon. Lay out the definitions of the terms you are using in your SRS to ensure all parties understand what you're trying to say.
- Table of contents: A thorough SRS document will likely be very long. Include a table of contents to help all participants find exactly what they're looking for.

Make sure your introduction is clear and concise. Remember that your introduction will be your guide to the rest of the SRS outline, and you want it to be interpreted the same by everyone using the doc.

2. System requirements and functional requirements

Once you have your introduction, it's time to get more specific. Functional requirements break down system features and functions that allow your system to perform as intended.

Use your overview as a reference to check that your requirements meet the user's basic needs as you fill in the details. There are thousands of functional requirements to include depending on your product. Some of the most common are:

- If/then behaviors
- Data handling logic
- System workflows
- Transaction handling
- Administrative functions
- Regulatory and compliance needs
- Performance requirements
- Details of operations conducted for every screen

If this feels like a lot, try taking it one requirement at a time. The more detail you can include in your SRS document, the less troubleshooting you'll need to do later on.

3. External interface requirements

External interface requirements are types of functional requirements that ensure the system will communicate properly with external components, such as:

- User interfaces: The key to application usability that includes content presentation, application navigation, and user assistance, among other components.
- Hardware interfaces: The characteristics of each interface between the software and hardware components of the system, such as supported device types and communication protocols.
- Software interfaces: The connections between your product and other software components, including databases, libraries, and operating systems.
- Communication interfaces: The requirements for the communication functions your product will use, like emails or embedded forms.

Embedded systems rely on external interface requirements. You should include things like screen layouts, button functions, and a description of how your product depends on other systems.

4. Non-functional requirements (NRFs)

The final section of your SRS details non-functional requirements. While functional requirements tell a system what to do, non-functional requirements (NFRs) determine how your system will implement these features. For example, a functional requirement might tell your system to print a packing slip when a customer orders

your product. An NFR will ensure that the packing slip prints on 4”x6” white paper, the standard size for packing slips.

While a system can still work if you don’t meet NFRs, you may be putting user or stakeholder expectations at risk. These requirements keep functional requirements in check, so it still includes attributes like product affordability and ease of use.

The most common types of NFRs are called the ‘Itys’. They are:

- Security: What’s needed to ensure any sensitive information your software collects from users is protected.
- Capacity: Your product’s current and future storage needs, including a plan for how your system will scale up for increasing volume demands.
- Compatibility: The minimum hardware requirements for your software, such as support for operating systems and their versions.
- Reliability and availability: How often you expect users to be using your software and what the critical failure time is under normal usage.
- Scalability: The highest workloads under which your system will still perform as expected.
- Maintainability: How your application should use continuous integration so you can quickly deploy features and bug fixes.
- Usability: How easy it is to use the product.

Other common types of non-functional requirements include performance, regulatory, and environmental requirements.

Software requirement specification (SRS) document example

1. Introduction

Describe the purpose of the document.

→ Who is this document intended for and why? How will it be used?

1.1 Product scope

List the benefits, objectives, and goals of the product.

→ What are the overall business goals of your product?

1.2 Product value

Describe how the audience will find value in the product.

→ Why is your product important? How will it help your intended audience?

1.3 Intended audience

Write who the product is intended to serve.

→ Who is your product for?

1.4 Intended use

Describe how will the intended audience use this product.

→ What will this product be used for?

1.5 General description

Give a summary of the functions the software would perform and the features to be included.