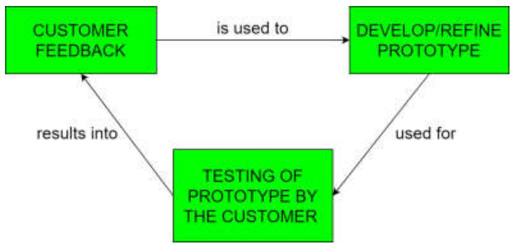
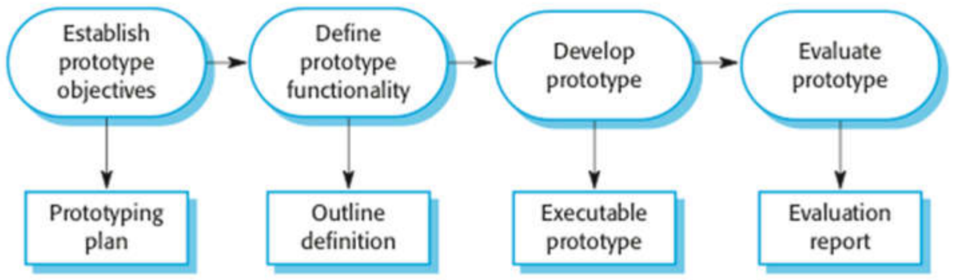
Rapid Prototyping Model

- Prototyping is defined as the process of developing a working replication of a product or system that must be engineered.
- It offers a small-scale replica of the product and is used for obtaining customer feedback as described below:





Rapid Prototyping Model (cont.)

- Rapid prototype characteristics:
  - Used in the requirements phase
  - Evaluated by the customer/user
  - Then, it is discarded -do not turn into product
- Rapid prototyping model is not proven and has its own problems
  - Possible solution
    - Rapid prototyping for defining requirements
    - Waterfall model for rest of life cycle

41

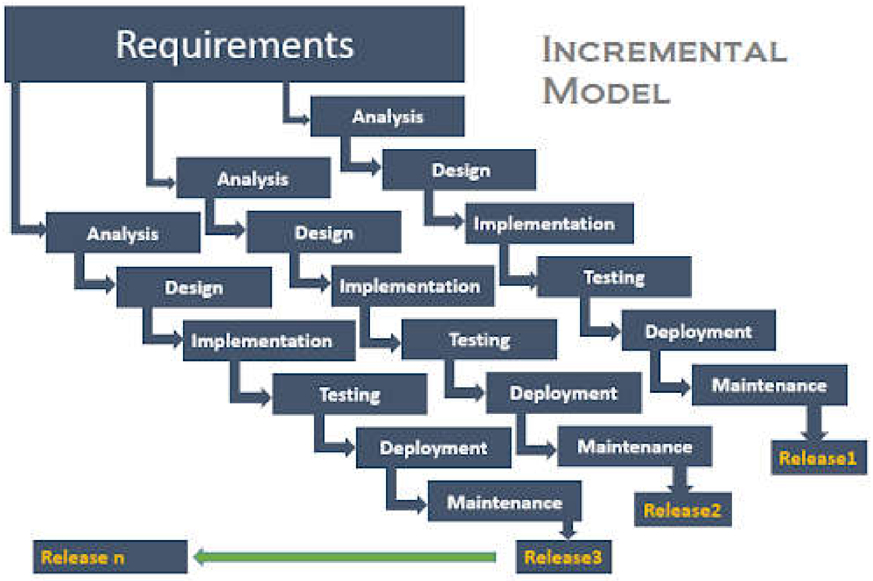
41

Incremental Model

- **Incremental Model** is a process of **software development** where requirements are broken down into multiple standalone modules of **software development** cycle.
- Each iteration passes through the requirements, design, coding and **testing** phases.
- Typical product takes from 5 to 25 builds (iterations).

42

42



43

43

Incremental Model (cont.)

- Advantages
  - The software will be generated quickly during the software life cycle
  - It is flexible and less expensive to change requirements and scope
  - Throughout the development stages changes can be done
  - This model is less costly compared to others
  - A customer can respond to each building
  - Errors are easy to be identified

44

44

Incremental Model (cont.)

- Disadvantages:
  - It requires a good planning designing
  - Problems might arise due to system architecture as not all requirements collected up front for the entire software lifecycle
  - Each iteration phase is rigid and does not overlap each other
  - Correcting a problem in one unit requires correction in all the units and consumes a lot of time

45

45

Incremental Model (cont.)

- Waterfall and rapid prototyping models
  - Deliver complete product at the end
- Incremental model
  - Deliver portion of the product at each stage

46

46

When to use Incremental models?

- Requirements of the system are clearly understood
- When **demand for an early release** of a product arises
- When software engineering **team are not very well skilled** or trained
- When high-risk features and goals are involved
- **Such methodology is more in use for web application and product-based companies**

47

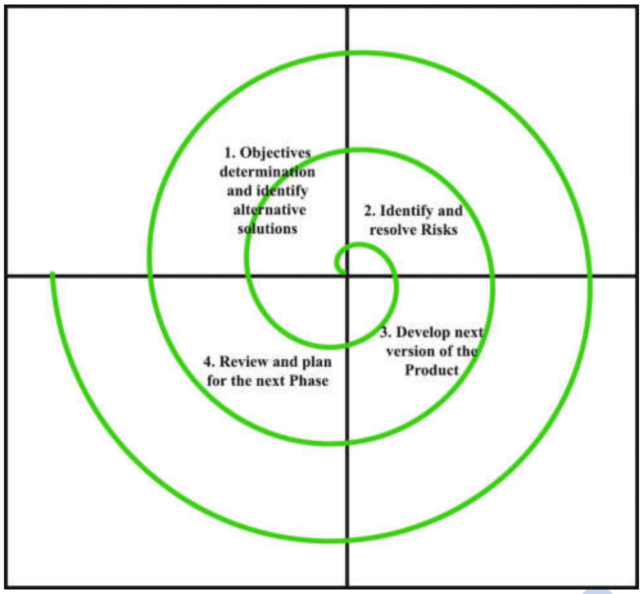
47

Spiral Model

- The **spiral model** is a risk-driven **software development** process **model**.
- Based on the unique risk patterns of a given project, the **spiral model** guides a team to adopt elements of one or more process **models**, such as incremental, waterfall, or evolutionary prototyping.
- **Risk Analysis:** Identification of potential risk is done while risk mitigation strategy is planned and finalized
- Precede each phase by
  - Alternatives
  - Risk analysis
- Follow each phase by
  - Evaluation
  - Planning of next phase

48

48


1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

49

49

When to use Spiral Methodology?

- When project is large
- When releases are required to be frequent
- When creation of a prototype is applicable
- When risk and costs evaluation is important
- For medium to high-risk projects
- When **requirements are unclear** and complex
- When **changes may require at any time**
- When long term project commitment is not feasible due to changes in economic priorities

50

50

Advantages of Spiral Model

- Additional functionality or changes can be done at a later stage
- Cost estimation becomes easy as the prototype building is done in small fragments
- Continuous or repeated development helps in risk management
- Development is fast and features are added in a systematic way
- There is always a space for customer feedback

51

51

Disadvantages of Spiral Model

- Risk of not meeting the schedule or budget
- It works best for large projects only also demands risk assessment expertise
- For its smooth operation spiral model protocol needs to be followed strictly
- Documentation is more as it has intermediate phases
- It is not advisable for smaller project, it might cost them a lot

52

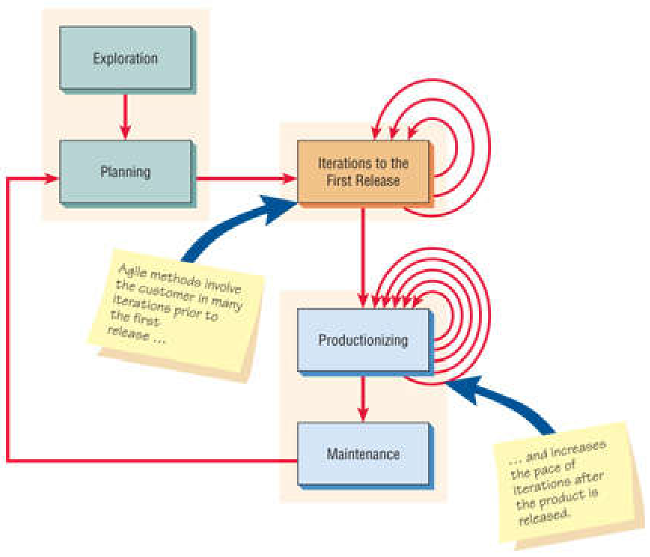
52

Agile Process Models

- Agile software engineering combines a philosophy and a set of development guidelines
- **Philosophy**
  - Encourages customer satisfaction and early incremental delivery of the software
  - Small highly motivated project teams
  - Informal methods
  - Minimal software engineering work products
  - Overall development simplicity
- **Development guidelines**
  - Stress delivery over analysis and design
  - Active and continuous communication between developers and customers

53

53


Exploration
Planning
Iterations to the First Release
Agile methods involve the customer in many iterations prior to the first release ...
Productionizing
Maintenance
... and increases the pace of iterations after the product is released.

54

54

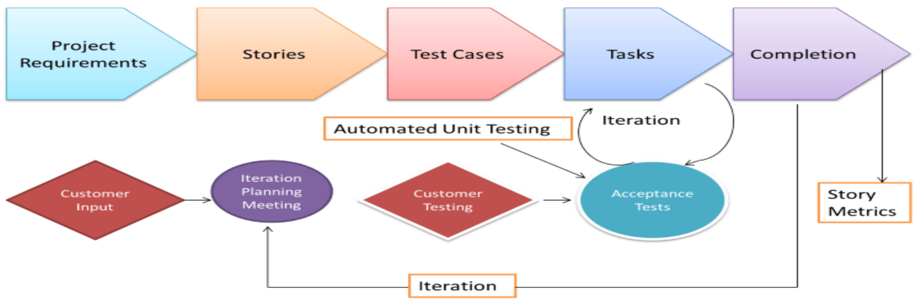
Extreme Programming (XP)

- Somewhat controversial new approach; variation of the incremental model
- First step
  - Determine features that client wants (stories)
  - Estimate duration and cost of each feature
- Client selects stories for each successive build
- Each build is divided into tasks
- Test cases for a task are drawn up
- Pair programming –working with a partner on one screen
- Continuous integration of tasks

55

55

Extreme Programming (contd.)



Extreme Programming (XP)

56

56

Features of XP

- Computers are put in center of large room lined with cubicles
- Client representative works with the XP team at all the times
- There is no specialization
  - all members of the XP team work on specification, design, code, and testing
- There is no overall design phase before various builds are constructed – Refactoring

57

57

Advantages of Agile Model

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

58

58

Disadvantages of Agile model

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

59

59

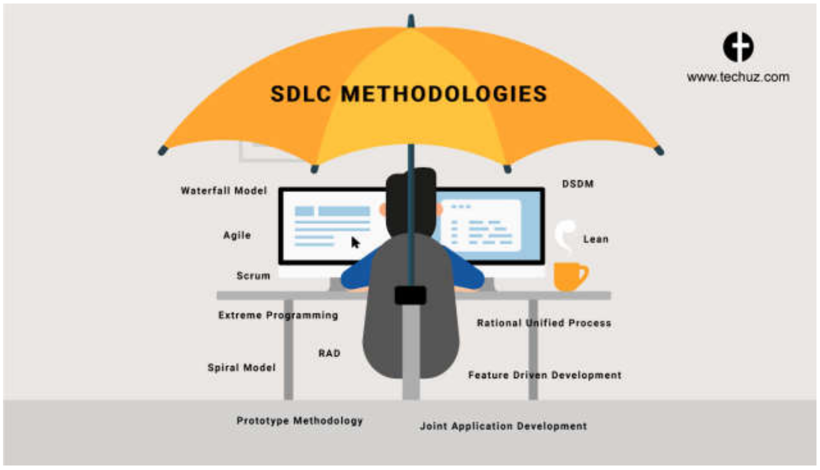
When to use Agile model

- When **new changes need to be implemented**.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way.
- Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project c an continue to move forward without fear of reaching a sudden standstill.

60

60

How to Choose between SDLC Methods?



61

61

How to Choose between SDLC Methods?

- To know which is the best model out of all the different types of SDLC models, it **is important to understand that each of these approaches** are suitable for different projects, environments, and requirements.
- For example, if your project is simple and straightforward with set requirements that do not need to be changed, then Waterfall is best suited for it.
- However, if your project is large-scale and consists of multiple components and segments, then choosing Iterative or Spiral methodology would suit your project better.

62

62

How to Choose between SDLC Methods?

- To answer the question simply, there is **no ONE model is best from all the SDLC models** discussed.
- A preference of one method over the others cannot be determined.
- However, to select the right SDLC methodologies, you should know all the types of SDLC models, assess the requirements of all the stakeholders and then decide on a method that best fits your needs.

63

63

Criteria for deciding on a model include

- Criteria for deciding on a model include
  - Product Complexity
  - Product Size
  - Magnitude of Changes
  - Frequency of Changes
  - Skills of the Dev Team
  - Time constraints
  - Access to Users

64

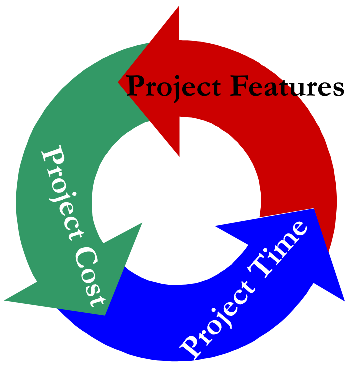
64

Requirement Engineering

65

Goal of software development

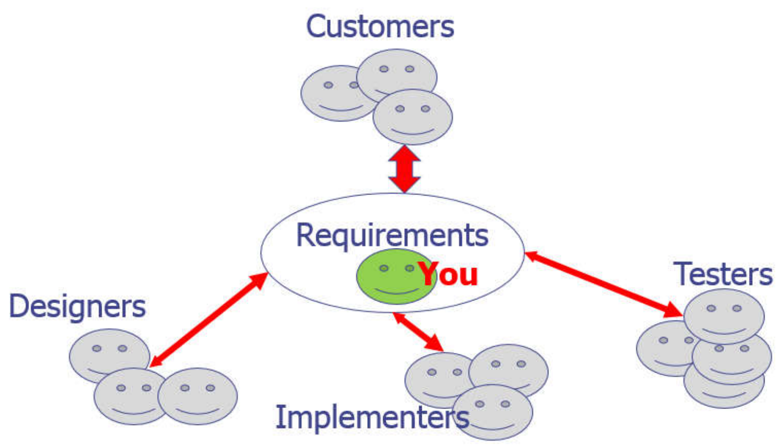
- Develop quality software—on time and on budget—that meets customers' real needs.



66

66

Importance of Requirements Analysis



Requirements Definition

- Abstract description of the services which the system should provide and the constraints under which the system must operate;
- Should be written in such a way that it is understandable by customers without knowledge of specialized notations

Requirements Specification

- Structured document which sets out the system service in detail; Should be precise.

69

69

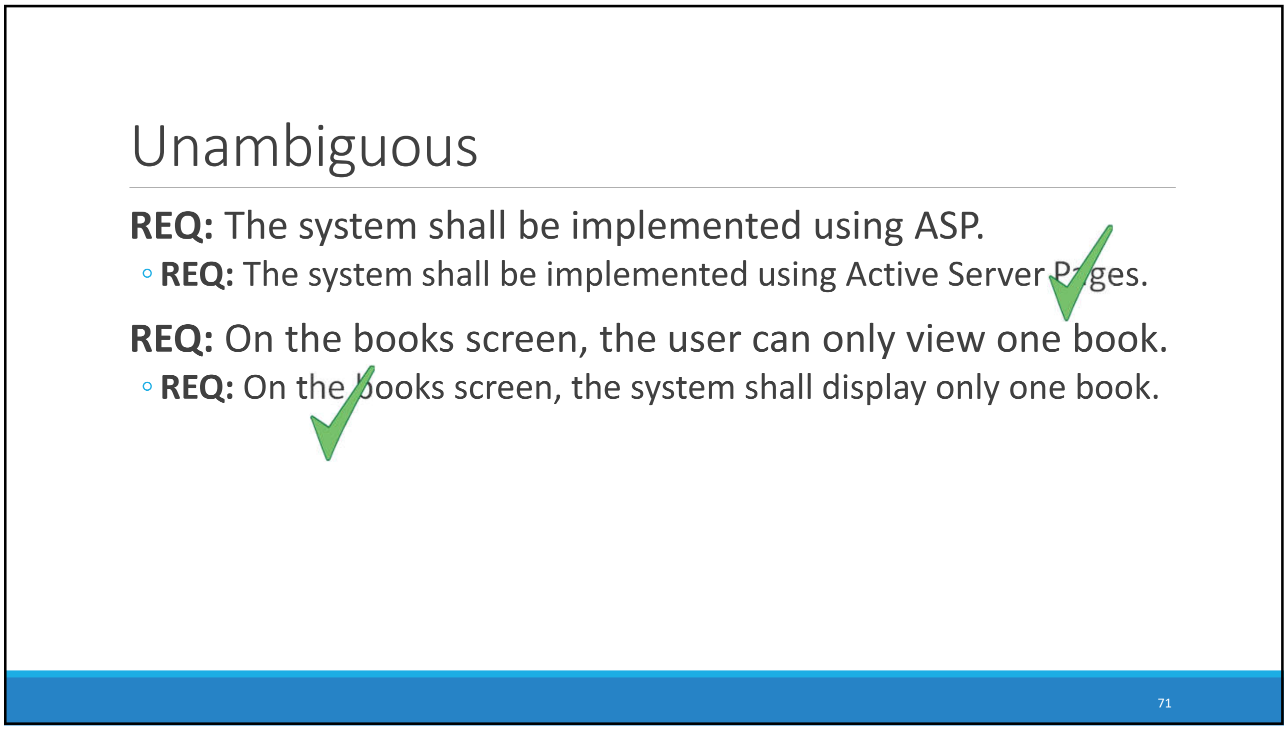
Characteristics of a Good Requirement

✓Unambiguous
✓Testable (verifiable)
✓Concise
✓Correct
✓Understandable

Feasible (realistic)
Independent
Atomic
Necessary
Implementation-free (abstract)

70

70



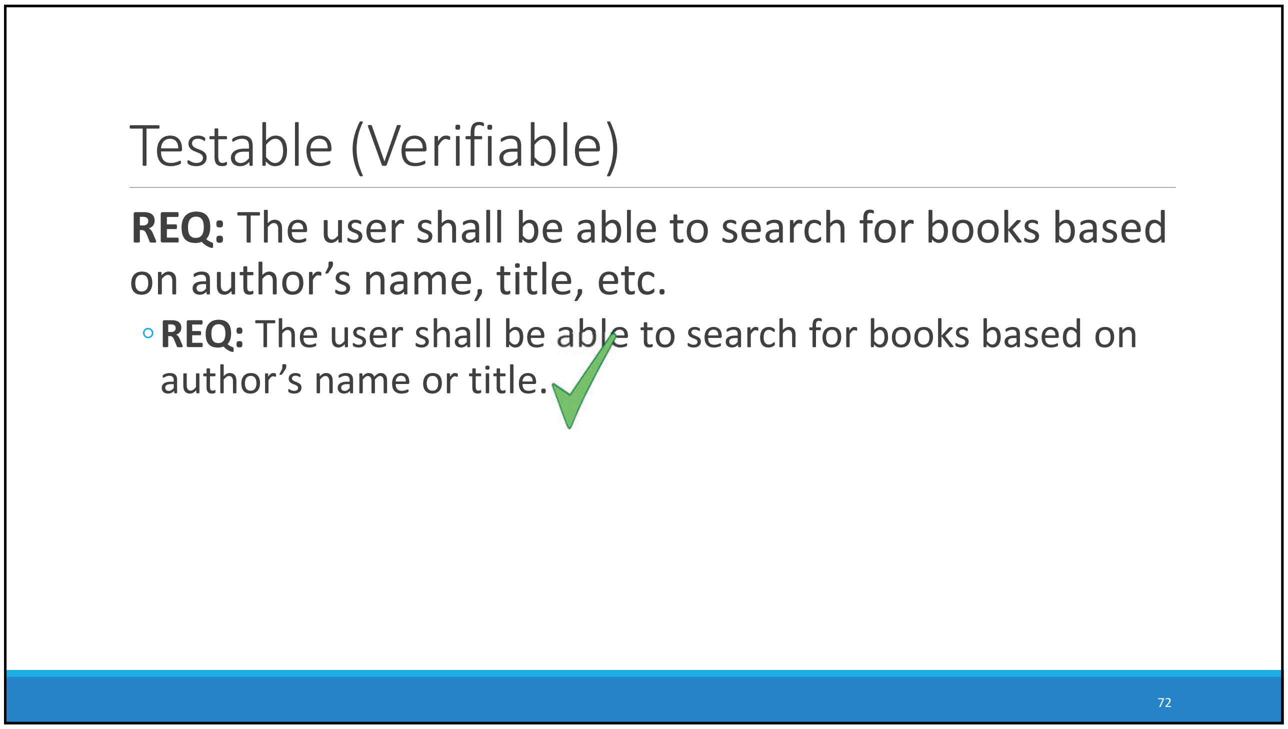
Unambiguous

REQ: The system shall be implemented using ASP.
- **REQ:** The system shall be implemented using Active Server Pages. ✓

REQ: On the books screen, the user can only view one book.
- **REQ:** On the books screen, the system shall display only one book. ✓

71

71

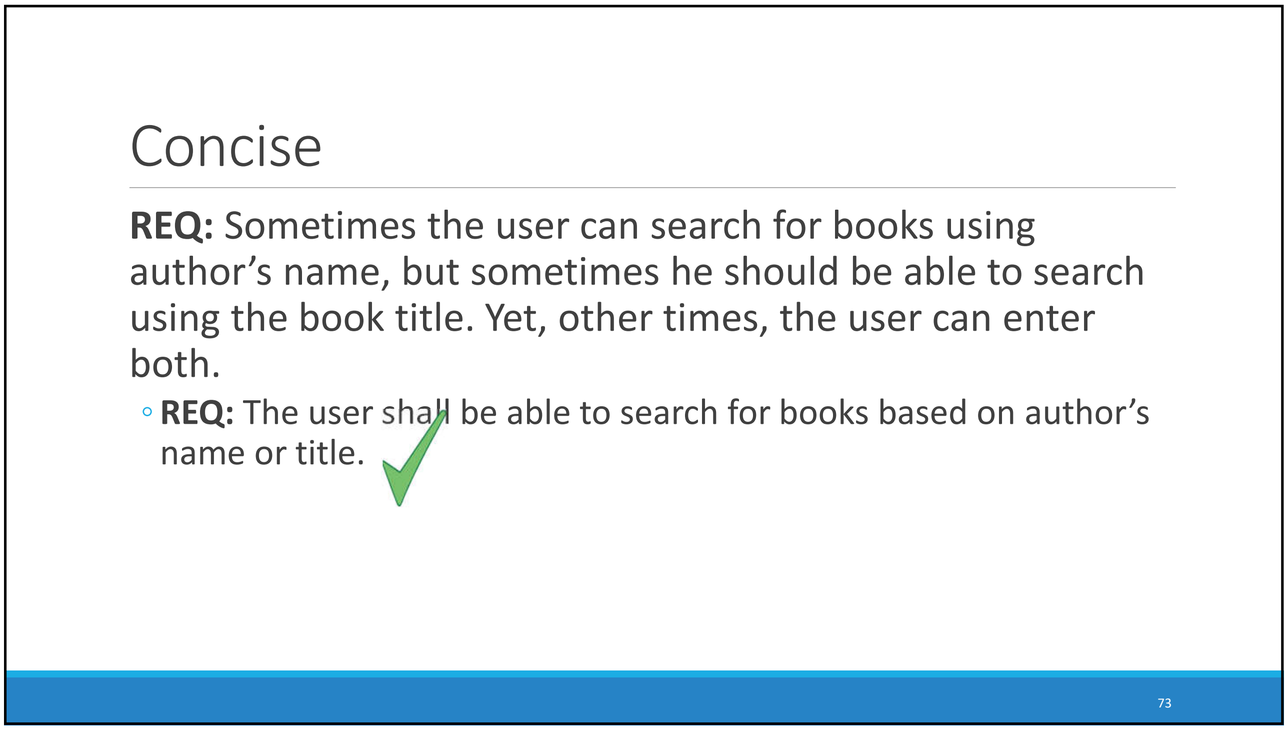


Testable (Verifiable)

REQ: The user shall be able to search for books based on author's name, title, etc.
- **REQ:** The user shall be able to search for books based on author's name or title. ✓

72

72



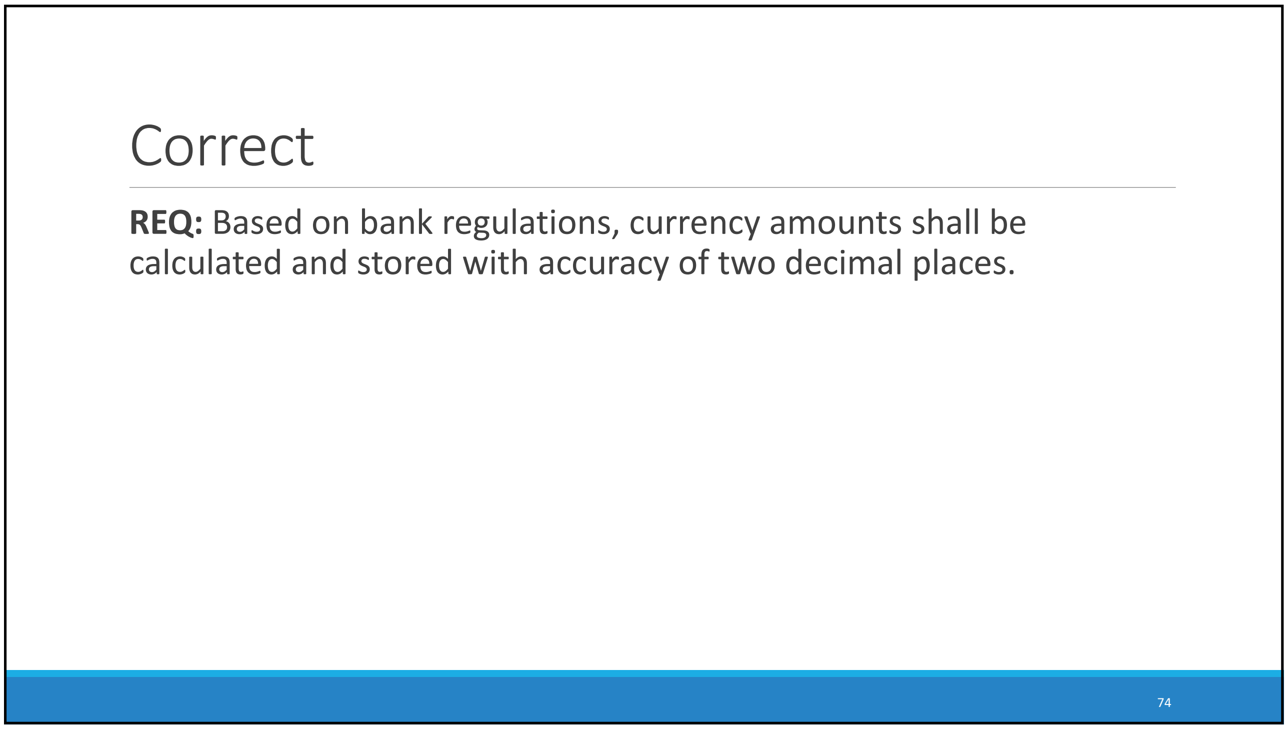
Concise

REQ: Sometimes the user can search for books using author's name, but sometimes he should be able to search using the book title. Yet, other times, the user can enter both.

- **REQ:** The user shall be able to search for books based on author's name or title. ✓

73

73



Correct

REQ: Based on bank regulations, currency amounts shall be calculated and stored with accuracy of two decimal places.

74

74

Understandable

Requirements should be

- grammatically correct
- Written in a consistent style e.g. the word “shall” should be used instead of “will”, “must”, “can”, or “may”

REQ: The system shall remember customer data. ✗

REQ: The system shall displayed order details. ✗

75

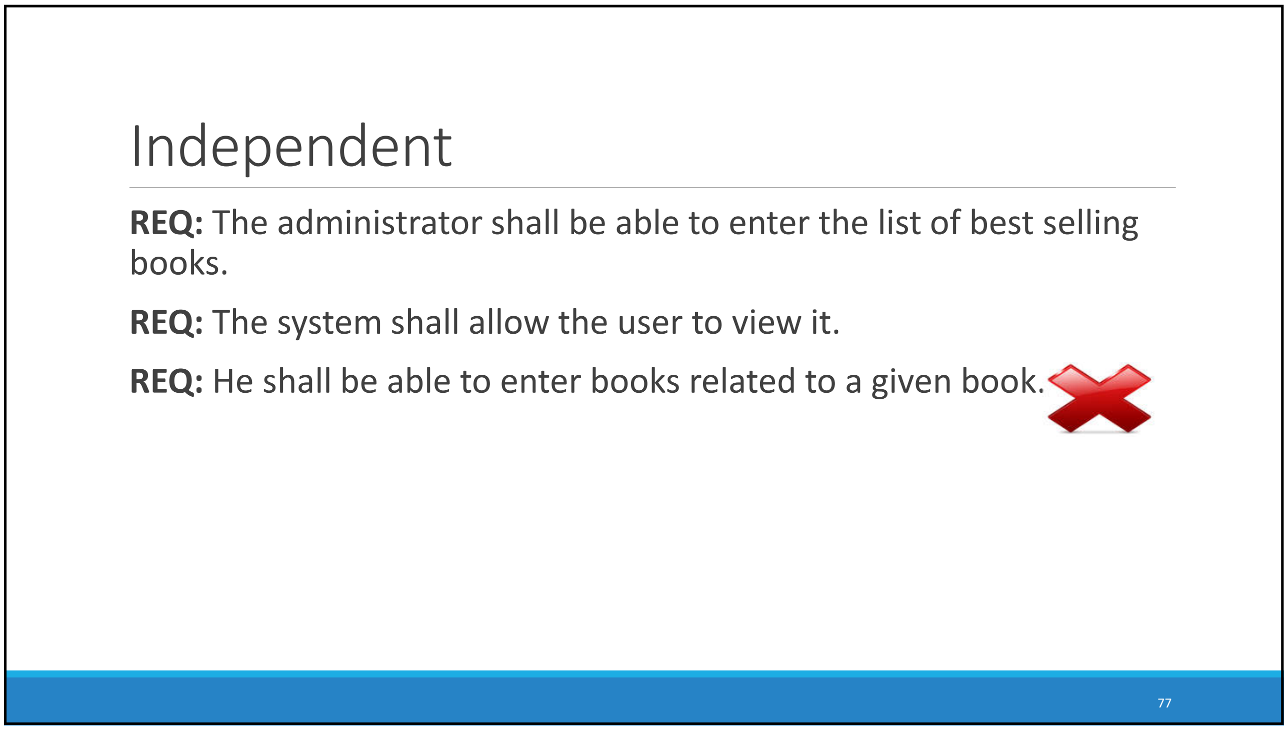
75

Feasible (Realistic)

REQ: The system shall be able to understand commands given in Arabic language. ✗

76

76



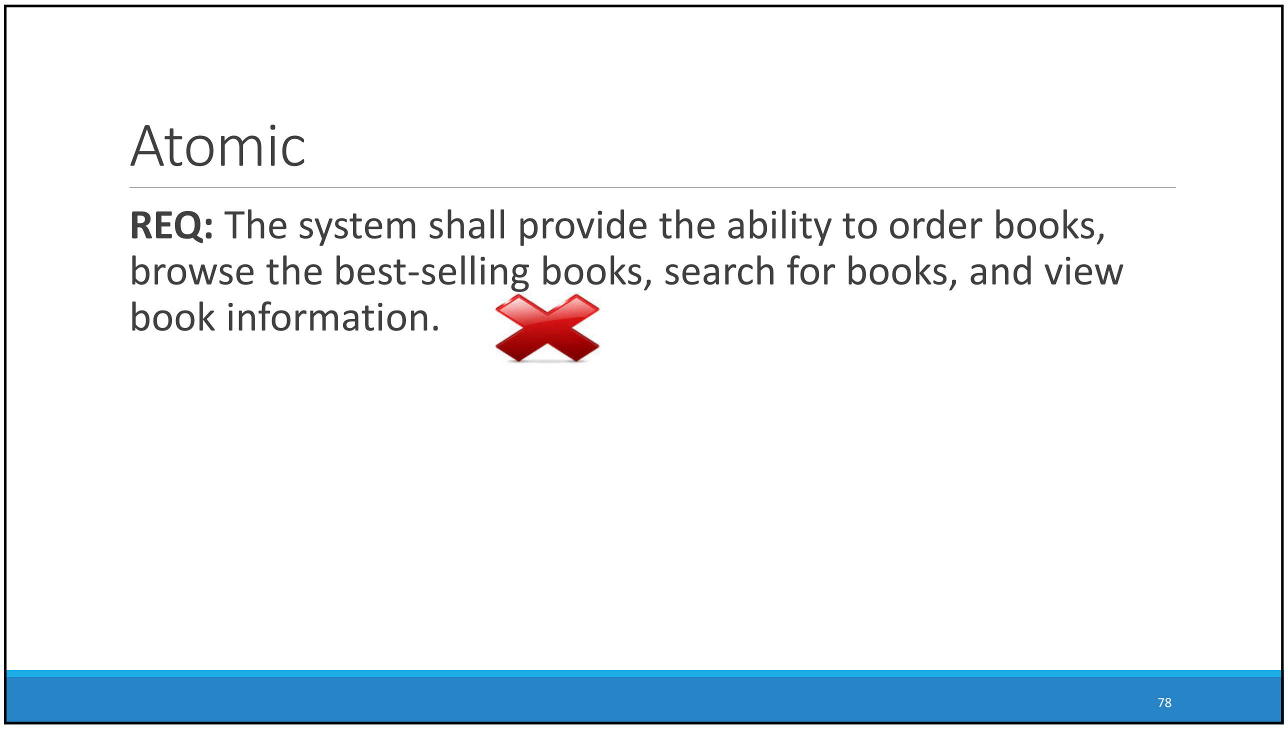
Independent

REQ: The administrator shall be able to enter the list of best selling books.

REQ: The system shall allow the user to view it.

REQ: He shall be able to enter books related to a given book. ✗

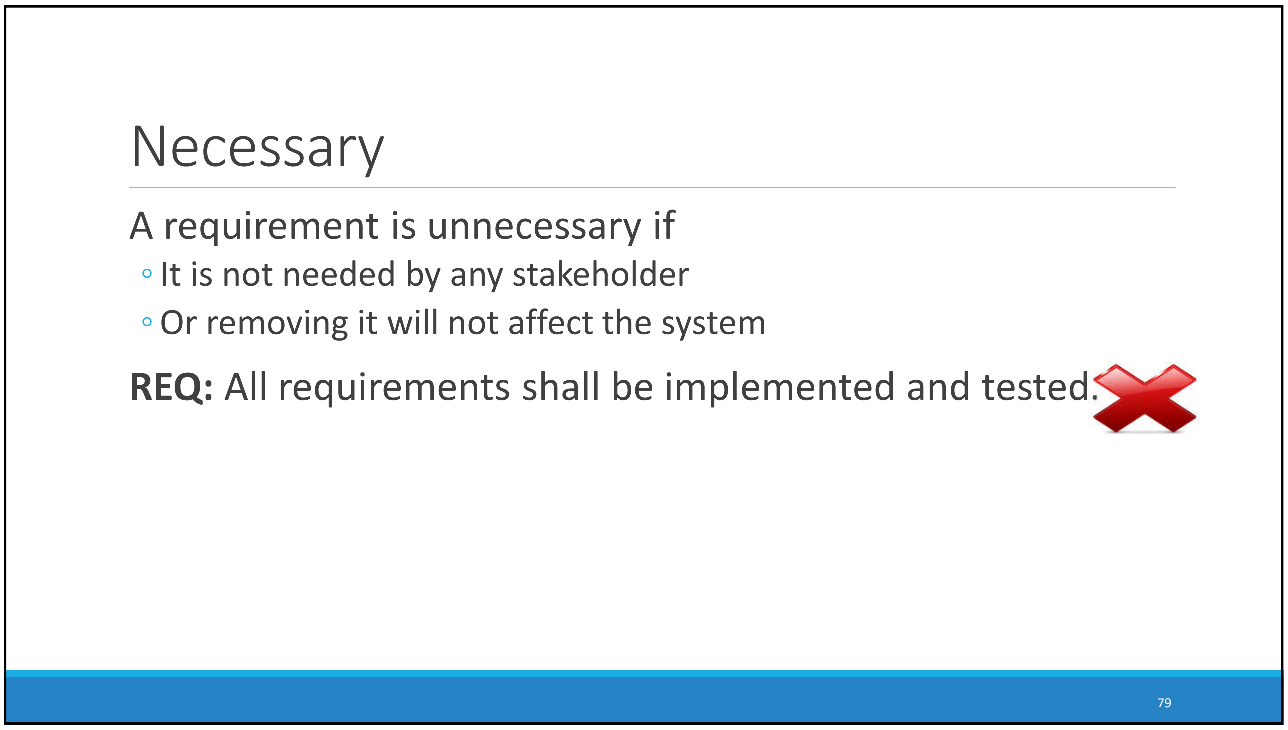
77



Atomic

REQ: The system shall provide the ability to order books, browse the best-selling books, search for books, and view book information. ✗

78



Necessary

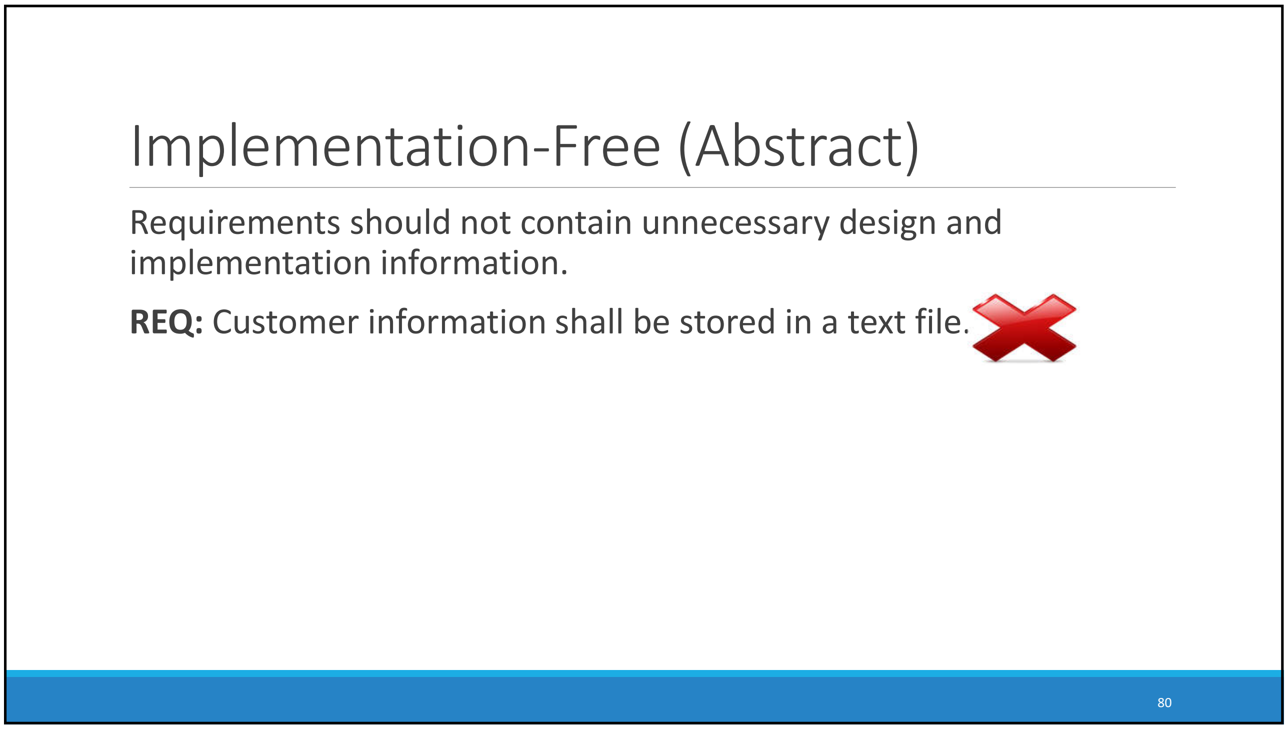
A requirement is unnecessary if

- It is not needed by any stakeholder
- Or removing it will not affect the system

REQ: All requirements shall be implemented and tested. ✗

79

79



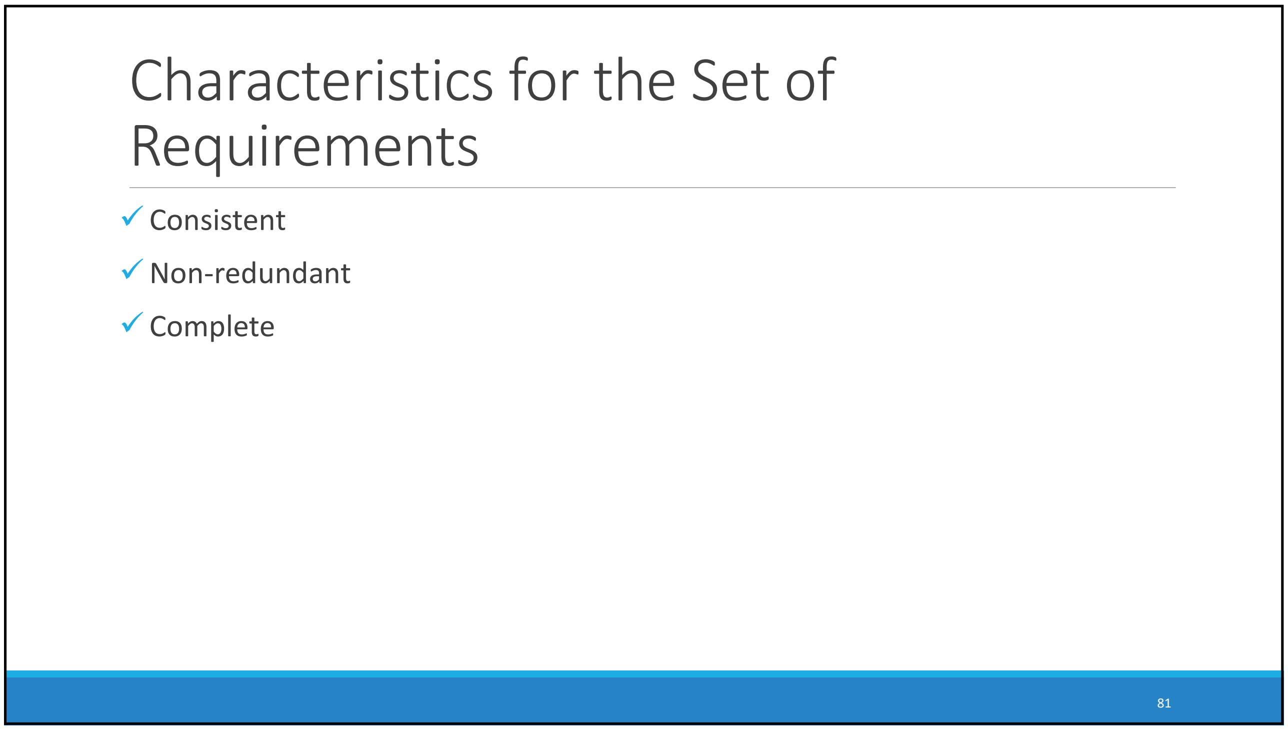
Implementation-Free (Abstract)

Requirements should not contain unnecessary design and implementation information.

REQ: Customer information shall be stored in a text file. ✗

80

80

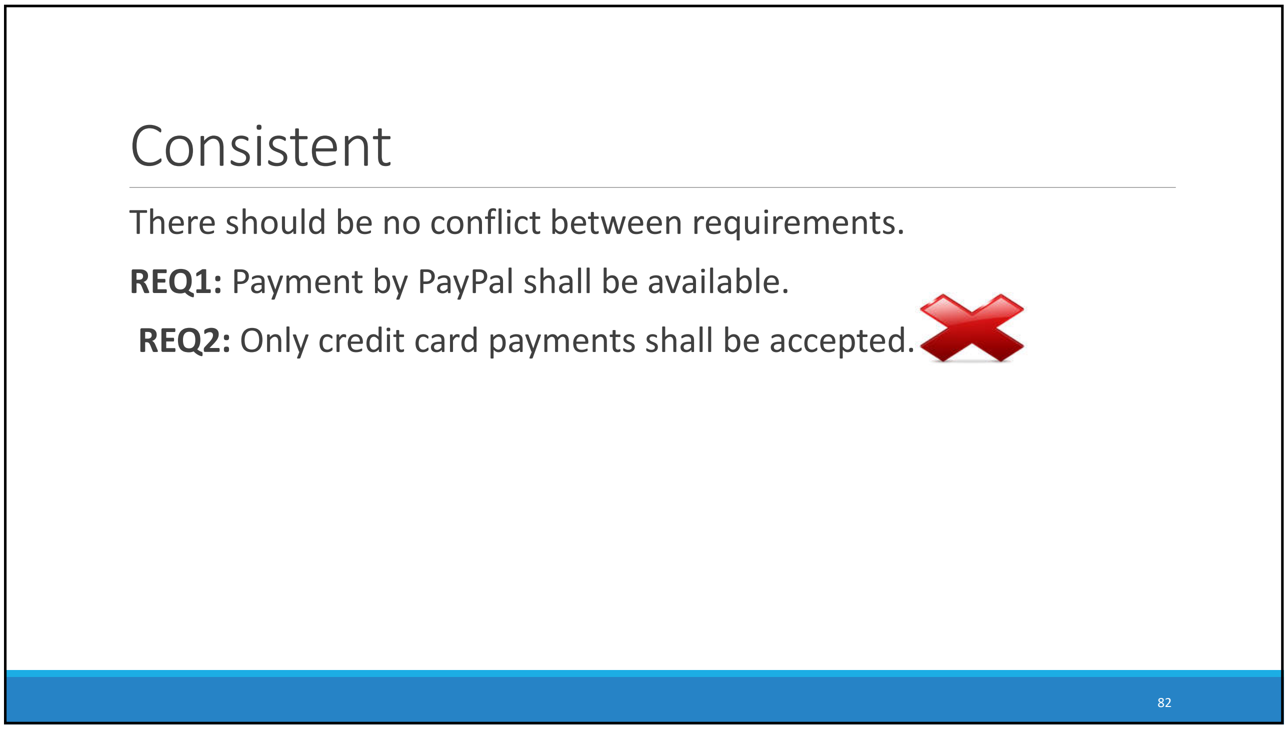


Characteristics for the Set of Requirements

- ✓ Consistent
- ✓ Non-redundant
- ✓ Complete

81

81



Consistent

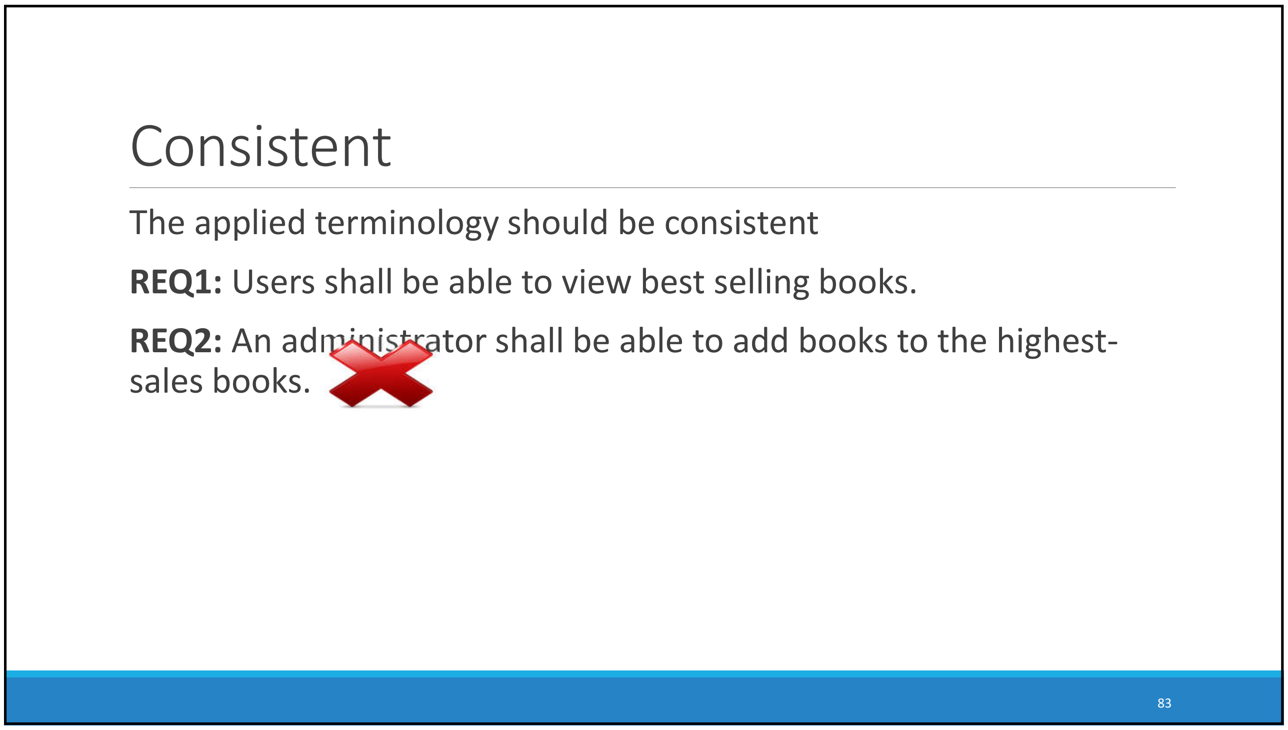
There should be no conflict between requirements.

REQ1: Payment by PayPal shall be available.

REQ2: Only credit card payments shall be accepted. ✗

82

82



Consistent

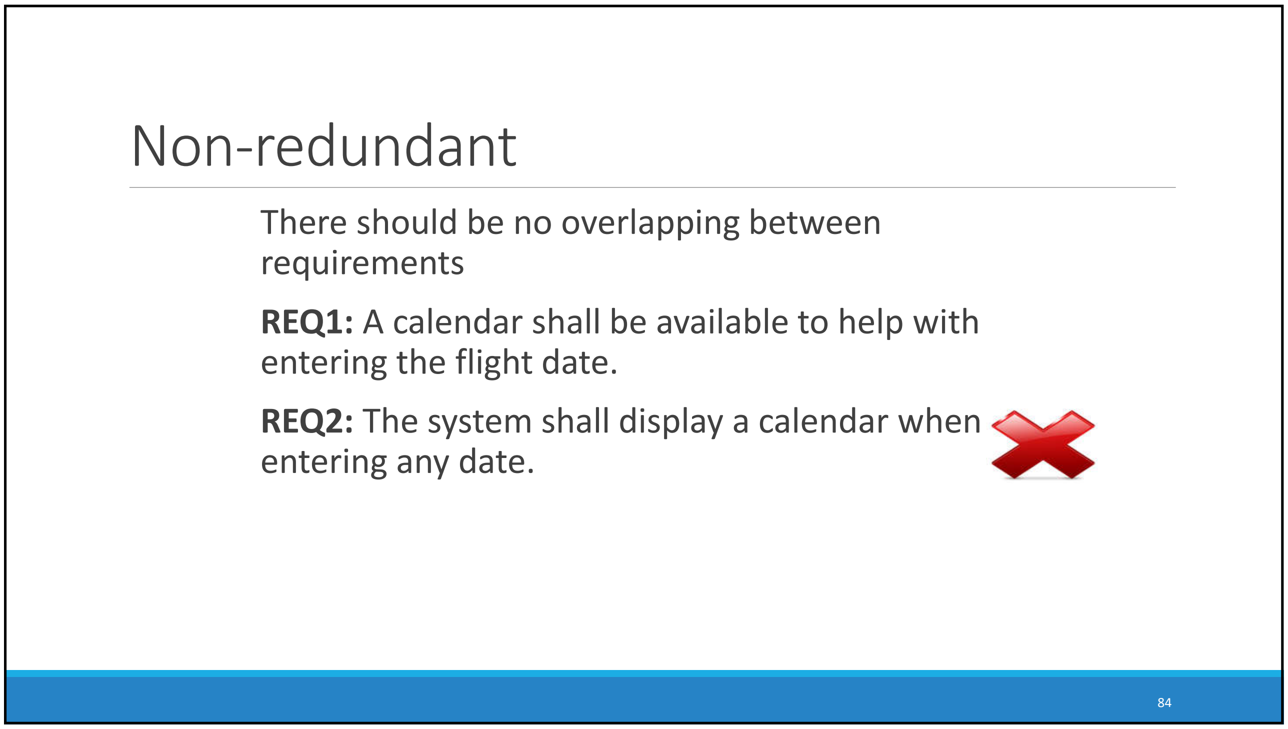
The applied terminology should be consistent

REQ1: Users shall be able to view best selling books.

REQ2: An administrator shall be able to add books to the highest-sales books. ✗

83

83



Non-redundant

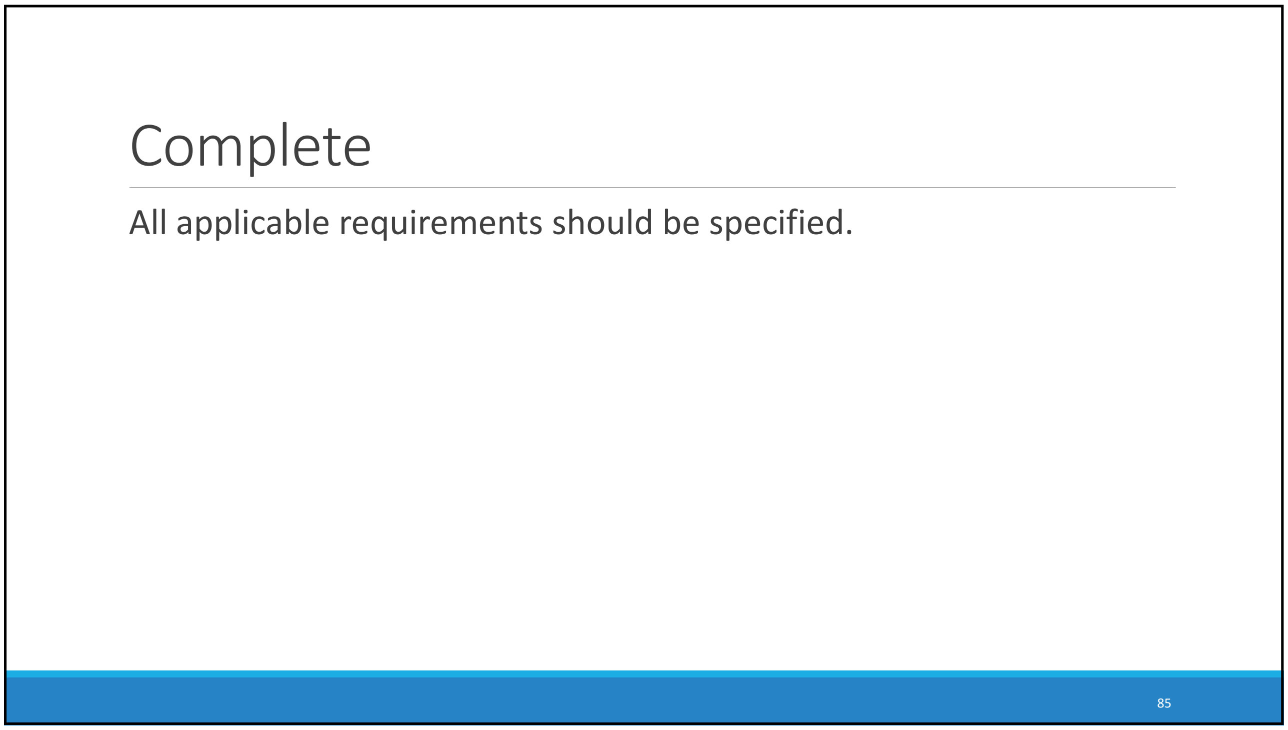
There should be no overlapping between requirements

REQ1: A calendar shall be available to help with entering the flight date.

REQ2: The system shall display a calendar when entering any date. ✗

84

84



Complete

All applicable requirements should be specified.

85

85