Chapter 11: Software engineering and problem solving

This chapter in context:
The following topic appears in Grade 11 IT CAPS Term 2. Some of the sections have been done in Chapters 2 and 5 as indicated:

FROM CAPS:
Solution Development: Software Engineering Principles and Problem solving (±1 week / 4 hours) Delphi (Chap 11 Term 3)

- What is software development?
- Planning and implementing a solution
  - Define/understand the problem/task
    - Read the specifications and analyse the problem/task to determine the requirements
  - Design the interface and the solution
    - Develop a logical solution based on the specifications and analysis as well as sound principles
    - Consider functionality and usability issues in designing the interface
  - Code/implement
    - Incorporate suitable programming constructs in the development of a solution
  - Test and debug the program Delphi (Chapter 5 term 2)
    - Use testing and debugging techniques and methods
  - Document, implement and maintain the program
- Planning techniques using any appropriate tools

Notes:
Diagrams/visual tools for design purposes:
Use any appropriate tools/techniques:
TOE (Task, Objects, Events) charts Delphi (Chapter 2 Term 1)
Noun-Verb analysis
IPO diagrams Delphi (Chapter 2 Term 1)
UML: Use case diagram

Basic principles of software development have been introduced right from the beginning of programming in Grade 10 and continued to be applied from Chapter 1 in Grade 11. Other aspects of this topic have been done in Chapter 2 and 5 of this book.

Chapter 11 Software engineering and problem solving is just a formal description of what learners have been doing all along and therefore provides a nice formal summary of the whole topic.
Activity:

1. List the THREE things that are the minimum requirements for a system specification.
   - What kind of input takes place and when
   - What the processing must accomplish
   - All the different types of output that are required and when they should be generated

2. Why is it a bad idea to work on written specifications that you get from a single person?
   That person might not know all areas of the program, they might never actually do the work
   the program is supposed to make easier, it is better to get input from multiple people.

3. What does it mean to ‘be consistent’ in screen / UI design for your program?
   Keep the same design, colour, fonts, etc. on all pages / screens of your design.

4. Planning your software structure often means following / using a specific formal design
   method.
   a. Name at least FOUR such formal methods.
      Any FOUR from TOE, Flowcharts, IPO, Noun-Verb analysis, UML
   b. Choose one method and briefly describe how it works.
   c. What usually happens in real life?
      People use the methods specified by their company – and these are usually hybrids
      that include features from more than one formal method.

5. Give TWO important tips to follow when writing code.
   Stick to the names and structures decided on in your planning
   COMMENT your code to help make it more readable

6. List THREE things involved in formal testing of software.
   Drawing up a list of possible user actions.
   Creating a table of variables / data structures with low, normal, high and ‘out of range’
   values.
   Repeatedly running the program and carrying out the user actions / entering the data values
   and checking to see if the program behaves as expected.

7. When you have finished designing, writing, debugging and testing your program you are not
   yet finished. List THREE things that still need to be done.
   Any THREE of:
   - create documentation (either printed or online),
   - create help files,
   - work out a way to install the software,
   - come up with a way to distribute the software (CD / DVD / Online) and, finally,
   - set up ways to support, maintain and update the software.