Teacher's Comments:

*This is a realistic example of the **analysis** of a sample problem. The write-up is a bit too brief – this might only get 8 or 9 points out of 12. But it contains all the necessary pieces. You could start like this, and then clean it up and expand it to make a really nice analysis – try to get all 12 points. Read the Assessment Criteria at the end so you know what is required. Make sure you don't leave anything out – a single missing item could cost several marks.*

GOPA - Grades Online with Parent Access

== Current System ==

Mr M is a Computer Science teacher at our school. He teaches programming, graphics, and computer technology classes. The students do lots of practical work, like making web-pages, graphics designs, and writing computer programs. He collects most of the assignments electronically by copying onto a server. After grading the work, he uses a Word Processor to record grades and comments. He prints these on paper and gives them back to the students. He also types the same grades into a spreadsheet. He finds the whole process cumbersome and annoying.

Mr M used to have a normal paper gradebook. He says that was unsuitable, because he needs to write comments somewhere. Since he doesn't collect paper documents (like English teachers collecting essays), he has no good place to write the comments and return them - hence the Word Processor.

-- Sample Data --

Grades and Comments in WP document	Spreadsheet Gradebook
Alex Robertson Graphics Design : A bit too simple - how about drawing some diameters or radius? Sensible Problems : Yes Random Numbers : Yes General Programming : Clear and correct Grade = 7- --- Ann Prince Graphics Design : Brilliant picture!! Sensible Problems : Excellent! Random Numbers : Yes General Programming : Clear and correct Grade = 7+	***Graphics*** **Greenfoot Stickman** **Alex Robertson 7- inc** **Ann Prince 7+ inc** **David Anderson 6 6.7** **Ellen Lynx abs 7** **Fred Fight 5 6.7** **Herman Adams 7 6** **..... ....**

== User Stories (scenarios) ==
I discussed Mr M's needs and recorded the following stories (scenarios.)

- **Returning grades to the students**
  "I often type comments and grades into a WP document, then print it and cut it up into pieces and give each student a little slip of paper. This is easy, but the slips are small and go missing. And printing, cutting, and distributing is just another bunch of meaningless steps."

- **Getting grades to parents**
  "I'd like the parents to be able to see their child's grades without me sending them an E-mail or making a phone call. I type comments anyway - I could E-mail them but that would be very time consuming (unless it were automated).

- **Record keeping**
  "I use a WP to write grades and comments, then copy the grades into a spreadsheet (type them), and at the end of term I type report cards into the school's database. I'd like to just keep one copy of the grades - maybe copy the the comments into the report system. But I end up with a bunch of disorganized WP files - it all needs to be simpler and better organized."

- **Sharing student work**
  "My students make nice web-pages and write impressive programs. I'd like to share these with the parents so they could see why their child is getting a good (or bad) grade. It would also be nice to show really good results to other students. But I don't manage to keep the students' work organized sufficiently to make this a reality."

- **Missing work**
  "It's hard to track missing work. Students miss deadlines due to absences, field trips, computer problems, etc. Then I have a blank in my spreadsheet, but I don't really review this often enough. And parents don't know when their children are behind - unless I notify them. I'd like that all to be easier - a bit more automatic - so we can monitor progress easily."

- **Easier report cards**
  "I hate writing reports. Every term I intend to look back at my comments on specific assignments when writing reports. But my records are poorly organized, so I seldom do this. I think if I could call everything up automatically from a database, it might actually work."

- **Easy and Automatic**
  "It's really important that this is easy and automatic. I spend lots of time preparing lessons - I don't have enough time for marking, and I'm sure not going to waste time making extra web-sites and writing E-mails to parents, even if it does help the students."

== Intended Improvements ==

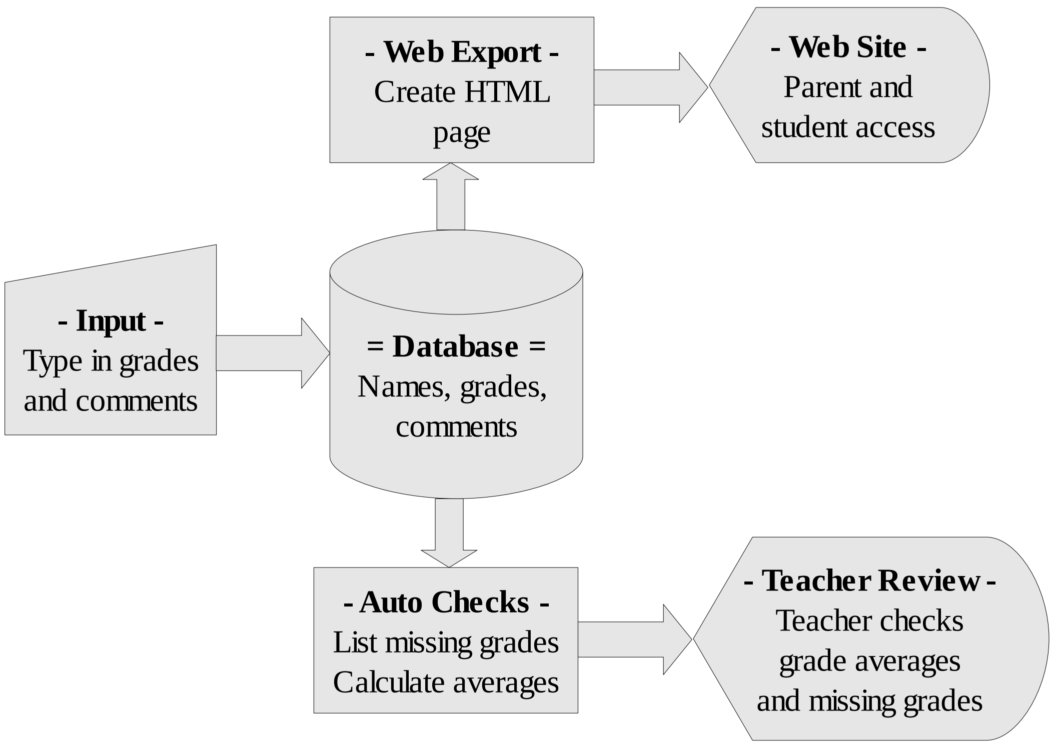
Mr M would like a simpler, more integrated system for recording his grades and sending them back to students. He tried using Email, but that is very cumbersome as the students have various Email providers. And he ends up with lots and lots of single E-mails - very disorganized. He also wants to share the grades and comments with parents. This should all be done without paper, as he finds printing and distributing paper cumbersome. Mr M already has a web-site. He wants to integrate this grading software with the existing web-site. For example, if parents and students can access their grades they should do it at this web-site.

-- General Wishes --

- All electronic - no paper
- Online with parent and student access (with password restrictions)
- Flexible formats (grades and comments)
- Integrated with current web-site

== Initial System Design ==

This application will need to store student names, grades and comments in a data-file. Mr M needs access to record grades and comments. The system will need to export the gradebook to a web-page format, to be stored on Mr M's web-site. And parents and students will need access to the web-site.



== Prototype ==
Following are sample user-interface screens for various tasks. These were discussed with Mr M.

-- Inputting grades and comments --

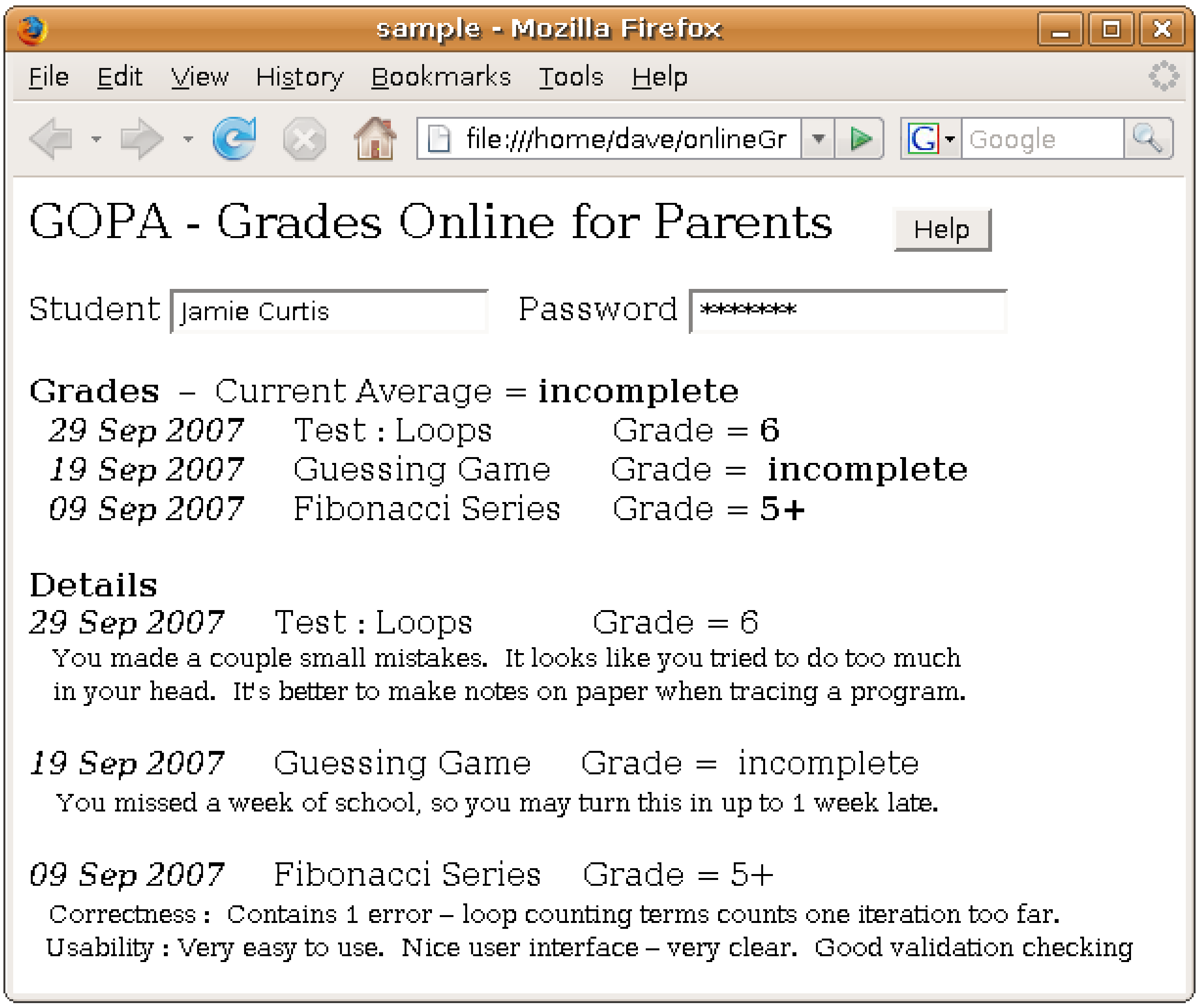
Grade entry screen

Assignment [Fibonacci Series]
Student [Jamie Curtis]
Grade [5+]

Comments

Correctness :
Contains 1 error – loop counting terms counts one iteration too far.
Usability :
Very easy to use. Nice user interface – very clear. Good validation checking on inputs.

-- Student and Parent Access to Student Grades --


sample - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///home/dave/onlineGr
Google
Help

GOPA - Grades Online for Parents

Student Jamie Curtis   Password ********

Grades – Current Average = **incomplete**
29 Sep 2007 Test : Loops Grade = **6**
19 Sep 2007 Guessing Game Grade = **incomplete**
09 Sep 2007 Fibonacci Series Grade = **5+**

Details
29 Sep 2007 Test : Loops Grade = 6
You made a couple small mistakes. It looks like you tried to do too much in your head. It's better to make notes on paper when tracing a program.

19 Sep 2007 Guessing Game Grade = incomplete
You missed a week of school, so you may turn this in up to 1 week late.

09 Sep 2007 Fibonacci Series Grade = 5+
Correctness : Contains 1 error – loop counting terms counts one iteration too far.
Usability : Very easy to use. Nice user interface – very clear. Good validation checking

Notice - this is a web-page that parents can access. The HTML code must be exported by the Java application. The actual mechanism for the password must still be determined.

== **Grade Summaries** ==

```
Missing Grades

Bozo Clown
  19 Sep 2007     Guessing Game
     Unless you bring an excuse for your absence, this grade will be a zero.
  09 Sep 2007     Fibonacci Series
     Unless you bring an excuse for your absence, this grade will be a zero.

Jamie Curtis
  19 Sep 2007     Guessing Game     Grade =  incomplete
    You missed a week of school, so you may turn this in up to 1 week late.
```

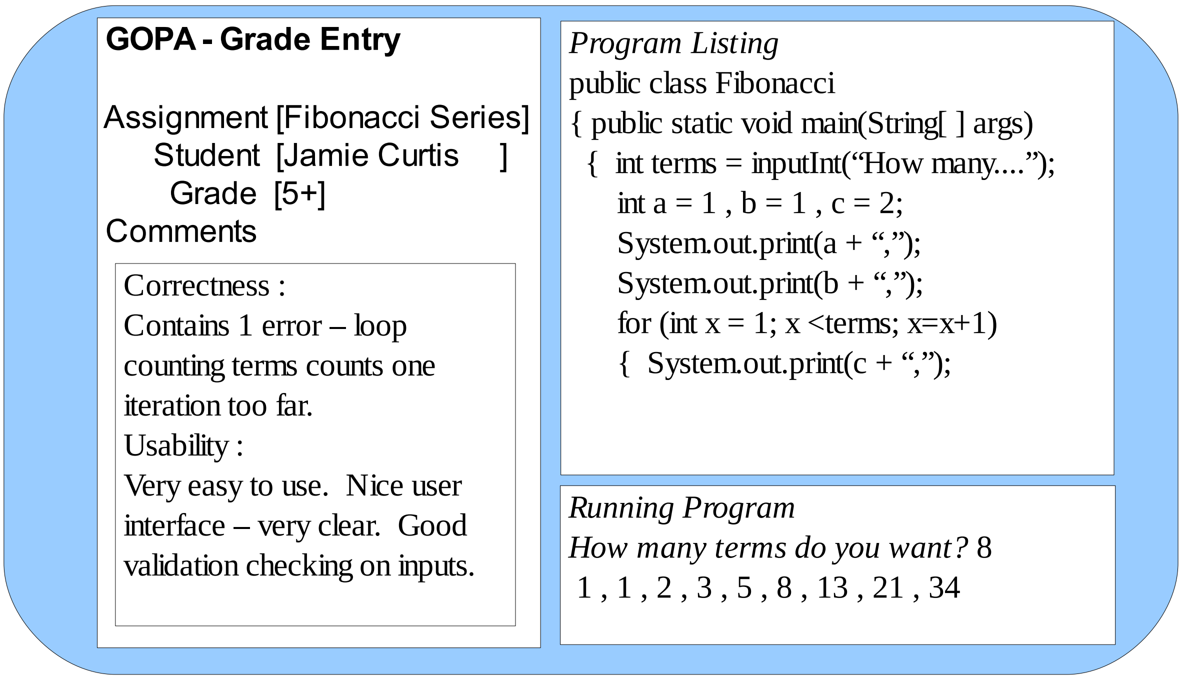
```
Grade Summaries

Bozo Clown -  Ave = 2.33
  29 Sep 2007     Test : Loops          Grade =  7
  19 Sep 2007     Guessing Game     Grade =  0
  09 Sep 2007     Fibonacci Series    Grade =  0

Jamie Curtis – Ave = incomplete
  29 Sep 2007     Test : Loops          Grade = 6
  19 Sep 2007     Guessing Game     Grade =  incomplete
  09 Sep 2007     Fibonacci Series    Grade = 5+
```

```
Individual Student  Summary

= Jamie Curtis =

Grades  –  Current Average = incomplete
29 Sep 2007     Test : Loops          Grade = 6
19 Sep 2007     Guessing Game     Grade =  incomplete
09 Sep 2007     Fibonacci Series    Grade = 5+

Details
29 Sep 2007     Test : Loops          Grade = 6
   You made a couple small mistakes.  It looks like you tried to do too much
   in your head.  It's better to make notes on paper when tracing a program.

19 Sep 2007     Guessing Game     Grade =  incomplete
   You missed a week of school, so you may turn this in up to 1 week late.

09 Sep 2007     Fibonacci Series    Grade = 5+
  Correctness :  Contains 1 error – loop counting terms counts one iteration too far.
  Usability : Very easy to use.  Nice user interface – very clear.  Good validation
checking on inputs.
```

== Specific Goals and Features ==
After the initial discussion with Mr M, I developed the prototype interface screens above. I showed these prototype screens to Mr M and he suggested the following changes:

- The same module could be used for parent access and for teacher review of student grades, as Mr M likes the "look and feel" of web-pages. But when the teacher is reviewing grades, the results should be produced directly from the database in real-time, so that all updates are included. Then the web-page should appear in a browser automatically.

- Mr M said he wants to have a programming IDE (or other application) running at the same time as the gradebook when he is marking work. So the grade-entry screen should be a bit smaller, like this:



GOPA - Grade Entry

Assignment [Fibonacci Series]
Student [Jamie Curtis]
Grade [5+]
Comments

Correctness :
Contains 1 error – loop counting terms counts one iteration too far.
Usability :
Very easy to use. Nice user interface – very clear. Good validation checking on inputs.

Program Listing

```
public class Fibonacci
{ public static void main(String[ ] args)
  {  int terms = inputInt(“How many....”);
     int a = 1 , b = 1 , c = 2;
     System.out.print(a + “,”);
     System.out.print(b + “,”);
     for (int x = 1; x <terms; x=x+1)
     {  System.out.print(c + “,”);
```

Running Program

```
How many terms do you want? 8
 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34
```

- Mr M pointed out that he needs to be able to add students to his class list.
  He also needs several different class lists. So he suggested this interface:

Class Lists Management

IB Comp Sci yr 1	*IB Comp Sci yr 2*	*Graphics*	*Computer Tech*
Adams, Alan	Duck, Donald	Giraffe, George	Mouse, Mickey
Baker, Billie	Engel, Eva	Hen, Henna	......
Cho, Carla	Fox, Francis	......	
......	......		

- Mr M was worried about the passwords. He said he didn't know how to assign them and did not want to do it by hand. He suggested an automated process, like the following:

```
Which class do you want to assign passwords to?  Graphics
Finished assigning 20 passwords.
Results are stored in  c:\GOPA\Graphics.pwd
```

== Goals - Criteria for Success ==
Consistent with Mr M's needs and the prototype, we agreed on the following goals:

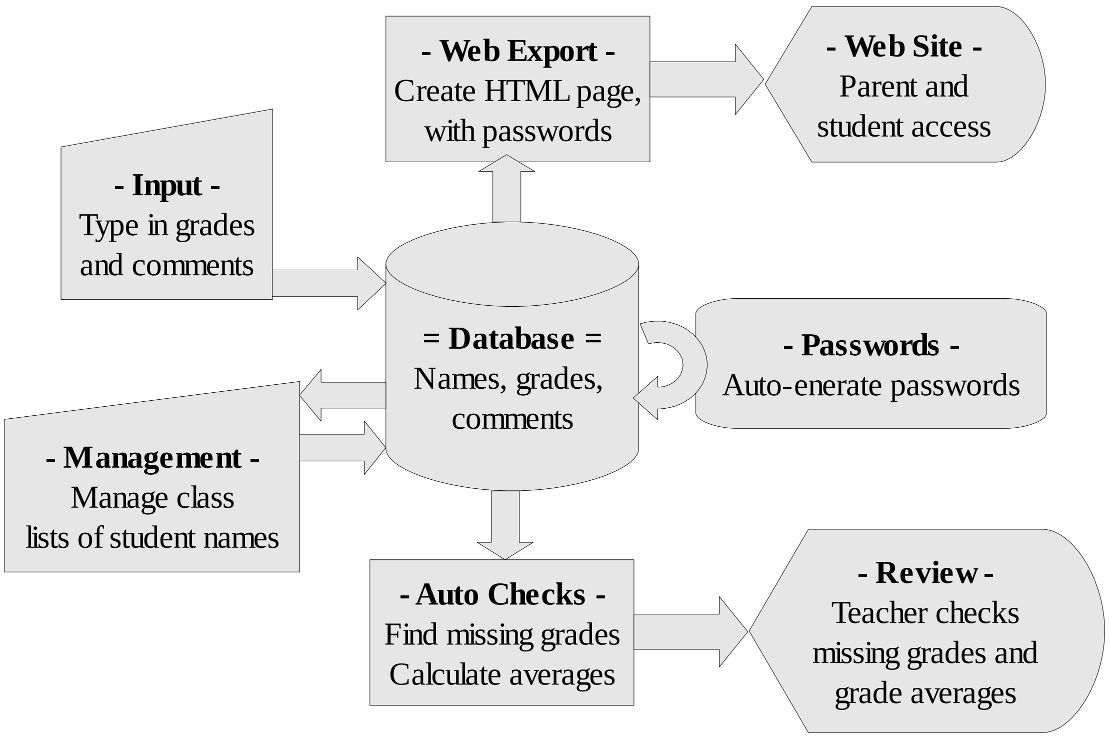
- **Grading (storage)**
    - Grades include a number grade, comment, and web-link to student work
    - Grades stored in a random-access file
    - Automatically calculates an average grade for each student

- **Access (output)**
    - Export data to an HTML page (web-page) and post it on a web-site
    - Security - passwords required so students (and parents) can only see their own grades
        - Need a custom programmed security system - probably won't find a ready-made one

- **Enter grades (input)**
    - Should be easy to open and view work submitted (e.g. essay, program, web-site, etc)
    - Type in grade(s)
    - Type in comments

- **Management**
    - Adding (and removing) students to class lists
    - Calculating average grades
    - Search for missing grades

== Revised System Design ==



== Limitations/Restrictions ==

- Equipment -
Mr M will use his PC and the intranet at school. He will use his laptop and the Internet at home. He has a drive-mapping to the Web-Server, so it is easy to copy the generated web-pages to his web-site. Thus, there will be now upload facilities provided in the application. There will be no attempt to test the application on other platforms such as Linux and Mac OS. It might work anyway, but there are no guarantees.

- Web Access -
Parents will need a normal Internet connection to view their child's grades. Little data is transferred, so there should be no bandwidth issues. The results will be tested on a PC with Internet Explorer. If they use a different platform, it might work but there are no guarantees.

- Database Size(s) -
The total size of the database is limited by available disk storage and network restrictions. It is not anticipated that the database will require more than one megabyte of storage, so this should not be a problem. The size of comments will be restricted to 1024 bytes for each assignment. Comments will be stored as pure text (no formatting).

- Portability/Compatibility -
The application will be configured to run from a single folder on a disk drive. It should still function when the entire folder is copied to a USB stick. Mr M will type everything in English, so there is no need for foreign language support.

Criterion A1: Analysing the problem

The documentation should be completed first and contain a thorough discussion of the problem that is being solved. This should concentrate on the **problem** and the goals that are being set, not on the method of solution. A good analysis includes information such as sample data, information and requests from the **identified end-user**, and possibly some background of how the problem has been solved in the past. A **systematic method** is one that takes into account what input and output will occur and what calculations and processes will be necessary to obtain the desired output.

0 : The student has not reached a standard described by any of the descriptors given below. For example, the student has simply described the programmed solution.
1 : The student only **states** the problem to be solved **or shows** some evidence that relevant information has been collected.
2 : The student **describes** the problem to be solved.
3 : The student describes the problem **and provides evidence** that information relating to the problem has been collected.
4 : The student provides evidence that a **systematic method** has been used in the analysis of the problem.

This section of the program dossier would typically be two to three pages in length. It should include a brief statement of the problem as seen by the end-user. A discussion of the problem from the end-user's point of view should take place, including the user's needs, required input and required output. For example, evidence could be sample data, interviews and so on, and could be placed in an appendix.

Criterion A2: Criteria for success

This section of the program dossier will clearly state the objectives/goals of the solution to the problem. The expected behaviour of the solution should be clearly described and the limits under which it can operate outlined.

0 :	The student has not reached a standard described by any of the descriptors given below.
1 :	The student **states some** objectives of the solution.
2 :	The student **describes most of** the objectives of the solution.
3 :	The student **relates all of** the objectives of the solution to the analysis of the problem.
4 :	The student relates all of the objectives of the solution to the analysis of the problem, and **outlines** the limits under which the solution will operate.

This section of the program dossier would typically be one to two pages in length. Objectives should include minimum performance and usability. These criteria for success will be referred to in subsequent criteria, for example criterion C2 (Usability), C4 (Success of program); D2 (Evaluating solutions) and D3 (Including user documentation).

The limits under which the solution will operate will vary. Some examples are:
• Time taken to return a research result from a data file
• The response of the program to invalid and extreme data input
• Limitations on the volume of data stored in the program
• Usability of user input screen
• The proper response of the program to user input.

Criterion A3: Prototype solution

The prototype solution **must** be preceded by an initial design for some of the main objectives that were determined to be the criteria for success. A prototype of the solution should be created.
A prototype is: "The construction of a simple version of the solution that is used as part of the design process to demonstrate how the system will work."

0 : The student has not reached a standard described by any of the descriptors given below.
1 : The student includes only an **initial design**.
2 : The student includes an initial design **and a prototype**, but they do not correspond.
3 : The student includes an initial design and a prototype that **corresponds**.
4 : The student includes an initial design and a complete prototype that corresponds to it **and documents user feedback** in evaluating the prototype.

The prototype need not be functional, it could be constructed using a number of tools such as: Visual Basic, PowerPoint, Mac Paint, Corel Draw for a simple Java program. The intent is to show the user how the system is expected to operate, what inputs are required and what outputs will be produced. A number of screenshots will be required for the user to be able to evaluate the solution properly. The prototype, at its simplest, could be a series of clear, computer-generated drawings, a hierarchical outline of features in text mode, or a series of screenshots.

Documentation of user feedback could be, for example, a report of the user's comments on the prototype

== Tips and Suggestions ==

-- How do I get started? --

Don't start by writing a computer program. You can do a bit of programming early (functional prototype), but some investigation and design **must be done first**. Choose a **specific problem** and identify an **intended end-user** before doing anything else.

-- Aiming for Maximum Marks --

IA assessment is **criterion** based. This is not a test. There are no "right answers". There are no "points" to add up.

Your documentation must **meet the criteria** to score maximum marks. For example, if you don't have an "intended user", you cannot score 4 marks in section A3, and probably cannot score more than 2 marks in section A1.

** **Read** the criteria and **follow them. ****

-- Mastery Factors --

Check with your teacher early on to ensure the problem and your solution are sufficiently complex to encompass 10 mastery factors. For example, if you don't store any data in data files, you won't fulfill mastery of files (SL) or random-access files (HL). Even in the early stages, you must be looking forward to the programming process.
A missed mastery mark carries a 10% penalty!

-- Okay, how do I proceed? --

You need to complete the **(A) Analysis** and **(B) Detailed Design** before you start writing the program. The following process for Stage A is intended to :

(1) meet the **criteria** for maximum marks;

(2) be **doable** by a student;

(3) get **maximum benefit** from a moderate investment of time and effort.

** Be sure to **keep records** of all your work (even scrap paper) – you will need them later. **

== Suggested Process for Stage A - Analysis ==

1. Choose a **problem** (be very **specific**)

2. Choose the **intended-end-user(s)** (at least one real person) –
   this should **not** be "everybody" or "me myself"

3. **Describe** the problem, including **current/previous solutions.**
   What happens now? How is it done? What is unsatisfactory?

4. **Collect** sample **documents** and **data** from the current solution

5. Outline **intended improvements** of a new system

6. Create an **initial system design** (simple) for the new system

7. Create a **prototype** – either **functional** (a program) or **mocked-up** (interfaces only).

8. *optional:* Use a functional prototype to investigate the **feasibility** of programming the new system (solve some tricky problems)

9. Use the prototype(s) to **discuss** the new system with the user

10. Collect the user's **reactions** and **suggestions** (written down)

11. **Document** the most significant **user stories** which identify the major **tasks** to be performed with the new system

12. **Improve** the **system design** and **interfaces** to meet the **user's suggestions** and fulfill the needs in the **user stories**

13. Write a **clear** and **complete** set of **goals** – extracted from the user stories and revised design – specifying what it **WILL do**

14. **Clarify limitations** of the intended system – what it **WON'T do**

15. **Review goals and limitations with the intended user and revise (if necessary)
    until the user is satisfied.**