**Analysis Checklist**

**Hand-in deadline 8 September 2018**

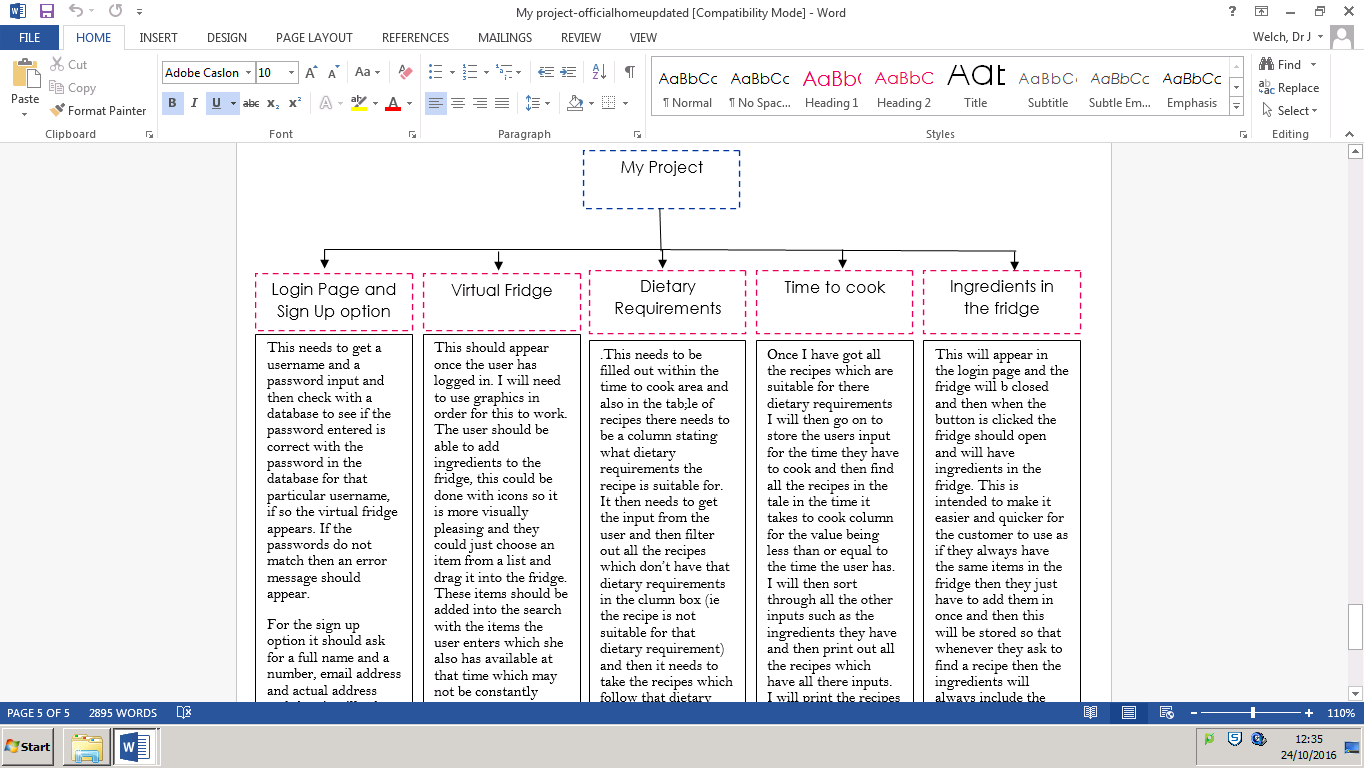
|  |  |
| --- | --- |
| D | I have written a paragraph or two saying **what the project’s about** and maybe why/how I came up with the idea (“the story”) |
| D | I have explained why this problem is **suitable to be computerised** … (computational solution: this should express “the inputs and outputs” of the problem and why a program can produce the outputs from the inputs better than a human can with a piece of paper e.g. calculations, sorting/searching, ) |
| D | I have done some **research on similar** programs/problems and written a paragraph each about their good and bad points. If nothing similar, say so and write about the nearest things you can find. |
| D | I have **referenced** my research fully *e.g.* [1] http://info.com/path/page2.php |
| D | I have drawn some **conclusions** from the research about how I propose to do my project including **justifying** those choices. For example: desktop or client-server app (why?), the sort of GUI (tk or web – why?), which data structures to use and why … etc. |
| D | I have identified some **stakeholders** (users, admin staff, friends) and written about how their life will be better by using my program. |
| D | I have planned some questions to ask them (*e.g.* functions needed, user interface, when and where they need access) and written-up their answers to those polls, questionnaires or **interviews**. |
| D | I have taken some points from the interviews to include in my project and some to ignore or put into ‘future work’ (separate *essential* and *desirable* **success criteria**). My stakeholders have *signed* this page (scan signatures on paper and paste in). |
| D | I have a paragraph subtitled **‘limitations’**. This describes parts of a perfect solution that I will not be including. I might refer to this as abstraction and/or possible future developments e.g. “ideally also a phone app in future”, “won’t work on old browsers”. |
| D | I have a list of **hardware and software requirements**. E.g. I have explained why I chose Python 3 and tkinter (or whatever) and why that was a good choice (which means explaining why other things are not so good a choice). Consider different web browsers and specific *versions* *e.g.* if requiring HTML5. Hardware probably not relevant. |
| D | Finally a list of bullet-points called **success criteria**. A finished, successful solution will …  *See example Database System p12-13 (not 14 – no need for generic PC hardware spec)*  This list will be your evaluation testing so each must be *provable e.g.* “can navigate to X in less than 3 clicks” or “user can login to system”: both can be proved by screenshot |

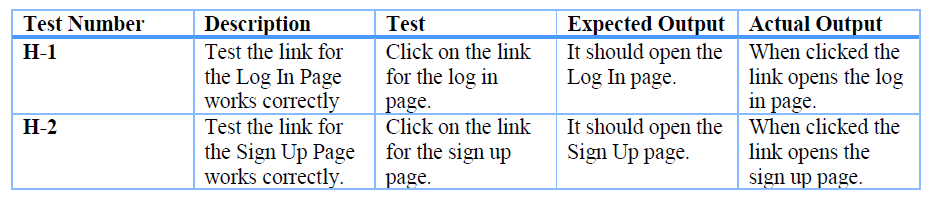
**Design Checklist**

**Hand-in deadline: end of first half-term**

|  |  |
| --- | --- |
| D | I have one or more **project breakdown diagrams** (see below) – the bottom boxes should be something easy to program, ideally a separate subroutine. Separate, or identify separately, the GUI, database, client, server etc. (See Puzzle Website example.) |
| D | I have explained the above process and called it “top down design” suitable for “stepwise refinement”. |
| D | Included the OCR “**process** for success” diagram (suitably cropped) and used the terms iterative development, RAD or Agile method. |
| D | Described the **order** in which you will code the steps *e.g.* build the database, write the algorithms in Python/PHP, add the GUI elements, write the CSS |
| A | I have full **pseudocode** for the bottom boxes (and possibly flowcharts).  I have a flowchart showing how the boxes fit together. |
| D | I have diagrams or tables to show the **data structures** I will use: that is, graphs, trees, simple lists, and any key variables, especially any global variables. |
| D | I may have a diagram showing what code is where for client-server. Which boxes on the breakdown diagram are on client and which on server? |
| D | I have sketched-out designs for the **user interface** – on paper and scanned is fine. |
| D | I have described all the necessary **validation** for all inputs – use keywords like presence check, length check (see Teach ICT for a good list) |
| D | I have written a **test strategy** – usually a paragraph saying you will do alpha testing and fixes while coding then users will beta test and give feedback for you to address before you go on to develop the next bit; include validation tests, use keywords (white/black box, alpha/beta, user acceptance) |
| D | I have a table of **test data** I will use for each module separately based on the pseudocode and validation *e.g.* test all paths (if appropriate), test specific use cases |
| D | I have identified tests in the test plan that are designed to be **destructive** *e.g.* click ‘close window’ without logging out, login then click Back and see if I’m still logged in |

Examples of breakdown and test data:





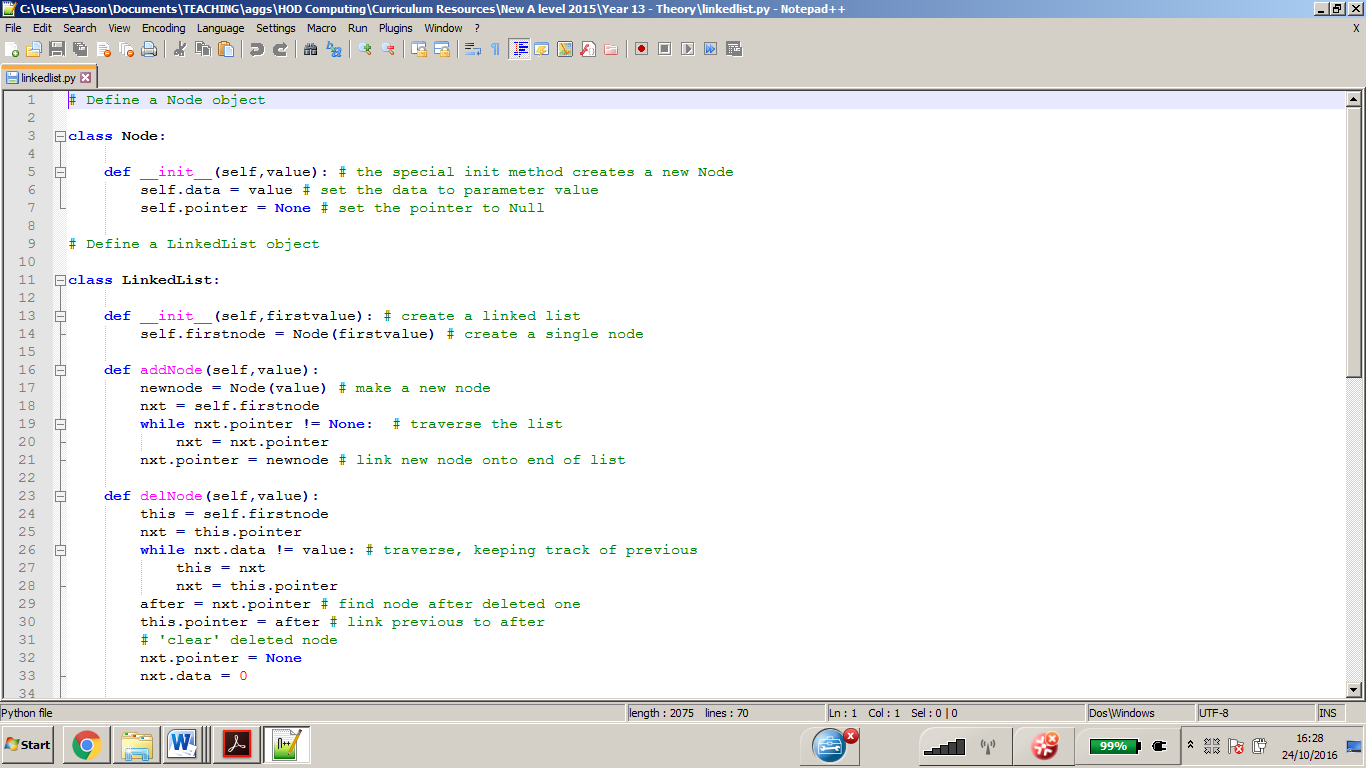
**Development Checklist**

**Code completion deadline: start of Christmas holiday**

**Don’t underestimate the time it will take you to code this: minimum 50 hours outside class**

|  |  |
| --- | --- |
| D | I have written the **code** for each sub-box in my design breakdown |
| D | I have a **full-listing** in a separate file |
|  | I have a paragraph and/or screenshot about each block of code (subroutine or section of main program or web page) which describes in words any obviously **complex** bits so that the marker can easily annotate with ‘Complexity shown here’ |
|  | The code was **tested** for each box in the breakdown according to the test plan and I have written the results of the tests |
|  | I have explained why I changed the test plan, if I did (e.g. changes to the development due to some re-planning) |
|  | I have a paragraph about each block of code which describes any major errors/difficulties and how I overcame them. This includes any changes to the design: fine so long as you say so, and explain why. Highlight your **failures, solutions and changes** – they want to see this! (However, forgotten brackets etc. do not need to be reported or your report will be 100 pages of screenshots.) |
|  | I have used the word “prototype” when completing a block and I have referred to the process for success / RAD / Agile / **iterative development** etc. etc. |
|  | My **users** have seen/tested each block and *signed* to say it roughly does what it’s supposed to as well as giving feedback which I have acted upon |
|  | My code has clearly named **identifiers** for everything |
|  | My code has **structured comments** for each function / block without the comments distracting from the code itself |
|  | Blocks of code that are there for **validation** are highlighted as such e.g. with a comment at the top of the block |

Example (see also GCSE NEA write-up example under KS4):

*<I am writing a simulation of an operating system scheduler.>*

I first developed the LinkedList and Node classes which I will use to implement priority queues for my scheduler simulation.

This code is complex because it uses object-oriented methods and a complex data structure.

Alpha Testing – results of planned tests

|  |  |  |
| --- | --- | --- |
| Create and print a linked list with one item | *<output …>* | Pass |
| Add one more node and print |  | Failed |

I had trouble getting addNode to work initially because I had written nxt = self for the second line giving an error in the third (*object has no such member*). It took me a while to realise that self would be a LinkedList object and what I needed was a pointer to the first node of that object rather than the object itself. Realising this was important as I needed to use the same idea in all the other methods of the class.

*<further detail on testing here and any changes made to design>*

**Evaluation Checklist**

**Evaluation hand-in deadline: start of February HT holiday**

|  |  |
| --- | --- |
|  | I have provided annotated screenshot evidence of black-box **testing** the final version – this will be either repeating them (if changes were made after testing) or just referencing the test evidence earlier in the report |
|  | I have provided annotated screenshot evidence of some ‘**destructive** testing’ to show robustness (should be already identified in the earlier testing so can just reference) |
|  | I have provided annotated screenshot evidence of some ‘**usability** testing’: user has followed some typical usage patterns and found it easy to use; I have described how this was planned for originally |
|  | My user has *signed* off the final version and its evaluation |
|  | For each **success criterion** I have:   * stated whether it is met, partial or unmet * given some evaluation of how well it was met which might include saying how it can be met in future if currently not met (i.e. desirable criteria) |
|  | I have listed potential future **improvements**: either to make a feature useful/nice rather than just-about working; or to extend the number of features |
|  | I have explained the **benefit** of each future improvement proposed |
|  | I have a sub-section on **maintenance**: both fixing any bugs found in the code (how easy have you made it for yourself/another person to do this); and how the user’s requirements might change in future including addressing the limitations in the analysis |
|  | I have a comprehensive **bibliography** with fully-referenced sources |

Example referencing:

Do not repeat yourself. Refer back to earlier pages where the evidence of testing can be found. Make sure in your final-final version that the **page numbering** is right.

*If you do not know how to auto-page number and/or do footnotes/referencing in Word, come and ask.*

Referencing websites: you might reference Stack Overflow like this:

… I was able to research a solution [SO11] and used this method in my showScreen() function …

and then in the bibliography:

[SO11] *How do I use multiple frames in a tkinter program?* <http://www.stackoverflow> … etc.