DAP405 Coursework Summary. Part 1. (60%)

2021-22

Submission deadline: see Moodle.

Requirements

Part 1 involves planning and developing aspects of a program to meet the needs of a particular scenario or problem. Students will need to write a clear specification, develop aspects of the program (build a website) and document this **(1,500 words equivalent).**

Deliverables:

A **report** to be uploaded onto Moodle (no paper copy is required). You need only include relevant code samples in the report.

Your working web site **must** be uploaded uploaded as a .zip file to Moodle.

Your student number must be in the file name of the report.

Please note. 5% will be deducted if your PDF file name does not contain your student number

System logins

If your system requires a user to log in, the details **must** be:

Role	Username	Password
Administrative access	Admin	password
Standard user	User	password
Other users	user1, user2 etc	password

The Data

There are two data files, available on Moodle

- **employees-final.json** – contains data for each employee including annual salary.
- **tax-tables.json** – gives the tax bands for certain salary levels.

The calculations

For **each employee** you must calculate the **take home (i.e. after tax) pay**.

The tax-tables JSON file gives the amount that is tax-free and then incremental bands.

Taxation 'Rules'

- Every employee gets a pay slip detailing gross pay (before tax), net pay (after tax) and tax deducted. Other details like pension contributions are at your discretion.
- Every employee gets taxed according to the tax band their salary is in.
- Most employees earning £10,000 or under, do not pay any tax.

- The first £10,000 of anyone's income is tax free apart from **company car users and those earning over £150,000 pa**
- Anyone earning in currency other than sterling (£ GBP) is paid tax-free on the assumption they pay tax in their home country
    - You can change this and let foreign currency earners pay UK tax by converting their salary into £GBP, deducting tax as normal and converting back into their currency.
- Ignore National Insurance deductions.

Remember! - Income tax works on *graduated bands.* If you pay the higher rate for example, that is only deducted from the portion of salary *above* a certain level, not the whole amount. This makes the calculation more challenging!

Example for someone earning £50,000 pa

Portion of income	% deduction	Remainder taxable
First £10,000	0	£40,000
£10,000.01 to £40,000	20%	£10,000
£40,000+	40%	

Website minimum requirements

To meet the minimum standard to pass, your program must produce a functional PHP-based website that will:

- Read the JSON input file of employee data to import a **list** of employees and their salaries.
- Import tax tables from the JSON file to determine what rate of tax each employee will pay.
- Produce a list of all employees with their **monthly take-home** (net, after tax) pay.
- Calculate take home-page ('net' or after tax) for each employee for the current month.
- Display a list of **all** employees.
- Allow click-through to individual employee's pay slip.
- Have thoroughly commented code.

Marking criteria

Assessment criteria (see ‘further details’ below for a full explanation)

1. Design a program specification that assists with solving a problem; 15%
2. Implement aspects of the specification as a program; 45%
3. Document the program created and discuss how it has met the specification; Outline programming techniques used; 15%
4. Explain how any problems encountered were overcome; 15%
5. Document future developments. 10%

Word limit

Word limit 1,500 equivalent.

	Criteria 1: Weight 15% Design a program specification that assists with solving a problem	Criteria 2: Weight 45% Implement aspects of the specification as a program
80% and above	The specification is of a standard that gives a near-total reflection of how the site will work.	The site looks and functions like a professional production system. Code quality is of a very high standard with a variety of appropriate or innovative techniques used. The user interface and extra functionality all provides an excellent user experience.
70 to 79	The specification accurately reflects the site but with some minor omissions. Accurate and detailed flow chart or another schematic used.	The site is built to a high standard with efficient quality code and a good variety of extra functionality.
60 to 69	The specification reflects most of the site but with some notable omissions. Accurate flow chart or another schematic used.	The site structure, code quality and programming logic is of a high standard. The user experience is good and reflects a ‘real’ system. There are still some improvements that could be made or some functional omissions.
50 to 59	The specification gives a usable overview of the site but may have some ambiguity.	An attempt was made to improve on the minimum requirements and provide some additional functionality and / or usability.
40 to 49	The specification only has reasonable relation to the site.	Minimum system requirements met but code is lacking quality and creativity and user experience is poor.
30 to 39	The specification only has marginal relation to the site.	The site does not work in a meaningful way.
Below 30	There is little or no relevance to the site produced.	The site fails to meet minimum criteria.

	Criteria 3: Weight 15% Document the program created and discuss how it has met the specification; Outline programming techniques used	Criteria 4: Weight 10% Explain how any problems encountered were overcome	Criteria 5: Weight 10% Document future developments
80% and above	Evidence of a thorough understanding of techniques used. Comparison to other programming languages or what could be improved in a more complex environment.	A thorough analytical and technical understanding of a problem(s) and exactly why it is happening in the context of your program. Assessment of more than one way to fix, with evidence supporting each method and thorough understand the solution.	Evidence of an excellent, thorough and relevant understanding of how the website *could* develop in terms of technical changes, interface, user experience, performance and accessibility. Rationale and example technologies included and justified.
70 to 79	Analytically discusses techniques. Lists their pros and cons and other possible ways of achieving the same result.	Thorough understand the problem and solutions although you may not identify all possible solutions.	Evidence of a good understanding of how the website could develop in terms of technical changes, interface, user experience, performance and accessibility.
60 to 69	Analytically discusses techniques used and the reason(s) why they were used.	Correct identification and understanding of problems and a sound explanation of the solution(s).	Evidence of a good understanding of how the website could develop but do not give enough rationale for the additions or example technologies.
50 to 59	Accurate but limited description of techniques used and little evidence of thorough understanding.	Identification of some issues but lacking understanding of what is causing the problem.	Documenting some useful future developments with reasonable explanation of the reason(s).

40 to 49	Basic description of techniques used but little evidence of understanding with some mistakes or omissions.	Basic understanding of some of the problems and while a solution may have applied there is little evidence of a clear understanding of either or both.	Documenting some future developments but do not adequately explain the reason(s) and how this would be implemented technically or what issues may arise.
30 to 39	Minimal overview of techniques used or with errors.	Identification of few problems and either do not explain them well or misunderstand them.	Minimal understanding of what could be added and how this would add value.
Below 30	Weak, missing or inaccurate overview of techniques used.	Failure to identify problems or solutions.	Little or no attempt made to document any future developments.

Further details:

Design a program specification that assists with solving a problem;

The 'problem' is the task at hand – building a payroll system.

Using a flow chart and text, describe how your program works. This does not have to be a specification from which a developer could code, but it must clearly show how the program logic works.

Implement aspects of the specification as a program;

This is the bulk of the work and is the website / program itself.

Please see the minimum requirements above.

The brief allows for many ad-ons to be programmed to increase your mark. In general marks will be awarded for:

- Creative thinking about the solution in terms of code and the user experience.
- Use of efficient code (functions, includes, exception handling etc).
- Use of a variety of techniques, appropriately used.
- The delivery of a usable system.
- The correct presentation of information (e.g. correctly formatted currency symbols).
- Enhanced functionality.

An effective user interface is required but additional marks are award for functionality, not visual design.

Additional functionality is at **your discretion** but *could* include:

- Generation of PDF
- Email notification (be careful!)
- Password protection

- Image manipulation (user profile photo)
- Write out pay slips or monthly pay to .csv or another JSON file
- Error check JSON input
- Upload the JSON file before processing
- Allow user to alter their salary to see what their take-home pay would be
- Deduct pension contributions
- Show pay-slip history (define financial year start)
- Bootstrap ‘popovers’
- Order the list by more than one column
- Pagination (1|2|3) etc
- Work out each employee’s pay after annual pay rise of x% on 1 Aug (for example).

Other additions are at your discretion but must be relevant to the system and add value.

Outline programming techniques used

Detail the nature of your programming. Explain any particular functions or techniques used. You may wish to detail anything you have *not* been able to use due to the particular PHP configuration.

Explain how any problems encountered were overcome

Show an understanding of the problems you encountered, your diagnosis and analysis of the problem and how it was rectified. How many other ways did you consider? What was your approach?

This means coding or logic problems. Please do not list lack of time, problems with hardware and software or inexperience in programming etc as a problem. These will not score any marks.

Document future developments

Not a wish list, but a **meaningful** extrapolation of your site in terms of what would or could be effective value-adding enhancements. You may wish to think about how this small system could scale to a large organisation. Ideas must be supported with some level of understanding of how it can be achieved. Unsupported statements such as "Data encryption could be added" will not gain marks.