Unit 8: Mobile Apps Development

Level: **1 and 2**

Unit type: **Optional specialist**

Guided learning hours: **30**

Assessment type: **Internal**

Unit introduction

How many people do you know who have smartphones or mobile tablet devices? This means that they are carrying considerable computer power around with them. There has been an explosion of software applications, known as apps, to use on these devices. You can use apps for many different purposes; for example, a location app helps you to find your nearest shop, and a leisure app makes it easy to download your favourite music.

Software developers and engineers have scrambled to meet the demand for mobile apps, that are increasingly being used by businesses and organisations. The market for Apple, Android and other apps have boomed. Software engineers are involved with the design, development, testing and maintenance of apps. In addition, software businesses that develop apps employ other professionals, including creative designers, artists and sound engineers.

In this unit you will investigate the characteristics and uses of mobile apps, and learn how mobile apps are developed. Then you will design, develop, test and review your own mobile app. Rather than producing large amounts of original code from scratch, the emphasis in this unit is on you integrating predefined programs/code snippets (specific instructions for a mobile computer) with ready-made and original assets (e.g. buttons and sounds) by using some original code. This will save you significant amounts of time when developing your mobile app. You will review your finished app, having obtained feedback from others, and evaluate possible improvements.

You may even be able to get it published on the internet and take pride in people using an app you made.

In particular this unit develops skills from *Unit 1: The Online World* and *Unit 2: Technology Systems*. In addition, this unit develops the skills from the following specialist units: *Unit 4: Creating Digital Animation*, *Unit 5: Creating Digital Audio* and *Unit 6: Creating Digital Graphics*. It also complements the delivery of *Unit 12: Software Development*.

Learning aims

In this unit you will:

A understand the characteristics and uses of mobile apps

B design a mobile app

C develop and test a mobile app

D review the finished mobile app.

Learning aims and unit content

What needs to be learnt
Learning aim A: Understand the characteristics and uses of mobile apps
The purpose of mobile apps Mobile apps are computer programs that instruct a computer's Central Processing Unit (CPU) to carry out the set of specific instructions given in a program for a specific reason and use. **Typical uses of mobile apps** Know why we develop mobile apps and know typical uses, including: ● to give information (e.g. BBC, photo, video, music) ● for navigation (in the physical world) (e.g. location identification, nearest tube stations, sandwich shop) ● for entertainment (e.g. YouTube, Spotify) ● for leisure and fitness (e.g. tracking fitness, RunKeeper) ● for communication (e.g. Skype mobile, Live Messenger, Fone Time) ● for augmented reality (e.g. Layar, Junaio). **Features of mobile apps** Key features and characteristics, e.g.: ● purpose of the app ● user requirements ● user-friendliness (e.g. what are the features of the interface/screens that are presented to the user? How does the user communicate with the app and make things happen?) ● dependence on particular hardware ● interface elements ● integration with standard operating-system software (e.g. contacts list, text messaging) ● platforms and compatibility. **Programming mobile apps** Know there are types of programming language; including C++, Java and XML. Understand the reasons for compiling programs.

What needs to be learnt

Learning aim B: Design a mobile app

Software development life cycle

Software development life cycle, including:

- requirements of the problem
- design specification (i.e. scope, inputs, outputs, processing, user interface)
- constraints (i.e. programming language and timescales for development)
- develop code
- test
- maintain code.

Designing a mobile app

Design to include:

- purpose
- user requirements or problem to solve.
- a proposed solution using design tools, e.g.:
  - a description of the main program tasks – input and output format (e.g. to add two numbers together and display a result)
  - screen layouts and navigation including prototypes (initial splashscreen, main activity screen, other screens or screen elements)
  - algorithms with a description of the method of solution and processing structure (flow charts, pseudocode and events)
  - control structures
  - data validation
- a brief outline of alternative solutions for the intended app (e.g. screen layouts and navigation)
- a list of any pre-defined programs, or code snippets to be used
- ready-made and original assets
  - video, graphics, audio and animation, e.g. sprites, sounds, images, movies, animations and buttons that will be integrated into the app (these are available on the internet and other media, such as CD or DVD)
  - all sources for pre-defined programs and ready-made assets documented and referenced
- test plan with test data (e.g. testing the inputs and expected outputs and compilation of the code)
- constraints (e.g. device capabilities, such as connectivity, screen size, memory storage or programming language).

What needs to be learnt

Learning aim C: Develop and test a mobile app

Preparing content to develop an app

- Prepare and gather pre-defined programs, snippets and/or subroutines, and ready-made and original assets.
- Edit (using appropriate editing software) and optimise assets for a mobile platform (e.g. sacrificing quality for smaller file size).
- Use file formats that are appropriate for the intended platform.

Develop and refine an app

Use a development environment to write the code for a mobile app.

Integrate ready-made programs, code snippets and assets with some original code.

Use suitable program constructs to edit and create code:

- command words, e.g.:
  - o comments
  - o constants (variables with a constant value that cannot change)
  - o operators; arithmetic (+, -, *, /, %) and logical (<, <=, >, >=, AND, OR, true, false)
  - o reserved words that have special meaning within the programming language and are used to write instructions in a program (e.g. in Java 'const' and 'goto' are reserved words)
  - o input and output commands
  - o local variables (variables that only exist inside the subroutine/function where they are declared and used)
  - o global variables (variables that exist throughout the entire program and in subroutines/functions)
  - o assignment
  - o loops, (counter-controlled, conditional, iteration, [while do, repeat…until, for…to do])
  - o sequential statements, selections (If… then…).
- Subroutines/functions/procedures (e.g. reading in data, printing out information).
- a range of data types, e.g.:
  - o character
  - o string (text)
  - o integer and real (numbers)
  - o Boolean.
- basic string handling commands to examine individual characters and substrings.

continued

What needs to be learnt

- Event handling:
  - o forms
  - o assigning properties to screen components (e.g. buttons, boxes, data validation and drop-down lists)
  - o actions.

Annotate code to demonstrate understanding and to allow effective repair/debugging of the program.

When required, compile the program into a suitable format to create an executable program.

Quality of software programs

Know that software design and techniques affect the quality of the app developed:

- efficiency/performance, e.g. the amount of system resources a program consumes (processor time, memory space, accessing storage media)
- maintainability, i.e. the ease with which a program can be modified by its present or future developers in order to carry out corrective, perfective or adaptive alterations to the code
- portability, i.e. the range of computer hardware and operating system platforms on which the source code of a program can be run/compiled/interpreted
- usability, i.e. the ease with which an end user can use the program for its intended purpose or, in some cases, even unanticipated purposes.

Test the app

Test the program solution:

- for functionality against the test plan with the test data
- is fit for purpose
- by reviewing the quality of the program in terms of efficiency/performance, maintainability, portability and usability
- gather feedback from others on the quality (efficiency/performance, maintainability, portability and usability) of the solution.

Document any changes to the design, including changes to the source table for pre-defined programs/snippets and ready-made assets.

Improve or refine the app (e.g. efficiency/performance, maintainability, portability, usability).

What needs to be learnt
Learning aim D: Review the finished mobile app
Review the app Review the finished mobile app for: ● user requirements ● fitness for purpose ● constraints (e.g. programming language, time, copyright, device capabilities – connectivity and screen size) ● quality of the program (e.g. efficiency/performance, maintainability, portability, usability) ● strengths and improvements.

Assessment criteria

Level 1	Level 2 Pass	Level 2 Merit	Level 2 Distinction
Learning aim A: Understand the characteristics and uses of mobile apps			
1A.1 Identify the uses and features of two different apps.	2A.P1 Explain the uses and features of two different apps.	2A.M1 Review how the features of the apps affect the usability and intended use by the audience.	2A.D1 Discuss the strengths and weaknesses of the apps.
Learning aim B: Design a mobile app			
1B.2 Identify the purpose and user requirements for the app.	2B.P2 Describe the purpose and user requirements for the app.	2B.M2 Produce a detailed design for a mobile app, including: • alternative solutions • a detailed proposed solution using a range of design tools • test data.#	2B.D2 Justify the design decisions, including: • how they will fulfil the purpose and the user requirements • any design constraints.#
1B.3 Produce a design for a mobile app with guidance, including an outline of the proposed solution.	2B.P3 Produce a design for a mobile app, including: • a proposed solution • a list of any pre-defined codes/programs • a test plan • a list of sources for any pre-defined code and assets.#		

Level 1	Level 2 Pass	Level 2 Merit	Level 2 Distinction
Learning aim C: Develop and test a mobile app			
1C.4 Prepare predefined code and assets with guidance.	2C.P4 Prepare predefined code snippets and assets for the app, demonstrating awareness of purpose, listing sources for assets used.	2C.M3 Optimise assets for the app, demonstrating good awareness of the user requirements, with all sources for assets fully referenced.	2C.D3 Refine the app, taking account of the quality of the code and user feedback.*
1C.5 Edit predefined code and integrate with assets to develop an app, with guidance, containing: ● one or more screens ● constructs.*	2C.P5 Edit predefined code and integrate with assets to develop an app which includes: ● one or more screens ● constructs ● commentary throughout the code.*	2C.M4 Develop a functional multi-screen app containing original code, that meets the user requirements and purpose.*	
1C.6 Test the app for functionality and purpose, repairing any faults and documenting any changes made, with guidance.	2C.P6 Test the app for functionality and purpose, repairing any faults and documenting any changes made.	2C.M5 Gather feedback from others on the usability of the app, and use it to improve the app, demonstrating awareness of audience and purpose.	

Level 1	Level 2 Pass	Level 2 Merit	Level 2 Distinction
Learning aim D: Review the finished mobile app			
1D.7 For the final app, identify how the final app is suitable for the user requirements and purpose.	2D.P7 For the final app, explain how the final app is suitable for the user requirements and purpose.	2D.M6 Review the extent to which the final app meets the user requirements and purpose, considering feedback from others and any constraints.	2D.D4 Evaluate the final app and the initial designs and justify any changes made to the quality of the code, making recommendations for further improvement.

*Opportunity to assess mathematical skills

#Opportunity to assess English skills

Teacher guidance

Resources

The special resources required for this unit are:

- a software development kit for a mobile device programming language, e.g. Android App Inventor
- an onscreen emulator for a mobile device
- example mobile device(s) on which to run apps
- graphic and audio-editing software to edit and optimise ready-made and original assets.

Teachers may wish to use an application such as Scratch to introduce learners to the concepts of programming. Scratch is available free from http://scratch.mit.edu/download.

Learners should be provided with a brief to design and develop an app, or decide on their own user requirements and purpose for the app.

Assessment guidance

This unit is assessed internally by the centre and externally verified by Pearson. Please read this guidance in conjunction with *Section 8 Internal assessment*.

Learning aim A

In this learning aim, learners will investigate the characteristics of mobile apps and how they are used.

For 2A.P1: learners should consider a range of existing mobile apps. It may be beneficial to offer a selection of apps, preferably ones with a clear purpose and audience, from which they can choose two. Learners should explain the features and intended use of each app. The two apps should be designed for different purposes.

For level 1, as a minimum, learners should be able to identify the purpose and some of the features of at least two existing mobile apps, including presentation interface elements and compatibility.

For 2A.M1: learners should review how the features of the two apps affect the intended use, usability and appeal to the audience.

For 2A.D1: learners should look at one app in more detail and consider the strengths and weaknesses of the product. They should discuss at least one strength and one weakness.

Learning aim B

Learners are not expected to find their own problems or create their own project brief. Suitable scenarios should allow learners to achieve all assessment criteria. The user requirements should be given in the brief:

- purpose of the software program
- task(s) the software must perform
- a list of the required user inputs and outputs
- an outline of any processing/functions required.

Centres are encouraged to use evidence for the development of the software program as part of the learner's digital portfolio (Unit 3). For instance, a movie showing snapshots throughout the development process would be appropriate, as would an audio diary of the process, or blog entries as developments are made.

For 2B.P2: learners should describe the user requirements and purpose of the app for their design, as well as the user requirements for their design.

For level 1, as a minimum, learners should identify the purpose of the app and the user requirements for the design.

For 2B.P3: for a given problem outlined in a brief, learners should design their proposed solution. The design documents should include:

- a proposed solution using basic design tools, including a description of the main program tasks (data input and output, screen layouts and navigation, and descriptions of the method of solution)
- a list of any pre-defined programs/code snippets (including any functions or sub-routines) and assets, documenting the sources appropriately
- a test plan (to test for the logic and functionality).

Learners will produce design ideas for apps. Please note that learners do not have to create original assets unless they choose to do so. The original assets may have been produced in a unit such as *Unit 5: Creating Digital Audio* or *Unit 6: Creating Digital Graphics.*

For level 1, as a minimum, learners should suggest an outline of a proposed solution which will contain:

- *a description of the main program tasks – input and output (e.g. to add two numbers together and display a result)*
- *screen layouts (input and output) – templates or design sheets can be used to help learners with their design.*

For 2B.M2: in addition to the requirements for the pass grade, learners should produce:

- a detailed proposed solution, using a range of suitable tools (in addition to those used at pass) such as flowcharts, control structures pseudocode, events, data handling, and error handling and reporting)
- a brief outline of any alternative solutions for the intended software program,
- test data.

For 2B.D2: at this level, learners are expected to be able to justify their design decisions and how the chosen design fulfils the purpose and user requirements. They should consider the suitability for end users and the quality and thoroughness of their design work. They also need to review their design in light of any constraints (e.g. screen size) arising from the device and the programming language used. Learners should explain why alternative designs were rejected.

Learning aim C

The designs will be used to create the mobile app. Although learners may deviate from their designs (as happens with any project), they should aim to develop final products that closely resemble their original design. The teacher should recognise that the activities of gathering and preparing code and assets, along with original code, is an iterative process.

For 2C.P4: learners should prepare (including gathering) predefined code and ready-made assets, such as buttons and images, and list them in a table of sources (please note, many assets will be included within the development environment, e.g. buttons). Chosen assets should demonstrate awareness of user requirements and purpose. The sources of assets should be listed.

For level 1, as a minimum, learners should prepare ready-made assets required for the app. This may include sprites, sounds, images, movies, animation and buttons from a variety of sources.

For 2C.M3: learners should optimise ready-made assets. For instance, bitmap images should be optimised (e.g. be an appropriate file type and size to increase the responsiveness of the app). Learners should demonstrate good awareness of audience and purpose. All predefined code and assets should be fully listed in a sources table, which should be detailed enough for another person to independently obtain the assets used.

For 2C.P5: learners should integrate the pre-prepared code snippets and assets by editing the code. They should then develop the app by:

- creating an interface which demonstrates an awareness of the user requirements and purpose of the app
- assigning code to assets, e.g. buttons to control behaviour
- writing comments within the code to explain how it works.

The interface may be a single screen with a number of assets that cause an event to happen.

For level 1, as a minimum, learners should integrate and edit the assets and code to develop the app. Their app should contain one or more screens and simple constructs.

For 2C.M4: learners should edit defined code and develop some original code to fulfil the design requirements of the app. The development process will include creating a multi-screen interface that reflects the planned interface, with assets on each screen. The app should be multifunctional.

For 2C.P6: learners should test the functionality of code, ensuring it is fit for purpose and adjust the code as required to fix any problems. They should document any changes to the program. Testing documentation might range from a simple checklist to a more elaborate testing schedule that includes due dates for completion of different parts of the project.

For level 1, as a minimum, learners should test the app for functionality and fitness for purpose. They should fix any faults and document their changes.

For 2C.M5: learners should gather feedback from others when testing the app, considering the user requirements and purpose of the app, and use it to improve the app.

For 2C.D3: teachers should recognise that the activities of developing and testing computer programs is iterative process and not sequential. Consequently, the Distinction criteria for this learning aim is assessed through using the learners' work from both the Pass and Merit criteria.

Learners should refine their mobile app, taking account of user feedback, where appropriate to do so, and the quality of the code, e.g. maintainability (how easily the code can be modified), portability (on different platforms) and usability. All of the ideas from testing, feedback and improving their designs as they create the apps should have been considered as how best to refine the product.

Learning aim D

For 2D.P7: for the final review, learners should be able to explain why their app is suitable for the user requirements and purpose. Learners should give one reason for audience and one for purpose.

For level 1, as a minimum, learners should identify why their app is suitable for the user requirements and purpose. This could be achieved through a discussion with the teacher about the outcomes of their project and evidence with a witness statement and observation record.

For 2D.M6: for the final review, learners should review their app with others, discussing the extent to which their solution meets the needs of the original requirements and purpose of the app. Learners should consider how constraints, user feedback and testing has affected the suitability of the app.

For 2D.D4: at this level, learners should evaluate their initial designs and the completed app. They should identify any changes made from the design stage and justify these changes.

Learners should make at least three specific suggestions for improvement for the completed program to ensure it is fully functional, well coded and fit for purpose, including considerations of any constraints, user requirements and purpose.

Learners do not need to implement the enhancements.

Programming constructs and techniques for level 1 assessment

It is recognised that some learners may fail to achieve a full Pass at level 2. Learners being assessed for the level 1 criteria for learning aims B and C are therefore not required to include all of the different programming constructs in their work for assessment.

The constructs that learners working at level 1 should be familiar with and include in their assessment evidence are shown below.

Use program constructs e.g.:

- command words:
  - comments
  - constants (variables with a constant value that cannot change)
  - arithmetic operators (+, -)
  - reserved words which have special meaning within the programming language and are used to write instructions in a program e.g. in Java ‘const’ and ‘goto’ are reserved words
  - local variables – only exist inside the subroutine/function where they are declared and used
  - global variables – exist throughout the entire program and in subroutines/functions
  - assignment
  - counter controlled loops.
- a range of data types, e.g.:
  - string (text)
  - integer and real (numbers)

- event handling:
  - forms
  - assigning properties to screen components (e.g. buttons, boxes and drop down lists)
  - actions.

Suggested assignment outlines

The table below shows a programme of suggested assignment outlines that cover the assessment criteria. This is guidance and it is recommended that centres either write their own assignments or adapt any assignments we provide to meet local needs and resources.

Criteria covered	Assignment	Scenario	Assessment evidence
1A.1 2A.P1, 2A.M1, 2A.D1	Reviewing Apps	You work for a publishing company that is moving into increasing its digital publishing for handheld devices and smartphones. Your manager is going to ask you to lead on an app design project, and to investigate educational mobile apps currently available. Your manager wants to know how mobile apps could be used to: • support people with their learning and development • create a guidebook for a major city. They have asked you to prepare a presentation that describes the purpose, quality and features of both these apps.	• Presentation.
1B2, 1B.3 2B.P2, 2B.M2, 2B.P3, 2B.D2	Design an Interactive Alphabet App	A leading educational publisher has asked you to develop an interactive app. The app is intended to help children learn their alphabet. Learners should design an interactive app which will include pre-defined and edited code with assets. You should describe how your design meets the user requirements and purpose. Designs should include: • list of assets or code • proposed solution • test plan • alternative ideas for designs	• Processing structures, e.g. flowcharts and structure diagrams. • List of tools, functions and subroutines. • Report. • Screens (input and output).

Criteria covered	Assignment	Scenario	Assessment evidence
1C.4, 1C.5, 1C.6 2C.P4, 2C.M3, 2C.P5, 2C.M4, 2C.P6, 2C.M5, 2C.D3	Developing and Testing	You should now have everything you need to develop your interactive learning app. 1. Gather, prepare and optimise predefined programs/code snippets and assets (e.g. images, sound, video) that you will use in the interactive app. 2. Reference sources in a table. 3. Develop the interactive app, using the predefined programs/code that you have gathered. Edit pre-defined code and any original code to create the app. Do not forget to include commentary within the code. 4. Test the app against the test plan, checking the user requirements are still being met, and document any changes to the app. Correct any errors in the app. 5. Get feedback from one other person on the app, including how easy it is to use and the quality of the code. 6. Review and refine your app in light of feedback.	• Annotated code. • Annotated design. • User feedback. • Functional apps. • Test plans and data. • Source table for assets.
1D.7 2D.P7, 2D.M6, 2D.D4	Review the App	How could the app be improved? Consider strengths and points you could improve in your design. How suitable is your app for the audience and purpose? Does it meet the user requirements? Justify where your design has changed through the development, including what has changed following feedback, and explain how you would improve the program further.	• Annotated code. • Annotated design. • User feedback. • Evaluation report.

