6COSSC004W Mobile Native Development

Coursework 1 – iPhone Application (2021/22)

Weighting:	**4**0%
Qualifying mark	30%
Description	**iPhone Finance App (UIKit)**
Learning Outcomes Covered in this Assignment:	This assignment contributes towards the following Learning Outcomes (LOs): LO1 understand language features and programming practice required for native development LO2 apply industry standard tools for design and development LO3 communicate and defend work by both written and oral means
Handed Out:	~~8th February 2022~~ 2nd Mar 2022
Due Date	12th Apr 2022 ~~22nd March 2022~~ Submissions via Blackboard by 1pm
Expected deliverables	Submit on Blackboard a zip file containing: **Complete XCode 12.X solution - Swift language only**
Method of Submission:	Electronic submission on BB via a provided link close to the submission time. The file you upload should have the following naming format: E.g. **6COSC004W_CW1**_StudentNumber_firstName_lastName.zip
Type of Feedback and Due Date:	Formative feedback will be provided during tutorial sessions. Verbal feedback on the submitted CW will be provided during the CW presentation/viva. Students are encouraged to record this feedback at this time. Feedback is due by the ~~5/4/22.~~ Feedback shall also be given on the Blackboard. 26th Apr 2022 Note: All marks will remain provisional until formally agreed by an Assessment Board.

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, **what constitutes plagiarism etc.**

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website :http://www.westminster.ac.uk/study/current-students/resources/academic-regulations.

6COSC004W Mobile Native Development Coursework 1 - Specification

iPhone Financial Calculator App

This coursework is worth 40% of the module mark

Please read **carefully** and read all footnotes. Raise any issues about this coursework early with your lecturer/tutor.

Synopsis

You are to create a financial app for the iPhone. It must render well on all iPhones from the 8 to the iPhone 13. The app shall have the capability of undertaking typical financial calculations such as savings, loans, and mortgages. The app shall have the ability to solve for **one** unknown from the given parameters noted in **bold**:

t – time in years (synonymous with number of payments)
r (%) – interest rate – for simple savings only[1]
P – present value
PMT – Payment
A – future value

[1] You only need to solve for interest rate in problems where there is no monthly payments. For example, simple lump sum investments.

PayPY – number of payments per year (always assumed to be 12)
CpY – number of compound payments per year (always assumed to be 12)
PmtAt – payment due at the beginning or end of each period (assumed to be END)

Table 1 - Financial parameters (see examples below in the requirement notes)

The app shall split typical financial problems up over typically four views:

1) Compound Interest savings (fixed sum investment with no further payments)
2) Savings – compound interest with regular contributions (this is savings where there might be sum invested with a subsequent further monthly contribution
3) Loans/Mortgage - compound interest with regular payments

In addition to this the software **shall contain a help view** that will contain instructions and guidance to the user on how to use the software. You have complete freedom on how to implement this view and this can be done as separate view or modal context views for example, e.g. a pop- overview activated by a help button.

User Requirements

R1 The software shall allow the user to estimate **interest rate** based on other financial data given in Table 1 - Financial parameters.
R2 The software shall allow the user to estimate **final value** based on other financial data given in Table 1 - Financial parameters.
R3 The software shall allow the user to estimate **present value** based on other financial data given in Table 1 - Financial parameters.
R4 The software shall allow the user to estimate **the payment** based on other financial data given in Table 1 - Financial parameters.
R5 The software shall allow the user to estimate **number of payments**[2] based on other financial data given in Table 1 - Financial parameters.
R6 The software shall persistently save all user data[3]
R7 The software shall provide a help view.
R8 The software shall allow the user to switch between number of payments and years[4].

System Requirements

SR1: When sufficient data is entered in any one view that software shall populate the empty field.
SR2: The system shall persist user data if the application close or quits.
SR3: The system shall repopulate fields with any saved user data on app start-up.

Notes:

[2] For the purpose of this application all payments and compound interest and be considered to be monthly. So, for example 60 payments is equivalent to 5 years.
[3] If the app is backgrounded or closed by the user or runtime, all user data will be preserved persistently and the entry text fields repopulated with the last data used when the app is started or foregrounded.
[4] This assumes 12 monthly payments per year.

Typical user scenarios and examples:

Savings

A user wishes to calculate the interest required to return a **future value** for a fixed initial investment sum over a known period of time.

E.g. A user wishes to know if they invest £1000 for 5 years, what interest rate would be required to return a **future value** of £3000.

A user wishes to calculate the length of time in years (**or number of compound payments**[5]) required to return a future value for a known fixed initial investment given a known interest rate.

E.g. A user wished to know how long it would be before an investment of £1000 at an interest rate of 4% returned a **future value** of £3000.

A user wishes to calculate the initial investment (**present value**) that will return a known future value given a known interest rate and investment period (time).
E.g. A user wished to know how much to initially invest to get a **future value** of £3000 on a savings account with an interest rate of 4% over a 5-year period.

E.g. A user wished to know how much to initially invest to get a **future value** of £5000 on a savings account with an interest rate of 4% over a 5-year period where they make monthly payments into their savings account of £100.

Loans and Mortgages

A user wished to borrow £3000, they can afford to pay £350 a month, how many monthly payments (remember this is the same as time) would they need to make if the best loan interest rate they could get was 7%?

A user wished to borrow £350000, the best interest rate available is 3.5%, what would be the required monthly payment over 25 years?

UI Requirements

Some possible designs for the app are shown below in figures 1-3. Note that you are free to design this app as you wish but it must meet all requirements and constraints. Note that the optimum design is to have views for: 1) Mortgage/Loans 2) Compound Savings with regular contributions, 3) Lump sum savings – no regular contribution.

[5] Compounds are monthly payments from the interest on a on savings or are the monthly interest charges on a loan.

Savings

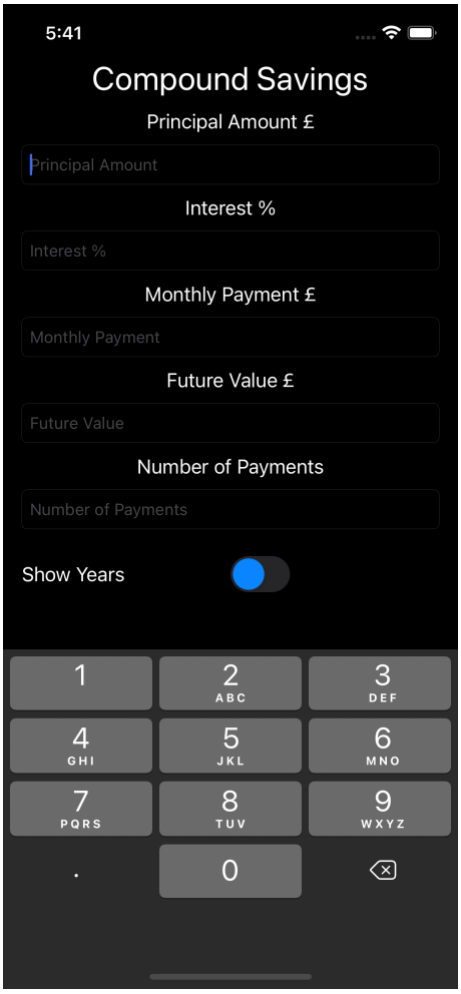

Figure 1 Example design for compound savings with monthly payments

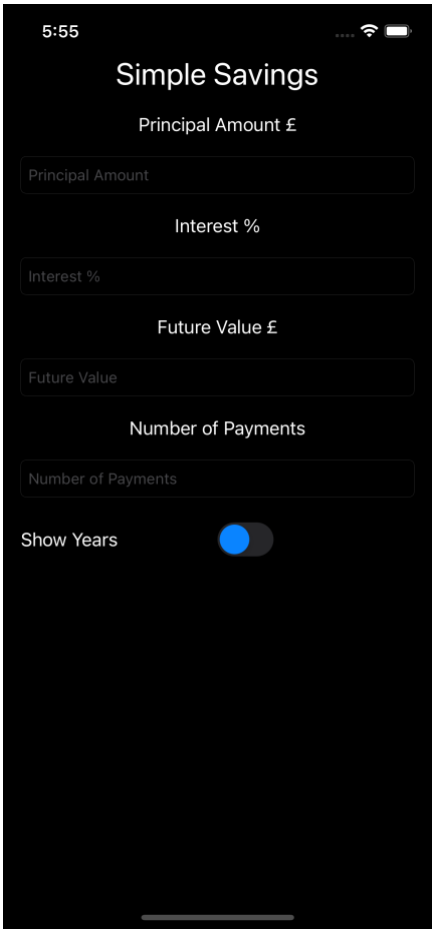

Figure 2 Example design for simple savings view

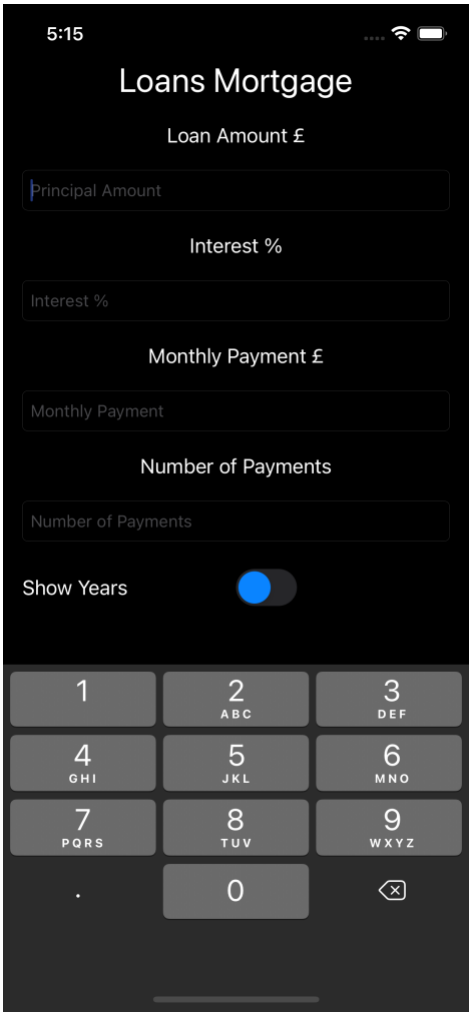

Figure 3 Example design for Mortgage/Loans

Figure 4 Mock up view that shows tab bar and icons with decimal pad keyboard

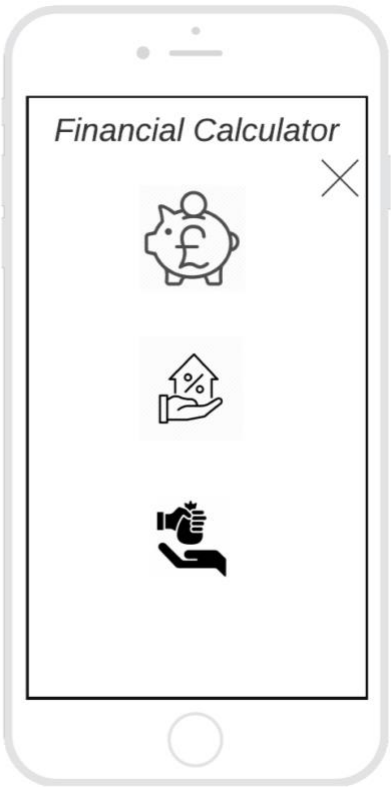

Figure 5 Alternative Icon based start view

Help View

The app shall contain a help view with simple instructions and optional images to help guide the user.

Operation

To use the application the user first selects one of the tabbed views from an array of icon buttons (either tabs or a toolbar) that can be accessed in all views. The field in which the user wishes to estimate on of the parameters should be left blank with the user filling in all other details. Note that you can use a button that begins the calculation, though 'autodetecting' when the calculation is possible is better and will receive higher marks in that section.

Keyboard Requirements

The app can use the standard decimal pad keyboard but not there is no negative button so you will need to make sure that this is handled in the calculation. For example, any payment going out from the saver/borrower is negative in the formula shown in the appendix. You are also free to use a custom keyboard.

Non-functional requirements

1) The app shall render well in iPhones from 10 to iPhone 13 series
2) The strings must be consistently formatted and correctly spelt.
3) Currency must be correctly formatted to two decimal places e.g. £345.21.
4) All financial calculations shall be mathematically correct to two decimal places.
5) Code shall be commented to assist marking and to assist understandability

The Coursework will be marked based on the following indicative marking criteria [6]

Criteria	Maximum component mark
Layout and Design, UI, Navigation and responsiveness Renders well in all target devices, good use of icons and UI element placement. App is simple to use, and all UI elements are clear to their function marked and is app is intuitive to use. Strings are correctly formatted. Help notes are available but unobtrusive. Note that apps with poor usability may score low in the section of the marking rubric.	25
Interest calculation (R1)	5
Future Value calculation (R2)	10
Present Value calculation (R3)	10
Time Calculations (R4)	10
Payment calculation (R5)	10
Help view(s) (R7)	10
Data persistence (SR2/SR3)	10
Switch between years/number of payments	5

[6] Note that you need to present your work in a viva after the due-by-date for marking during, in which you will need to demonstrate your understanding of your code – **note your mark may be reduced if you do not demonstrate good understanding and this is at the discretion of the marking tutor**. Note also that best marks are awarded for efficient and well-written code (good coding standards) and also for the correctness of each section (meets the specified requirement exactly). Note that the viva is **compulsory** element of the coursework and you may receive a zero mark if you do not attend a viva.

Coding standards, good separation of concerns. Flexible and maintainable code.	5
Total	100

Note for high marks your work must also meet the non-functional requirements.

Appendix A - Financial Formula[7]

Note there are many formulae available for these calculations are you are free to use any you wish.

Compound Interest – Simple savings with lump sum (no payments)

$$A = P\left(1 + \frac{r}{n}\right)^{nt}$$

Where:

A = the future value of the investment/loan, including interest
P = the principal investment amount (the initial deposit or loan amount – present value)
r = the annual interest rate (e.g. 3.2% is 0.032)
n = the number of times that interest is compounded per unit time (this is always monthly for the purpose of this coursework, i.e. **12** per year)
t = the time the money is invested or borrowed in **years**

Interest Rate

$$r = n\left[\left(\frac{A}{P}\right)^{\frac{1}{nt}} - 1\right]$$

Principle Amount

$$P = \frac{A}{\left(1 + \frac{r}{n}\right)^{nt}}$$

Formula for time

$$t = \frac{ln\left(\frac{A}{P}\right)}{n\left[ln\left(1 + \frac{r}{n}\right)\right]}$$

Note: that ln is the natural logarithm (this is **log()** in Swift)

Compound interest formula (with regular payment or contributions)

These equations assume frequency of contribution and compounding is the same (for example every month)

[7] Create and test all formula by using Swift Playgrounds before implementing these into the iOS UIKit application.

Compound interest formula (with regular contributions) for deposits made at the <u>end</u> of the period - (important note: any money given so going away from lender/saver e.g. a principle amount or payments is a negative number)

Note that the Total A = **[Compound interest for principal] + [Future value of a series]**

Compound interest for a principal amount

$$\boldsymbol{P\left(1+\frac{r}{n}\right)^{nt}}$$

Future value

$$A = PMT \times \left\{ \frac{\left[\left(1+\frac{r}{n}\right)^{nt}-1\right]}{\frac{r}{n}} \right\}$$

Payment

$$PMT = \frac{A}{\left\{ \frac{\left[\left(1+\frac{r}{n}\right)^{nt}-1\right]}{\frac{r}{n}} \right\}}$$

Time to achieve a certain future value A – these assume annual compounds

The following formula will calculate the **time taken in years** to **reach an investment goal A loan.**

$$t = \frac{\ln(1+\frac{rA}{PMT})}{\ln(1+r)}$$

The following formula will calculate the **time taken in years** to **completely pay a loan** – can be useful to express this also in number of monthly payments to a user also (i.e. t x 12). P is the loan amount.

$$t = \frac{\ln(1-\frac{rP}{PMT})}{\ln(1+r)}$$

Mortgage Payments (or any loan) – Note that the future value is assumed to be equal to zero for mortgage and loan calculations

$$PMT = \frac{A\frac{r}{12}\left(1+\frac{r}{12}\right)^{12t}}{(1+\frac{r}{12})^{12t}-1}$$

Mortgage Length of time (t) in years (same as loan)

$$t = \ln\frac{(1-\frac{rA}{PMT})}{\ln(1+r)\times 12}$$

Note: always check maths for correctness and test algorithms in isolation. Playgrounds in XCode is highly recommended for developing and testing the maths before deploying to an iOS app.

Appendix B - Grade Descriptors

Marking Grade Descriptors - for guidance only

80-100: An outstanding piece of work: All assessment criteria have been met at an exceptionally high standard

- Displays exceptional initiative, creativity, sophistication and originality.
- Demonstrates originality and rigour in technique
- Demonstrates complete and fit for purpose design and analysis – meet all usual software quality criteria (reliable, understandable, flexible, maintainable)
- Demonstrates substantial independent research on iOS development
- In the viva can communicate complexity clearly and succinctly with excellent standard of presentation and understanding

70-80: An excellent piece of work: All assessment criteria have been met at a high standard

- Demonstrates a sophisticated application and some originality
- Draws on a range of techniques and frameworks that may go beyond the taught material
- Provides a robust solution (reliable, expandable, understandable, maintainable)
- Demonstrates substantial independent research
- Communicates ideas clearly and succinctly with good standard of presentation

60-69: A good piece of work: Nearly all assessment criteria have been met at a good standard with just a few minor defects

- Demonstrates breadth and/or depth of understanding across frameworks and language
- Demonstrates research and critical use of resources.
- Communicates ideas clearly with a good understanding

50-59: A reasonable piece of work: Nearly all assessment criteria have been met at a good standard with a number of small defects or one major defect.

- Demonstrates breadth and/or depth of understanding across frameworks and the language

- Communicates ideas clearly with a good understanding but with some deficiencies in knowledge

40-49: An adequate piece of work: Assessment criteria have mostly been met but with a collection minor and major defects.

- Demonstrates understanding of appropriate range of framework concepts and practical approaches

- A basic application with some serious defects such as crashes

- No demonstrable original thinking in design

- Only partial or superficial understanding demonstrated

30-39 FAIL: An inadequate piece of work: Relevant assessment tasks are not met.

- Has clear limitations in the practice and approach to development
- Demonstrates lack of design and analysis
- Uses a narrow range of techniques/frameworks to support development
- Communication is unclear with significant weaknesses in explanation of code
- Significant defects

0-29 FAIL: Poor piece of work: Most of the assessment tasks have not been met.

- Demonstrates poor understanding of key concepts and techniques
- Shows significant weaknesses and omissions in completing the tasks
- No evidence of analysis, process or design

Communication is unclear with significant weaknesses in understanding

End of document