ITECH2000 Mobile Development Fundamentals

Week 3 Assessable Task

Overview

You are required to write pseudocode and create an app that utilises concepts taught up to (and including) week 3 to meet a provided problem specification.

Timelines and Expectations

Percentage Value of Task:	10% of semester total
Due:	11:59pm on Sunday, 4th August, 2019
	For late submissions, a penalty of 10% of available marks will be applied for each day the assignment is overdue.
Minimum time expectation:	Depending on your experience, this task will take at least half an hour. Some students may require as much as 3 hours to complete it.

Learning Outcomes Assessed:

The following course learning outcomes are assessed to some degree by completing this assessment:

- **K1.** Understand constructs typical of many programming languages such as: variables, expressions, assignment, sequence, selection, iteration, procedures, parameters, return values.
- **A1.** Design, develop, test and debug mobile apps from a given textual program specification.
- **S1.** Analyse the input, processing and output needs of small programming problems.

Assessment Details

Overview of Situation

A local grocery store is introducing a loyalty program that will reward points to customers who shop in the store. To help customers forecast the number of points they will receive for each shop, the grocery store wants you to create a "Rewards Point Calculator" mobile app. There are a range of factors that determine how many points a customer may receive each week.

The first factor that matters is the total cost of the groceries. If the customer has purchased less than $100 worth of groceries, they are awarded 1 reward point per dollar spent. If the customer has purchased at least $100 worth of groceries, they are awarded 2 points per dollar spent. If the customer has purchased $500 or more worth of groceries, they are awarded an additional 200 bonus points.

The second factor that affects the total points is the length of time that the customer has been in the loyalty program. If the customer has been a member for less than 12 months, they will receive no bonus points. If the customer has been a member for longer than 12 months, they will receive 10 bonus points for every additional month they have been a member.

Staff may also participate in the promotion, but the total number of points they receive is reduced by 30%.

Required Behaviour of the App

The app you design needs to address all of the following:

1. The user must be able to enter the total cost of the groceries, and the number of months they have been a loyalty member.
2. The user must be able to indicate whether they are a staff member (and thus receive the reduced number of points).
3. The user must be able to request a forecast of what the points awarded will be, but without actually going ahead with submitting the points. The rules given above must be followed to calculate the points. The user is not obliged to go-ahead with submitting the number of points.
4. After seeing the proposed points to be awarded, it must be possible for the user to actually go ahead with submitting the points. When this occurs, the app needs to:
    a. increase a variable noting the total points of all shops that have been submitted, since the app started running.
    b. increase a counter noting how many shops have been actually submitted since the app started running.
    c. display a message saying how many shops have been submitted and what the total number of points calculated has been, since the app started running.
    d. Clear/Reset the input components so that they are ready for the next shop's details to be entered.

Before you start building the app in MIT AppInventor, you should plan your program using pseudocode. This pseudocode will also form part of your submission for this assessment task.

Tasks

Based on the provided scenario and required app behaviour, the following tasks must be completed for submission:

1. Write an algorithm for the behaviour of each event that your app will respond to. Put this work into a Microsoft Word document.
2. Create an app in AppInventor, named "Week3_*YourName*", which implements in code the algorithms that you wrote for each event. Ensure you choose appropriate user interface components, component names, and variable names. Make sure that you test your app thoroughly using the emulator or MIT companion app.
3. Create a zip file containing both your algorithm (Word or PDF format) and completed app (.aia file exported from AppInventor), and submit this to Moodle. The name of the zip file should include both your name and student ID (i.e. *StudentName12345678.zip*).

Submission Instructions

This task can be submitted under "Week 3 Assessable Task (Early Intervention)" in the Assessments section of Moodle. First complete the "Assessment Declaration agreement: Assessment Task 1" to declare that you have not plagiarised your work. Once this is completed, the "Submit Assessment Task 1" submission point will become available. Upload your zip file here and make sure that you click the "Send for Marking" button.

Plagiarism

Plagiarism is the presentation of the expressed thought or work of another person as though it is one's own without properly acknowledging that person. You must not allow other students to copy your work and must take care to safeguard against this happening.
More information about the plagiarism policy and procedure for the university can be found at http://federation.edu.au/students/learning-and-study/online-help-with/plagiarism.

Feedback and Results

You will receive your marks broken down by each criteria, and the total, together with any comments giving suggestions on how you could have done better, within 2 weeks of the due date or the submission date (whichever was later). Before receiving your results, you may be required to attend an interview with your lecturer to discuss certain aspects of your assignment.

Marking Criteria

You will be assessed on the following things for the amount of marks as indicated. There are a total of 12 marks available.

Requirement	Marks	Awarded
1. Ability to assess problem specification and represent a solution in pseudocode	2	
2. Ability to choose appropriate components to use for input and output of information (including using components in the manner for which they are designed)	1	
3. Ability to set appropriate initial properties for components of a screen	0.5	
4. Ability to choose suitable events for which to write code sequences	0.5	
5. Ability to retrieve textual input and other input from the user	1	
6. Ability to programmatically change the value of properties of components, including to produce output messages.	1	
7. Ability to construct relational expressions and Boolean expressions to compare values	1	
8. Ability to make use of conditional code structures to control whether a sequence of code is to be executed or skipped-over	1	
9. Ability to make appropriate use of variables, including choosing whether to make a variable be global or local	0.5	
10. Correctness of algorithmic logic for dealing with the cost of groceries	1	
11. Correctness of algorithmic logic for dealing with number of months in loyalty program	1	
12. Correctness of algorithmic logic to deal with calculating the total points to be awarded (including staff offset)	1	
13. Correctness of algorithmic logic to deal with updates of counters and totals	0.5	
Total:	12	

Feedback: