MACQUARIE University

DEPARTMENT OF COMPUTING

Session 1 - 2024

ASSIGNMENT THREE

Gino Sunny, 47814403

COMP6750: Applications Modelling and Development

Task 2: Exploring the Mobile Application Development

2.1 Understanding Native vs. Cross-Platform App Development

When we talk about developing a mobile app, there are two main approaches: native app development and cross-platform app development. Think of it like building a house – you can either build it custom for the specific land (native) or build it in a way that works well on multiple types of land (cross-platform).

2.1.1 Native App Development

Native app development is like building a house designed specifically for the terrain of a particular plot of land. For example, building a house designed to perfectly fit the climate and soil of a coastal area. In this context, we build an app specifically for iOS (Apple devices) or Android (Google devices). For iOS, developers might use Swift or Objective-C with tools like Xcode (Apple Inc., no date), while Android developers typically use Java or Kotlin with Android Studio (*Kotlin and Android*, no date).

2.1.1.1 Benefits

Performance: The app runs very smoothly and quickly because it's optimized for one specific platform (Holzer and Ondrus, 2012).
User Experience: It feels more natural and intuitive for users because it's designed with all the unique features of the platform in mind (Holzer and Ondrus, 2012).

2.1.2 Cross-Platform App Development

Cross-platform development is like building a house that can be placed on different types of land without much modification. We create one app that works on both iOS and Android platforms. Emerging frameworks like React Native, Flutter, and Xamarin have become popular choices.

2.1.2.1 Benefits

Efficiency: It saves time and money because we write most of the code once and it works on both platforms (Palmieri, Singh and Cicchetti, 2012).

Consistency: The app looks and feels similar on both iOS and Android, which is great for brand consistency (Palmieri, Singh and Cicchetti, 2012).

2.1.3 Real-world Example

Imagine you're launching a shopping app. If you choose native development, you'd have a version specifically tailored for iPhones and another for Android phones. This means better performance and user experience, but higher costs and more time to develop. With cross-platform development, you'd create one app that works on both types of devices. It's quicker and cheaper to develop, though it might not be as finely tuned as a native app.

The choice between native and cross-platform depends on your priorities. If you want the best performance and user experience, and you have the budget, go with native. If you need to launch quickly and cost-effectively, cross-platform is a good choice. Both approaches have their advantages and trade-offs, and the best choice depends on what's most important for your project.

2.2 Ethical Dilemma Consideration

In the scenario where I'm facing a lack of credit for my contribution to a programming project, the following points from the ACS Code of Professional Ethics are most relevant to my situation.

2.2.1 Relevant Principles from the ACS Code of Professional Ethics

The believe, the following two principles from ACS Code of Professional Ethics are relevant for this case:

1. Honesty

Section 2.1a: It is expected of me to maintain honesty, openness, and truthfulness in all dealings with employers, colleagues, clients, stakeholders, and the public.

Section 2.1b: I must accurately represent all actions, circumstances, and abilities—my own, those of my colleagues, my employer, or anyone else I interact with, without distortion or omission, whether directly or indirectly.

Section 2.1c: I must not remain silent when I detect unprofessional conduct.

2. Trustworthiness

Section 2.2a: I need to be accountable for all that I undertake, prepared to take responsibility for failures as well as successes.

Section 2.2b: I am required to uphold integrity, maintain consistency in my opinions, speech, and behavior, and effectively handle any conflicts of interest.

Section 2.2d: I must respect the privacy, confidentiality, and integrity of any personal or proprietary information that comes into my possession.

2.2.2 Available Choices/Options for Action

1. Continue to Advocate Internally

Dialogue with Colleagues: I can attempt further discussions with my peer and development lead, presenting my case clearly and referencing the ACS Code of Professional Ethics to highlight the importance of ethical practices such as acknowledging contributions.

Formal Complaint: If informal discussions do not yield results, I can file a formal complaint with my organization's human resources department or ethics committee, detailing the breach in ethical conduct.

2. Document Everything

Evidence: I should ensure that I document all contributions I've made, along with any correspondence related to this issue. This documentation will be crucial if formal resolutions or external interventions become necessary.

3. Seek Guidance from Professional Bodies

External Advice: I can seek advice from external bodies like the Australian Computer Society (ACS) regarding the situation. They may offer mediation services or advice on how to proceed in line with professional standards.

2.2.3 Guidance from ACS Professional Ethics Principles

In addressing the scenario, the ACS Code of Professional Ethics guides me to prioritize several key actions and ethical standards:

1. Uphold Honesty

Communicate Openly: The ACS code emphasizes the need for honesty in all professional interactions. It guides me to continue advocating for recognition of my contributions in a transparent and straightforward manner.

Address Misconduct: The code also specifies not to remain silent when detecting unprofessional conduct, guiding me to take active steps to address the issue if it is not being resolved adequately within the team.

2. Ensure Trustworthiness

Be Accountable: The code stresses accountability and taking responsibility. This not only pertains to my own work but also ensures that others are held accountable for their actions, including proper acknowledgment of contributions.

Manage Conflicts of Interest: This aspect of the code advises me to identify and address any potential conflicts of interest that may be influencing the situation, such as biases or personal gains from not crediting work appropriately.

2.2.4 Actions Based on Ethical Principles

Based on the above 3 points, I would take a structured approach to address the situation:

Step 1: Communicate and Advocate

Initiate Further Discussion: I would start by having another discussion with my peer and development lead. During this conversation, I would clearly outline my concerns, supported by specific examples of my contributions.

Documentation: Concurrently, I would document all interactions, contributions, and any relevant communications or events related to this issue. This record will be critical if further action is needed.

Step 2: Formal Actions

Formal Complaint: If the discussions do not lead to a satisfactory resolution, I would file a formal complaint with the human resources department or the ethics committee at my organization. This complaint would detail the situation, the steps I have already taken.

Seek External Guidance: Depending on the response from my organization, I might seek guidance from the ACS or another professional body. This could include asking for advice on how to proceed or even requesting mediation services if offered.

2.3 Test Case Documentation

Test Number	Test Name	Page Number	Date Run	Run by
Func_1	Calculate Total Cost when vegetable is selected & fruit not selected.	8	28/05/2024	Gino Sunny
Func_2	Show an alert when no vegetable or fruit is selected.	10	28/05/2024	Gino Sunny

2.3.1 Vegetable Selected, Fruit not Selected

When the user selects a vegetable and its quantity but does not select any fruit, the app calculates the total price based on the selected vegetable and its quantity. The calculation logic is triggered by the "Calculate Total Price" button, and the total price is displayed.

Reason for Choosing This Test Case:

This test case was chosen to verify the app's functionality when only a vegetable is selected, and no fruit is selected. It is crucial to ensure that the app correctly calculates the total price based on partial selections, which is a common user scenario. By testing this condition, we can confirm that the app handles single-category selections accurately and provides the correct total price without any errors.

****Selecting a specific test value isn't necessary since the test case should pass for all options available in the drop-down menu**.**

<u>Test Case Number:</u> Func_1

Test Name: Calculate Total Cost when vegetable is selected & fruit not selected.

Date test run: 28/05/2024

Test designed by: Gino Sunny

Test run by: Gino Sunny

Test purpose and goal

Calculate Total Cost of the Order when a vegetable is selected & fruit not selected and display it to the user.

Test setup instructions

Snack.expo.io; Run this on Android

Test preconditions and dependencies

None

Test Steps

Step#	Step Instruction	Expected Result *(What should the tester see after completing the step instruction)*	*Actual result seen in test*	*Step passed? (yes/no)*
1	The user selects the POGS app and navigates to the selection screen	The app displays the screen to calculate the total price with all the required fields.	The screen to calculate the total price is displayed with all the required fields.	Yes
2	Select a vegetable from the Vegetables drop-down	The vegetable is selected and displayed in the dropdown field	The selected Vegetable is displayed in the drop-down field	Yes
3	Select a quantity for the vegetable from the 'Qty'	The quantity is selected and displayed in the dropdown field	The selected quantity is displayed in the drop-down field	Yes

	drop-down			
4	Ensure no fruit or its quantity is selected	The fruit and its quantity drop-down field are empty	The fruit and its quantity are not selected and the fields are empty	Yes
5	Click on the "Calculate Total Price" button	The app replaces the ‘Press the above button to calculate’ text and displays the total price with the text ‘Total Cost of Order:’ before it based on the selected vegetable and its quantity	The ‘Press the above button to calculate’ text is replaced and the total price is displayed with the text ‘Total Cost of Order:’ before it.	Yes

Test Post Conditions:

N/A

Test Signed off: Positive

2.3.2 Both vegetables/fruit not selected

When the user does not select any vegetable or fruit and attempts to calculate the total price by clicking the "Calculate Total Price" button, the app shows an alert message indicating that at least one item and its quantity must be selected. This prevents the calculation from proceeding without valid inputs.

Reason for Choosing This Test Case:

This test case was chosen to verify the app's response when no vegetable or fruit is selected. This scenario is important because it tests the app's ability to handle invalid input conditions. Ensuring that the app provides appropriate feedback and not NaN, and also an alert message prompting the user to make a selection, helps maintain a good user experience and prevents unexpected behaviors or crashes. This test confirms that the app is robust and user-friendly by guiding users to complete necessary actions.

****Selecting a specific test value isn't necessary since the test case should pass for all options available in the drop-down menu**.**

<u>Test Case Number:</u> Func_2

Test Name: Show an alert when no vegetable or fruit is selected.

Date test run: 28/05/2024

Test designed by: Gino Sunny

Test run by: Gino Sunny

Test purpose and goal

Show an alert to the user when no vegetable or fruit and also no quantity is selected and the user attempts to calculate the price.

Test setup instructions

Snack.expo.io; Run this on Android

Test preconditions and dependencies

None

Test Steps

Step#	Step Instruction	Expected Result *(What should the tester see after completing the step instruction)*	*Actual result seen in test*	*Step passed? (yes/no)*
1	The user selects the POGS app and navigates to the selection screen	The app displays the screen to calculate the total price with all the required fields.	The screen to calculate the total price is displayed with all the required fields.	Yes
2	Ensure no fruit or its quantity is selected	The fruit and its quantity drop-down fields are empty	The fruit and its quantity are not selected and the fields are empty	Yes
3	Ensure no vegetable or its quantity is selected	The vegetable and its quantity drop-down fields are empty	The vegetable and its quantity are not selected and the fields are empty	Yes
4	Click on the "Calculate Total Price" button	The app should display an alert message indicating 'Please select at least one item and its quantity.''	The alert message indicating 'Please select at least one item and its	Yes

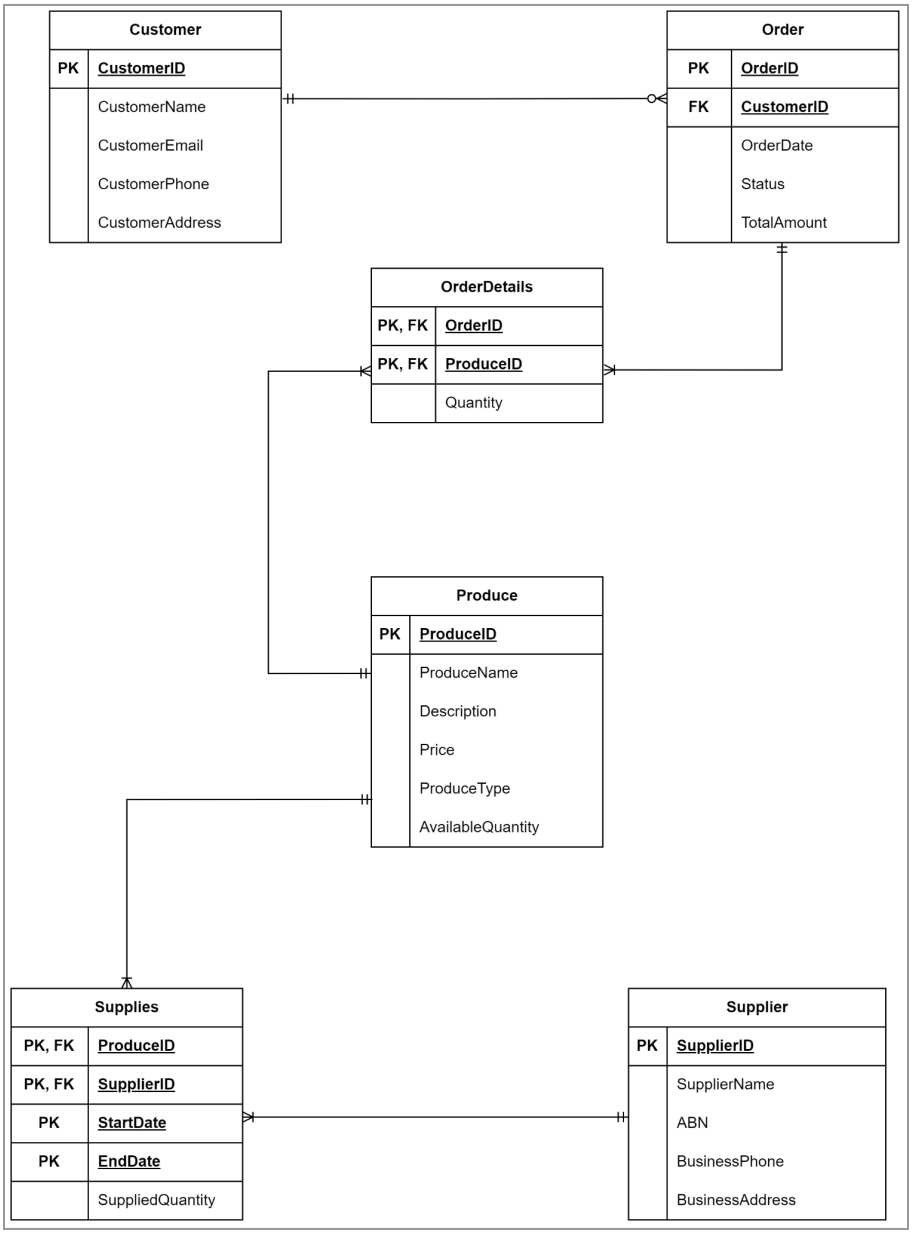
			quantity.' is displayed.	

Test Post Conditions:

N/A

Test Signed off: Positive

2.4 Schema Diagram



2.4.1 Entity Tables, Attributes & Constraints

- **Customer** (CustomerID[PK], CustomerName, CustomerEmail, CustomerPhone, CustomerAddress)
- **Order** (OrderID[PK], CustomerID[FK], OrderDate, Status, TotalAmount)
- **Order_Details** (OrderID[PK, FK], ProduceID[PK, FK], Quantity)
- **Produce** (ProduceID[PK], ProduceName, Description, Price, ProduceType, AvailableQuantity)
- **Supplies** (ProduceID[PK, FK], SupplierID[PK, FK], StartDate[PK], EndDate[PK], SuppliedQuantity)
- **Supplier** (SupplierID[PK], SupplierName, ABN, BusinessPhone, BusinessAddress)

2.4.2 Summary of Relationships

- **Customer to Order:** One-to-Many.
  Each customer can have zero or many orders, but each order is linked to one customer.
- **Order to Order_Details:** One-to-Many.
  Each order can consist of multiple line items, where each order detail is associated with one order.
- **Produce to Order_Details:** One-to-Many.
  Each Order_Details record references exactly one Produce item, while each Produce item can be associated with multiple Order_Details.
- **Produce to Supplies:** One-to-Many.
  Each produce can have multiple supplies records, indicating various supply instances by different suppliers.
- **Supplier to Supplies:** One-to-Many.
  Each supplier can be associated with multiple supply records for different produces.

2.4.3 Assumptions

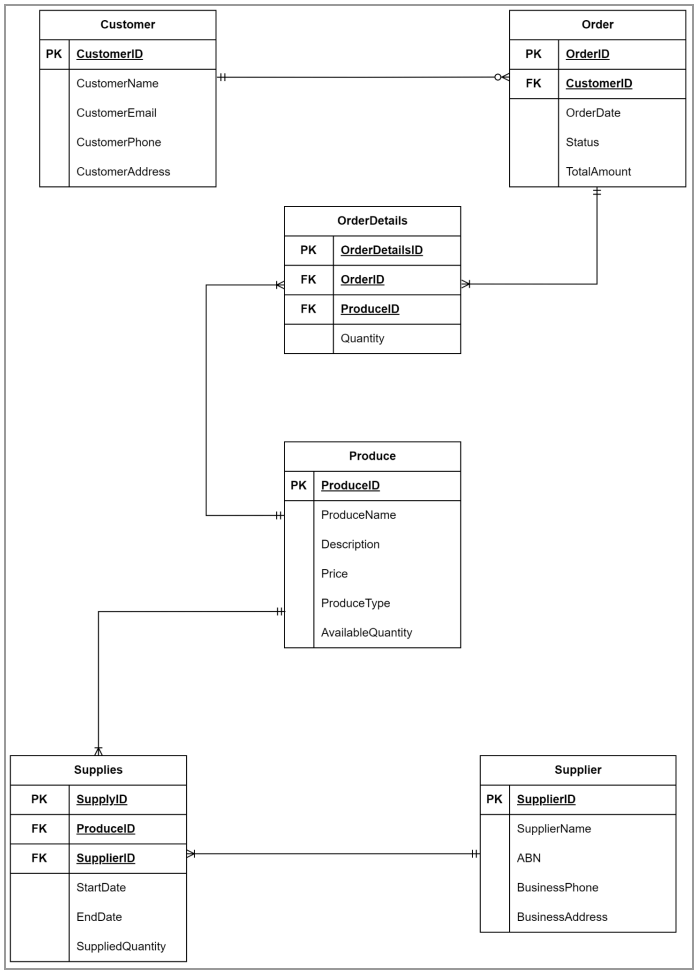
- It is assumed that Customer data is stored in the system and does not depend on the fact the customer has made an order or not.
- Order table will be used to store the order history of the customers.

- As fruit and vegetable are both ‘Produce’, the details of it are captured in the Produce table, with the attribute ProduceType distinguishing between a fruit and vegetable.
- The use of the table 'Supplies' is essential for effectively managing historical data and relationships between suppliers and produce. This approach addresses the need to track historical supplier-produce associations, which cannot be captured through direct foreign key constraints. Over time, multiple suppliers may supply the same produce, creating a many-to-many relationship historically, despite being one-to-one at any given instant. The 'Supplies' table not only accommodates the start and end dates for each supply period but also allows for the easy addition of attributes related to supply terms or conditions without restructuring the entire database schema.
- For Supplies, a composite key comprising ProduceID, SupplierID, StartDate & EndDate identifies records to ensure that there are no overlapping supply periods for the same produce from the same supplier.
- In the Supplies and Order_Details tables, using single unique identifiers like SupplyID and OrderDetailID instead of composite keys could streamline data management. SupplyID simplifies C.R.U.D operations and improves performance with easier indexing. Similarly, in Order_Details, the attribute OrderDetailID reduces complexity in application development and foreign key management.

  If the above assumptions are to be considered in the schema, then the Order_Details & Supplies tables would be as follows:

  Order_Details (OrderDetailID[PK], OrderID[FK], ProduceID[FK], Quantity)

  Supplies (SupplyID[PK], ProduceID[FK], SupplierID[FK], StartDate, EndDate, SuppliedQuantity).


Customer
PK
CustomerID
CustomerName
CustomerEmail
CustomerPhone
CustomerAddress
Order
PK
OrderID
FK
CustomerID
OrderDate
Status
TotalAmount
OrderDetails
PK
OrderDetailsID
FK
OrderID
FK
ProduceID
Quantity
Produce
PK
ProduceID
ProduceName
Description
Price
ProduceType
AvailableQuantity
Supplies
PK
SupplyID
FK
ProduceID
FK
SupplierID
StartDate
EndDate
SuppliedQuantity
Supplier
PK
SupplierID
SupplierName
ABN
BusinessPhone
BusinessAddress

2.5 References

Apple Inc. (no date) Swift - Apple Developer. https://developer.apple.com/swift/.

Holzer, A. and Ondrus, J., 2012. Mobile app development: Native or web?. In Proc. Workshop eBus.(WeB).

Kotlin and Android (no date). https://developer.android.com/kotlin.

Palmieri, M., Singh, I. and Cicchetti, A. (2012) 'Comparison of cross-platform mobile development tools', in 2012 16th International Conference on Intelligence in Next Generation Networks. 2012 16th International Conference on Intelligence in Next Generation Networks, pp. 179–186. Available at: https://doi.org/10.1109/ICIN.2012.6376023.