College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
IN721: Design and Development of Applications for Mobile Devices
Level 7, Credits 15

Assessment 02: Wishlist

Assessment Overview

For this assessment, you will use Kotlin with Android Studio to build a wishlist. As well as implementing the core functionality, you will be required to **independently** research & implement six components. In addition, marks will also be given for code elegance, robustness & git usage.

Assessment Table

Assessment Activity	Weighting	Learning Outcomes	Assessment Grading Scheme	Completion Requirements
Practicals	25%	1, 3, 4	CRA	Cumulative
Language Translator	20%	1, 3, 4	CRA	Cumulative
Wishlist	25%	1, 3, 4	CRA	Cumulative
Exams 1-5	30%	2, 3, 4	CRA	Cumulative

Conditions of Assessment

You will complete this assessment outside timetabled class time, however, there will be availability during the teaching sessions to discuss the requirements and progress of this assessment. This assessment will need to be completed by Friday, 19 June 2020 at 5pm.

Pass Criteria

This assessment is criterion-referenced with a cumulative pass mark of 50%.

Submission Details

You must submit your program files via **GitHub Classroom**. Here is the link to the repository you will be using for your submission – https://classroom.github.com/a/o712P_Rx. For ease of marking, please submit the marking sheet with your name & student id number via **Microsoft Teams** under the **Assignments** tab.

Group Contribution

All git commit messages must identify which member(s) participated in the associated work session. Proportional contribution will be determined by inspection of the commit logs. If the commit logs show evidence of significantly uneven contribution proportion, the lecturer may choose to adjust the mark of the lesser contributor downward by an amount derived from the individual contributions.

Authenticity

All parts of your submitted assessment must be completely your work and any references must be cited appropriately including, externally-sourced graphic elements. All media must be royalty free (or legally purchased) for educational use. Failure to do this will result in a mark of zero.

Policy on Submissions, Extensions, Resubmissions & Resits

The school's process concerning **Submissions, Extensions, Resubmissions and Resits** complies with Otago Polytechnic policies. Students can view policies on the Otago Polytechnic website located at https://www.op.ac.nz/about-us/governance-and-management/policies.

Extensions

Please familiarise yourself with the assessment due dates. If you need an extension, please contact your lecturer before the due date. If you require more than a week's extension, a medical certificate or support letter from your manager may be needed.

Resubmissions

Students may be requested to resubmit an assessment following a rework of part/s of the original assessment. Resubmissions are completed within a short time frame (usually no more than 5 working days) and usually must be completed within the timing of the course to which the assessment relates. Resubmissions will be available to students who have made a genuine attempt at the first assessment opportunity. The maximum grade awarded for resubmission will be C-.

Learning Outcomes

At the successful completion of this course, students will be able to:

1. Implement complete, non-trivial, industry-standard mobile applications following sound architectural and code-quality standards.
2. Explain relevant principles of human perception and cognition and their importance to software design.
3. Identify relevant use cases for a mobile computing scenario and incorporate them into an effective user experience design.
4. Follow industry standard software engineering practice in the design of mobile applications.

Instructions

Application Requirements - Learning Outcomes 1, 3, 4

The wishlist application **must** have the following functional requirements:

- System:
  - Open without modification in Android Studio. This includes the removal of the gradle & idea hidden directories.
  - Run without modification on multiple mobile devices.
- Features:
  - Add an item via a floating action button. Each item must have a picture, name, category, price, store, additional notes & whether it has been purchased.
    - Picture - captured via the mobile device's camera.
    - Store - display the name & location of the nearest store on Google Maps.
  - Edit an item.
  - Delete an item. Prompt the user via a dialog fragment.
  - Items persistently stored using SQLite.
  - Display all items in a recycler view.
  - **Research:** Display items in a recycler view by category, purchased & unpurchased.
  - Display a progress dialog while adding, editing & deleting an item.
  - Handle incorrect formatted input values.
  - **Research:** Sort items by newest first, oldest first & alphabetical order.
  - Localization which includes support for at least two languages.
  - Exit the application via an alert dialog.
  - Multiple mobile screen/device support. Include support for at least two screens/devices, excluding tablets & smartwatches devices.
  - Rate my app. Send the user to the application on Google Play Store.
  - **Research:** Display privacy policy in a web view.
  - Display the wishlist's total amount default is the New Zealand Dollar (NZD). For example, item one is $10.00 & item two is $5.00, the wishlist's total amount would be $15.00.
  - **Research:** Change the currency unit. Include at least two currencies excluding the New Zealand Dollar (NZD).
  - Notify the user of unpurchased items. For example, **"Currently, you have four unpurchased items."**
  - **Research:** Enable & disable push notifications.
- User-Interface:
  - Visually attractive user-interface with a coherent graphical theme & style. This can be a custom theme/style or Material Design.
  - Splash screen with a transition animation.
  - **Research:** Transition animations between activities/fragments.
  - Adaptive launcher icon.

Documentation - Learning Outcomes 1, 3, 4

Write the following documentation requirements in the README.md file:

- Step-by-step user guide. Must include a screenshot of each activity/fragment provided with a description.
- Code commented & documented using KDoc. Kotlin's documentation generation tool is called Dokka. Click here for usage instructions.
- Privacy policy.
- User-testing report which includes at least two subjects excluding assessment group members. **More information provided below.**
  - Detailed report of user-testing protocol including number of subjects run.
  - Detailed description of issues raised during user-testing & specifics of action taken to resolve each issue.

Git Usage - Learning Outcomes 1, 3, 4

The wishlist repository must have the following git requirements:

- At least five feature branches excluding master.
- Commit messages reflect the context of each functional requirement change.

Additional Resources

- Google Maps API - https://developers.google.com/maps/documentation
- Documenting Kotlin Code - https://kotlinlang.org/docs/reference/kotlin-doc.html
- Android Design - https://developer.android.com/design
- Adaptive Launcher Icon - https://developer.android.com/guide/practices/ui_guidelines/icon_design_adaptive

IN721: Design & Development of Applications for Mobile Devices
Semester 1, 2020

Assessment 02: Wishlist
Assessment Rubric

	10-9	8-7	6-5	4-0
Functionality & Robustness	Application thoroughly demonstrates functionality & robustness on the following: • Open without modification in Android Studio. This includes the removal of the gradle & idea hidden directories. • Run without modification on multiple mobile devices. • Add an item via a floating action button. • Edit & delete an item. • Items persistently stored using SQLite. • Display all items in a recycler view. • Display items in a recycler view by category, purchased & unpurchased. • Display a progress dialog while adding, editing & deleting an item. • Handle incorrect formatted input values. • Sort items by newest first, oldest first & alphabetical order. • Localization which includes support for at least two languages. • Exit the application via an alert dialog. • Multiple mobile screen/device support. Include support for at	Application mostly demonstrates functionality & robustness on the following: • Open without modification in Android Studio. This includes the removal of the gradle & idea hidden directories. • Run without modification on multiple mobile devices. • Add an item via a floating action button. • Edit & delete an item. • Items persistently stored using SQLite. • Display all items in a recycler view. • Display items in a recycler view by category, purchased & unpurchased. • Display a progress dialog while adding, editing & deleting an item. • Handle incorrect formatted input values. • Sort items by newest first, oldest first & alphabetical order. • Localization which includes support for at least two languages. • Exit the application via an alert dialog. • Multiple mobile screen/device support. Include support for at	Application demonstrates some functionality & robustness on the following: • Open without modification in Android Studio. This includes the removal of the gradle & idea hidden directories. • Run without modification on multiple mobile devices. • Add an item via a floating action button. • Edit & delete an item. • Items persistently stored using SQLite. • Display all items in a recycler view. • Display items in a recycler view by category, purchased & unpurchased. • Display a progress dialog while adding, editing & deleting an item. • Handle incorrect formatted input values. • Sort items by newest first, oldest first & alphabetical order. • Localization which includes support for at least two languages. • Exit the application via an alert dialog. • Multiple mobile screen/device support. Include support for at	Application does not or does not fully demonstrate functionality & robustness on the following: • Open without modification in Android Studio. This includes the removal of the gradle & idea hidden directories. • Run without modification on multiple mobile devices. • Add an item via a floating action button. • Edit & delete an item. • Items persistently stored using SQLite. • Display all items in a recycler view. • Display items in a recycler view by category, purchased & unpurchased. • Display a progress dialog while adding, editing & deleting an item. • Handle incorrect formatted input values. • Sort items by newest first, oldest first & alphabetical order. • Localization which includes support for at least two languages. • Exit the application via an alert dialog. • Multiple mobile screen/device support. Include support for at

	least two screens/devices, excluding tablets & smartwatches devices. • Send the user to the application on Google Play Store. • Display privacy policy in a web view. • Display the wishlist's total amount. • Change the currency unit which includes at least two currencies. • Notify the user of unpurchased items. • Enable & disable push notifications. • Visually attractive user-interface with a coherent graphical theme and style. • Splash screen with a transition animation. • Transition animations between activities/fragments. • Adaptive launcher icon.	least two screens/devices, excluding tablets & smartwatches devices. • Send the user to the application on Google Play Store. • Display privacy policy in a web view. • Display the wishlist's total amount. • Change the currency unit which includes at least two currencies. • Notify the user of unpurchased items. • Enable & disable push notifications. • Visually attractive user-interface with a coherent graphical theme and style. • Splash screen with a transition animation. • Transition animations between activities/fragments. • Adaptive launcher icon.	least two screens/devices, excluding tablets & smartwatches devices. • Send the user to the application on Google Play Store. • Display privacy policy in a web view. • Display the wishlist's total amount. • Change the currency unit which includes at least two currencies. • Notify the user of unpurchased items. • Enable & disable push notifications. • Visually attractive user-interface with a coherent graphical theme and style. • Splash screen with a transition animation. • Transition animations between activities/fragments. • Adaptive launcher icon.	least two screens/devices, excluding tablets & smartwatches devices. • Send the user to the application on Google Play Store. • Display privacy policy in a web view. • Display the wishlist's total amount. • Change the currency unit which includes at least two currencies. • Notify the user of unpurchased items. • Enable & disable push notifications. • Visually attractive user-interface with a coherent graphical theme and style. • Splash screen with a transition animation. • Transition animations between activities/fragments. • Adaptive launcher icon.
Documentation	Step-by-step user guide thoroughly describes each activity/fragment screenshot. Application code is thoroughly commented & documented with KDoc/Dokka. Privacy policy thoroughly discloses how the application collects, uses & shares data including the parties whom it's shared. User-testing report thoroughly describes the user-testing protocol, issues raised during testing & actions	Step-by-step user guide mostly describes each activity/fragment screenshot. Most application code is commented & documented with KDoc/Dokka. Privacy policy mostly discloses how the application collects, uses & shares data including the parties whom it's shared. User-testing report mostly describes the user-testing protocol, issues raised during testing & actions taken to resolve each issue.	Step-by-step user guide briefly describes each activity/fragment screenshot. Some application code is commented & documented with KDoc/Dokka. Privacy policy briefly discloses how the application collects, uses & shares data including the parties whom it's shared. User-testing report briefly describes the user-testing protocol, issues raised during testing & actions taken to resolve each issue.	Step-by-step user guide does not or does not fully describe each activity/fragment screenshot. Application code is not or is not fully commented & documented with KDoc/Dokka. Privacy policy does not or does not fully disclose how the application collects, uses & shares data including the parties whom it's shared. User-testing report does not or does not fully describe the user-testing protocol, issues raised during testing &

	taken to resolve each issue.			actions taken to resolve each issue.
Code Elegance	All Kotlin files contain no magic numbers/strings. All XML files contain no magic numbers/strings. Application thoroughly demonstrates code elegance on the following: • Correct use of intermediate variables, e.g., no method calls as arguments. • Idiomatic use of control flow, data structures & other in-built functions. • Sufficient modularity, e.g., code adheres to various OO design principles. • Adhere to a complex OO architecture, e.g., classes, methods, concise naming & methods assigned to the correct classes. • Efficient algorithmic approach.	Most Kotlin files contain no magic numbers/strings. Most XML files contain no magic numbers/strings. Application mostly demonstrates code elegance on the following: • Correct use of intermediate variables, e.g., no method calls as arguments. • Idiomatic use of control flow, data structures & other in-built functions. • Sufficient modularity, e.g., code adheres to various OO design principles. • Adhere to a complex OO architecture, e.g., classes, methods, concise naming & methods assigned to the correct classes. • Efficient algorithmic approach.	Some Kotlin files contain no magic numbers/strings. Some XML files contain no magic numbers/strings. Application demonstrates some code elegance on the following: • Correct use of intermediate variables, e.g., no method calls as arguments. • Idiomatic use of control flow, data structures & other in-built functions. • Sufficient modularity, e.g., code adheres to various OO design principles. • Adhere to a complex OO architecture, e.g., classes, methods, concise naming & methods assigned to the correct classes. • Efficient algorithmic approach.	Kotlin files contain frequent magic numbers/strings. XML files contain frequent magic numbers/strings. Application does not or does not fully demonstrate code elegance on the following: • Correct use of intermediate variables, e.g., no method calls as arguments. • Idiomatic use of control flow, data structures & other in-built functions. • Sufficient modularity, e.g., code adheres to various OO design principles. • Adhere to a complex OO architecture, e.g., classes, methods, concise naming & methods assigned to the correct classes. • Efficient algorithmic approach.
Git Usage	Git commit messages thoroughly reflect the functional requirement changes. Git branches thoroughly named & describe the context of the functional requirements.	Git commit messages mostly reflect the functional requirement changes. Git branches mostly named & describe the context of the functional requirements.	Git commit messages reflect some of the functional requirement changes. Git branches named & describe some of the context of the functional requirements.	Git commit messages do not or do not fully reflect the context of each solution. Git branches incorrectly named & do not or do not fully describe the context of the functional requirements.

Marking Cover Sheet

Assessment 02: Wishlist

IN721: Design & Development of Application for Mobile Devices

Level 7, Credits 15

Bachelor of Information Technology

Name:_____ Date: _____

Learner ID: _____

Assessor's Name: _____

Assessor's Signature: _____

Criteria	Out Of	Weighting	Final Result
Functionality & Robustness	**10**	**40**	
Documentation	**10**	**30**	
Code Elegance	**10**	**20**	
Git Usage	**10**	**10**	
		Final Result	**/100**
This assessment is worth 25% of the final mark for the Design & Development of Application for Mobile Devices course.			