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 01: Kotlin Travelling App

Assessment Overview

For this assessment, you will develop & publish a travelling application using Kotlin in Android Studio & Google Play Store. Intermediate Android features such as asynchronous tasks, recycler view widget, card view widget, shared preferences, SQLite & maps were covered formally in class. The main purpose of this assessment is not just to build a mobile application, rather to demonstrate your ability to effectively learn intermediate/advanced Android features independently. In addition, marks will be allocated for application robustness, code elegance, documentation & git usage.

The travelling application will help you sound like a local abroad & help you adapt to a new culture. You begin by selecting a continent & country tool. For example, if you are travelling to Japan, you will have text translation which will translate text in English to Japanese & vice-versa, text-to-speech, a selection of key Japanese phrases & an interactive quiz to test your knowledge of Japanese culture. A user of your travelling application should be able to select from at least two country tools per each continent **excluding** Antarctica.

Assessment Table

Assessment Activity	**Weighting**	**Learning Outcomes**	**Assessment Grading Scheme**	**Completion Requirements**
Practicals	10%	1, 3, 4	CRA	Cumulative
Kotlin Travelling App	35%	1, 3, 4	CRA	Cumulative
React Native Social Game App	25%	1, 3, 4	CRA	Cumulative
Kotlin Exam	15%	2, 3, 4	CRA	Cumulative
React Native Exam	15%	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 Wednesday, 14 October 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 use for your submission – https://classroom.github.com/a/Z5yfBBUS.

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 date. 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

Functionality & Robustness - Learning Outcomes 1, 3, 4

- Application must open without file structure modification in Android Studio.
- Application must run without code modification on multiple mobile devices.
- Text translation support for at least two countries per continent. If a country is multilingual (use of more than one language), choose only one language.
  - Use an asynchronous task & Yandex Translate API to translate the text from one language to another.
  - Display a progress dialog while the text is being translated.
  - Handle incorrectly formatted input fields, for example, a blank or empty edit text widget.
  - **Resource:** https://tech.yandex.com/translate
- Text-to-speech support for at least two countries per continent.
  - If a country is not supported, please handle correctly by either using a snackbar widget or a toast message.
  - **Resource:** https://developer.android.com/reference/kotlin/android/speech/tts/TextToSpeech
- Selection of at least four key phrases per country per continent.
- An interactive quiz for at least one country per continent.
  - Each quiz must have at least five questions.
  - Questions can be multi-choice &/or true or false.
  - Quiz topics may include animals, culture, food & drink, geography & sport.
  - Each question must have an image. Multi-choice questions must have four answers.
  - Display appropriate feedback for correct & incorrect answers by either using a text view widget or a toast message. If a question is answered incorrectly, display the correct answer.
  - At the end of each quiz, store the user's score in a SQLite database table.
  - For each quiz, display the user's top three scores in either a recycler view widget or a text view widget.
- Localization support for at least two countries per continent.
  - **Resource:** https://developer.android.com/guide/topics/resources/localization
- Application can be exited via a dialog fragment. The dialog fragment should prompt the user when the user double taps the mobile device's back button.
- For each country, display a Google marker on a Google map. The marker's information window should display the name of the country & capital city. The marker's coordinates will be the latitude & longitude of the capital city.
  - **Resource:** https://developers.google.com/maps/documentation/android-sdk/start
- Switch widget which toggles between light & dark mode.
  - The state (true or false) of the switch widget must be stored in shared preferences.
  - The mode will be based off the state of the switch widget, for example, true equals dark mode & false equals light mode.
  - **Resource:** https://developer.android.com/guide/topics/ui/look-and-feel/darktheme
- Splash screen with an image view widget & transition animation.
  - The transition animation must be a custom animation XML file.
  - **Resource:** https://developer.android.com/guide/topics/resources/animation-resource

- Adaptive launcher icon which displays a variety of shapes across different mobile devices.
  - **Resources:**
    - https://developer.android.com/guide/practices/ui_guidelines/icon_design_adaptive
    - https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html
- Bottom navigation view widget which navigates the user to the appropriate activities.
  - **Resource:** https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView
- Display the application's privacy policy in a web view widget.
  - **Resource:** https://play.google.com/about/developer-content-policy
- Visually attractive user-interface with a coherent graphical theme & style using Material Design.
- Application is published to Google Play Store.
  - When you create the application, please name the package appropriately, for example, **op.johndoe.travelling**. **Note:** replace **johndoe** with your Otago Polytechnic Ltd username.
  - **Resources:**
    - https://support.google.com/googleplay/android-developer/answer/113469?hl=en
    - https://developer.android.com/studio/publish/app-signing
- Ability to download the application from Google Play Store on to multiple mobile devices.

Documentation & Git Usage - Learning Outcomes 3, 4

- Provide the following in the repository README file:
  - Privacy policy which discloses user information collected by the application.
  - Sketched wireframes of the application. This can be hand-written or digitalised.
  - Step-by-step user application guide detailing each activity. The user guide must contain a screenshot of each activity in the application.
  - Commented code is documented using KDoc & generated to **Markdown** using Dokka.
    - **Resources:**
      - https://kotlinlang.org/docs/reference/kotlin-doc.html
      - https://github.com/Kotlin/dokka
- At least 10 feature branches excluding the **master** branch.
  - Your branches must be prefix with **feature**, for example, **feature-<name of functional requirement>**.
  - For each branch, merge your own pull request to the **master** branch.
- Commit messages must reflect the context of each functional requirement change.

Assessment 01: Kotlin Travelling App Assessment Rubric

	10-9	8-7	6-5	4-0
Functionality & Robustness	Application thoroughly demonstrates functionality & robustness on the following: • Application opens without file structure modification. • Application runs without code modification. • Text translation support. • Text-to-speech support. • Selection of key phrases. • Interactive quiz. • Localization support. • Exiting via a dialog fragment. • Google markers & map. • Light & dark mode. • Splash screen with image view widget & transition animation. • Adaptive launcher icon. • Bottom navigation view widget which navigates the user to the appropriate activities. • Privacy policy in a web view widget. • Visually attractive user interface. • Published to Google Play Store. • Application downloadable from Google Play Store.	Application mostly demonstrates functionality & robustness on the following: • Application opens without file structure modification. • Application runs without code modification. • Text translation support. • Text-to-speech support. • Selection of key phrases. • Interactive quiz. • Localization support. • Exiting via a dialog fragment. • Google markers & map. • Light & dark mode. • Splash screen with image view widget & transition animation. • Adaptive launcher icon. • Bottom navigation view widget which navigates the user to the appropriate activities. • Privacy policy in a web view widget. • Visually attractive user interface. • Published to Google Play Store. • Application downloadable from Google Play Store.	Application demonstrates some functionality & robustness on the following: • Application opens without file structure modification. • Application runs without code modification. • Text translation support. • Text-to-speech support. • Selection of key phrases. • Interactive quiz. • Localization support. • Exiting via a dialog fragment. • Google markers & map. • Light & dark mode. • Splash screen with image view widget & transition animation. • Adaptive launcher icon. • Bottom navigation view widget which navigates the user to the appropriate activities. • Privacy policy in a web view widget. • Visually attractive user interface. • Published to Google Play Store. • Application downloadable from Google Play Store.	Application does not or does not fully demonstrate functionality & robustness on the following: • Application opens without file structure modification. • Application runs without code modification. • Text translation support. • Text-to-speech support. • Selection of key phrases. • Interactive quiz. • Localization support. • Exiting via a dialog fragment. • Google markers & map. • Light & dark mode. • Splash screen with image view widget & transition animation. • Adaptive launcher icon. • Bottom navigation view widget which navigates the user to the appropriate activities. • Privacy policy in a web view widget. • Visually attractive user interface. • Published to Google Play Store. • Application downloadable from Google Play Store.

Code Elegance	All Kotlin files contain no magic numbers/strings & are stored in their appropriate XML files. All XML files contain no magic numbers/strings & are stored in their appropriate XML files. Application code thoroughly demonstrates code elegance on the following: • Correct use of intermediate variables, i.e., no method calls as arguments. • Idiomatic use of control flow, data structures & other in-built functions. • Sufficient modularity, i.e., code adheres to various OO design principles. • Adhere to an OO architecture, i.e., classes, functions, concise naming & functions assigned to the correct classes. • Efficient algorithmic approach.	Most Kotlin files contain no magic numbers/strings & are stored in their appropriate XML files. Most XML files contain no magic numbers/strings & are stored in their appropriate XML files. Application code mostly demonstrates code elegance on the following: • Correct use of intermediate variables, i.e., no method calls as arguments. • Idiomatic use of control flow, data structures & other in-built functions. • Sufficient modularity, i.e., code adheres to various OO design principles. • Adhere to an OO architecture, i.e., classes, functions, concise naming & functions assigned to the correct classes. • Efficient algorithmic approach.	Some Kotlin files contain no magic numbers/strings & are stored in their appropriate XML files. Some XML files contain no magic numbers/strings & are stored in their appropriate XML files. Application code demonstrates some code elegance on the following: • Correct use of intermediate variables, i.e., no method calls as arguments. • Idiomatic use of control flow, data structures & other in-built functions. • Sufficient modularity, i.e., code adheres to various OO design principles. • Adhere to an OO architecture, i.e., classes, functions, concise naming & functions assigned to the correct classes. • Efficient algorithmic approach.	Kotlin files contain frequent magic numbers/strings & are not or are not fully stored in their appropriate XML files. XML files contain frequent magic numbers/strings & are not or are not fully stored in their appropriate XML files. Application code does not does not fully demonstrate code elegance on the following: • Correct use of intermediate variables, i.e., no method calls as arguments. • Idiomatic use of control flow, data structures & other in-built functions. • Sufficient modularity, i.e., code adheres to various OO design principles. • Adhere to an OO architecture, i.e., classes, functions, concise naming & functions assigned to the correct classes. • Efficient algorithmic approach.

Documentation & Git Usage	Privacy policy thoroughly discloses user information collected by the application. Wireframes thoroughly sketched & reflect the final application. Step-by-step user guide thoroughly describes each activity in detail. Application code thoroughly commented with KDoc & generated with Dokka. Git branches thoroughly named with convention & contain the correct code relating to the functional requirement. Git commit messages thoroughly reflect the functional requirement changes.	Privacy policy mostly discloses user information collected by the application. Application wireframes mostly sketched & reflect the final application. Step-by-step user guide mostly describes each activity in detail. Application code mostly commented with KDoc & generated with Dokka. Git branches mostly named with convention & contain the correct code relating to the functional requirement. Git commit messages mostly reflect the functional requirement changes.	Privacy policy briefly discloses user information collected by the application. Application wireframes briefly sketched & reflect the final application. Step-by-step user guide briefly describes each activity in detail. Some application code commented with KDoc & generated with Dokka. Some git branches named with convention & contain the correct code relating to the functional requirement. Some git commit messages reflect the functional requirement changes.	Privacy policy does not or does not fully disclose user information collected by the application. Application wireframes are not or are not fully sketched & do not or do not fully reflect the final application. Step-by-step guide does not or does not fully describe each activity in detail. Application code is not or not fully commented with KDoc & generated with Dokka. Git branches are not or are not fully named with convention & do not or do not fully contain the correct code relating to the functional requirement. Git commit messages do not or do not fully reflect the functional requirement changes.

Assessment 01: Kotlin Travelling App Marking Cover Sheet

Name:

Date:

Learner ID:

Assessor's Name:

Assessor's Signature:

Criteria	Out Of	Weighting	Final Result
Functionality & Robustness	10	40	
Code Elegance	10	45	
Documentation & Git Usage	10	15	
Final Result			/100
This assessment is worth 35% of the final mark for the Design and Development of Application Mobile Devices course.			

Feedback: