Assessment 01: Language Translator
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. • Text translation which includes support for at least four languages, excluding English. • Display a progress dialog while in translation. • Handle incorrect formatted input values. • Localization which includes support for at least four languages. • Exit the application via an alert dialog. • Text-to-speech using the TextToSpeech Android class or a third-party library. • Multiple mobile screen/device support. Include support for at least four screens/devices, excluding tablets & smartwatches devices. • Interactive quiz for each language. • Visually attractive user-interface with a coherent graphical theme and style. • Toggle between light & dark mode. • Splash screen with a transition animation. • Adaptive launcher icon. • Bottom navigation which navigates the user to the appropriate activities. • Application published to Google Play Store.	Programs mostly 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. • Text translation which includes support for at least four languages, excluding English. • Display a progress dialog while in translation. • Handle incorrect formatted input values. • Localization which includes support for at least four languages. • Exit the application via an alert dialog. • Text-to-speech using the TextToSpeech Android class or a third-party library. • Multiple mobile screen/device support. Include support for at least four screens/devices, excluding tablets & smartwatches devices. • Interactive quiz for each language. • Visually attractive user-interface with a coherent graphical theme and style.	Programs demonstrate functionality & robustness on some of 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. • Text translation which includes support for at least four languages, excluding English. • Display a progress dialog while in translation. • Handle incorrect formatted input values. • Localization which includes support for at least four languages. • Exit the application via an alert dialog. • Text-to-speech using the TextToSpeech Android class or a third-party library. • Multiple mobile screen/device support. Include support for at least four screens/devices, excluding tablets & smartwatches devices. • Interactive quiz for each language. • Visually attractive user-interface with a coherent graphical theme and style.	Programs do not or do not fully demonstrate code elegance on any of 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. • Text translation which includes support for at least four languages, excluding English. • Display a progress dialog while in translation. • Handle incorrect formatted input values. • Localization which includes support for at least four languages. • Exit the application via an alert dialog. • Text-to-speech using the TextToSpeech Android class or a third-party library. • Multiple mobile screen/device support. Include support for at least four screens/devices, excluding tablets & smartwatches devices. • Interactive quiz for each language. • Visually attractive user-interface with a coherent graphical theme and style.

	• Download application from Google Play Store on mobile device.	• Toggle between light & dark mode. • Splash screen with a transition animation. • Adaptive launcher icon. • Bottom navigation which navigates the user to the appropriate activities. • Application published to Google Play Store. • Download application from Google Play Store on mobile device.	• Toggle between light & dark mode. • Splash screen with a transition animation. • Adaptive launcher icon. • Bottom navigation which navigates the user to the appropriate activities. • Application published to Google Play Store. • Download application from Google Play Store on mobile device.	• Toggle between light & dark mode. • Splash screen with a transition animation. • Adaptive launcher icon. • Bottom navigation which navigates the user to the appropriate activities. • Application published to Google Play Store. • Download application from Google Play Store on mobile device.
Documentation	Step-by-step user guide thoroughly describes each activity/fragment screenshot. Application code is thoroughly commented & documented with KDoc/Dokka.	Step-by-step user guide thoroughly describes each activity/fragment screenshot. Most application code is commented & documented with KDoc/Dokka.	Step-by-step user guide describes each activity/fragment screenshot. Some application code is commented & documented with KDoc/Dokka.	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.
Code Elegance	All Kotlin files contain no magic numbers/strings. All XML files contain no magic numbers/strings. Note: use dimen.xml (resource directory) to store magic numbers. 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.	Most Kotlin files contain no magic numbers/strings. Most XML files contain no magic numbers/strings. Note: use dimen.xml (resource directory) to store magic numbers. 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.	Some Kotlin files contain no magic numbers/string. Some XML files contain no magic numbers/strings. Note: use dimen.xml (resource directory) to store magic numbers. 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.	Some Kotlin files contain no magic numbers/strings. Some XML files contain no magic numbers/strings. Note: use dimen.xml (resource directory) to store magic numbers. Application does not or does not fully demonstrate code elegance on any of 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

	• Adhere to a complex OO architecture, e.g., classes, methods, concise naming & methods assigned to the correct classes. • Efficient algorithmic approach.	• Adhere to a complex OO architecture, e.g., classes, methods, concise naming & methods assigned to the correct classes. • Efficient algorithmic approach.	• Adhere to a complex OO architecture, e.g., classes, methods, concise naming & methods assigned to the correct classes. • Efficient algorithmic approach.	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 each solution.

Marking Cover Sheet

OTAGO POLYTECHNIC
Te Kura Matatini ki Otago

Assessment 01: Language Translator

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**	**35**	
Documentation	**10**	**25**	
Code Elegance	**10**	**30**	
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.			