College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
IN721: Mobile Application Development
Level 7, Credits 15

Project

Assessment Overview

In this assessment, you will develop & publish a travelling application using **Kotlin** in **Android Studio** & **Google Play Store**. **Android** features such as **ViewModel**, **LiveData**, **Room Database** & **Google Map** were formally covered in the teaching sessions. 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** & application development features independently. In addition, marks will be allocated for code elegance, documentation & **Git/GitHub** usage.

The travelling application will help you sound like a local abroad & help you adapt to a new culture. You will begin by selecting a continent & country tool. For example, if you were to travel to Spain, you would be provided with text translation & text to speech support, a selection of key Spanish phrases, an interactive quiz to test your knowledge of Spanish culture & a map containing locations of top-rated Spanish tourist attractions. A user of your travelling application must be able to select from at least two country tools per continent **excluding** Antarctica.

Learning Outcomes

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

1. Implement & publish complete, non-trivial, industry-standard mobile applications following sound architectural & code-quality standards.
2. Identify relevant use cases for a mobile computing scenario & incorporate them into an effective user experience design.
3. Follow industry standard software engineering practice in the design of mobile applications.

Assessment Table

Assessment Activity	Weighting	Learning Outcomes	Assessment Grading Scheme	Completion Requirements
Practical	20%	2, 3	CRA	Cumulative
Project	80%	1, 2, 3	CRA	Cumulative

Conditions of Assessment

You will complete this assessment during your learner managed time, however, there will be availability during the teaching sessions to discuss the requirements & your progress of this assessment. This assessment will need to be completed by **Wednesday, 23 June 2021 at 5:00 PM**.

Pass Criteria

This assessment is criterion-referenced (CRA) with a cumulative pass mark of **50%** over all assessments in **IN721: Mobile Application Development**.

Authenticity

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

Policy on Submissions, Extensions, Resubmissions & Resits

The school's process concerning submissions, extensions, resubmissions & resits complies with **Otago Polytechnic** policies. Learners can view policies on the **Otago Polytechnic** website located at https://www.op.ac.nz/about-us/governance-and-management/policies.

Submissions

You must submit all program files via **GitHub Classroom**. Here is the URL to the repository you will use for your submission – https://classroom.github.com/a/FWk_XkTA. The latest program files in the **main** branch will be used to run your application. Late submissions will incur a **10% penalty per day**, rolling over at **5:00 PM**.

Extensions

Familiarise yourself with the assessment due date. If you need an extension, contact the course 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

Learners may be requested to resubmit an assessment following a rework of part/s of the original assessment. Resubmissions are to be completed within a negotiable short time frame & usually must be completed within the timing of the course to which the assessment relates. Resubmissions will be available to learners who have made a genuine attempt at the first assessment opportunity & achieved a **D grade (40-49%)**. The maximum grade awarded for resubmission will be **C-**.

Resits

Resits & reassessments are not applicable in **IN721: Mobile Application Development**.

Instructions

You will need to submit an application & documentation that meet the following requirements:

Functionality - Learning Outcomes 1, 2, 3 (40%)

- Application must open without file structure modification in **Android Studio**.
- Application must run without code modification on multiple mobile devices. The mobile devices used to test your application will be:
  - Pixel 2 - 5.0"
  - Pixel XL - 5.5"
  - Pixel 3a XL - 6.0"
- Application must run on **API 28: Android 9.0 (Pie)**.
- Text translation support. If a country is multilingual (use of more than one language), choose only one language. For example, Canada's main languages are English & French.
  - Use **Retrofit** & the **Yandex Translate API** to translate text from one language to another. To use the **Yandex Translate API**, you will need an API key. A key is available in the **Microsoft Teams** course channel, under the **Files** tab.
  - Display a custom **ProgressDialog** while the text is being translated.
  - Handle incorrectly formatted input fields. For example, an **EditText** is blank or empty.
  - **Resource:** https://tech.yandex.com/translate
- Text to speech support.
  - If a country is not supported, handle gracefully with a **Toast**.
  - **Resource:** https://developer.android.com/reference/kotlin/android/speech/tts/TextToSpeech
- Selection of at least five key phrases. For example, "No worries, mate, she'll be right" is key phrase in Australia.
- An interactive quiz for each country.
  - Quiz data must be fetched from a **Spring Boot REST API** using **Retrofit**.
  - Quiz topics may include animals, culture, food, drink, geography & sport.
  - Each quiz must have at least five questions.
  - Questions are multi-choice & true or false.
  - Multi-choice questions must have four answers.
  - Each question must have an image.
  - Each question must be answered within a **30 second** time limit.
  - Display appropriate feedback for correct & incorrect answers with a **Toast**. If a question is answered incorrectly, display the correct answer.
  - At the end of each quiz, store the highest score in a **Room Database** table.
  - For each quiz, display the highest score in a **TextView**.
  - **Resource:** https://kotlinlang.org/docs/jvm-spring-boot-restful.html
- **Localization** support for each country.
  - **Resource:** https://developer.android.com/guide/topics/resources/localization
- Application can be exited via a **DialogFragment**. The **DialogFragment** must prompt the user when the user double taps the mobile device's back button.

- **Google Map** displaying top-rated tourist attractions.
  - Top-rated tourist attraction data must be fetched from a **Spring Boot REST API** using **Retrofit**.
  - Each data object will be represented by a marker on a **Google Map**.
  - The marker's information window must display the attraction's name & coordinates (latitude & longitude).
  - **Resource:** https://developers.google.com/maps/documentation/android-sdk/start
- **Switch** which toggles between light & dark mode.
  - The state (true or false) value of the **Switch** must be stored in **SharedPreferences**.
  - The mode will be based off the state value of the **Switch**. 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 **ImageView** & 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
- **BottomNavigationView** which navigates the user to the appropriate activities. For example, a menu icon for translation support, text to speech support, etc.
  - **Resource:** https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView
- Visually attractive UI with a coherent graphical theme & style using **Material Design**.
- Application is published to **Google Play Store**.
  - To published to **Google Play Store**, you will need a **Google Play Console** account. The account's credentials are available in the **Microsoft Teams** course channel, under the **Files** tab. The account will be available to all learners in the course. **Do not** disable any applications published on this account.
  - When you create your application, name the package appropriately. For example, **op.mobile.app.dev.<username>.travelling**. **Note:** replace **username** with your **Otago Polytechnic** username.
  - **Resources:**
    - https://support.google.com/googleplay/android-developer/answer/113469?hl=en
    - https://developer.android.com/studio/publish/app-signing
- Ability to download your application from **Google Play Store** on to multiple mobile devices. The mobile devices used to download your application will be:
  - Pixel 2 - 5.0"
  - Pixel XL - 5.5"
  - Pixel 3a XL - 6.0"
- At least **15** UI tests which verify that your application is functioning correctly.

Code Elegance - Learning Outcomes 1, 3 (45%)

- **Kotlin** & **XML** files contain no magic numbers/strings. Store the values in the appropriate **XML** files. For example, numbers must be stored in an integer or dimension file & strings must be stored in a string file.
- Use of intermediate variables. No method calls as arguments.
- Idiomatic use of control flow, data structures & other in-built functions.
- Code adheres to the **MVVM** architectural pattern &, **SOLID** & **DRY** design principles.
- Efficient algorithmic approach.
- **Kotlin** & **XML** files are code formatted.

Documentation & Git/GitHub Usage - Learning Outcomes 2, 3 (15%)

- Provide the following in your repository **README.md** file:
  - URL to your application's privacy policy.
    - **Resource:** https://play.google.com/about/developer-content-policy
  - Sketched wireframes of your application. The wireframes must be sketched/designed using online software. This must be completed before you start coding.
    - **Resource**: https://moqups.com
  - Step-by-step user guide detailing each screen. The user guide must contain a screenshot of each screen in your 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
  - **Spring Boot REST API** endpoints for the quiz & tourist-attraction features.
  - URL to your application on **Google Play Store**.
- Continuous integration using **GitHub Actions**.
  - **YAML** file must be configured for UI tests and generating an APK.
  - **Resource:** https://wkrzywiec.medium.com/github-actions-for-android-first-approach-f616c24aa0f9
- At least **10** feature branches excluding the **main** branch.
  - Your branches must be prefix with **feature**, for example, **feature-<name of functional requirement>**.
  - Code in the branch must relate to the **feature**.
  - Once you have completed a **feature**, create a pull request & assign the **GitHub** user **grayson-orr** to a reviewer. **Do not** merge your own pull request.
- Commit messages must reflect the context of each functional requirement change. **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.
  - **Resource:** https://www.freecodecamp.org/news/writing-good-commit-messages-a-practical-guide

Project Assessment Rubric

	10-9	8-7	6-5	4-0
Functionality	The application contains comprehensive & robust evidence on the following: • opens & runs on API 28: Android 9.0 (Pie) without file structure & code modification. • text translation, text to speech & localization support. • selection of key phrases. • interactive quiz. • exit application via dialog. • Google map displaying markers. • light & dark mode. • splash screen with image & transition animation. • adaptive launcher icon. • navigation to activities. • visually attractive UI. • published to & downloadable from Google Play Store. • UI tests verify correctness.	The application contains clear & detailed evidence of functionality on the following: • opens & runs on API 28: Android 9.0 (Pie) without file structure & code modification. • text translation, text to speech & localization support. • selection of key phrases. • interactive quiz. • exit application via dialog. • Google map displaying markers. • light & dark mode. • splash screen with image & transition animation. • adaptive launcher icon. • navigation to activities. • visually attractive UI • published to & downloadable from Google Play Store. • UI tests verify correctness.	The application contains evidence on the following: • opens & runs on API 28: Android 9.0 (Pie) without file structure & code modification. • text translation, text to speech & localization support. • selection of key phrases. • interactive quiz. • exit application via dialog. • Google map displaying markers. • light & dark mode. • splash screen with image & transition animation. • adaptive launcher icon. • navigation to activities. • visually attractive UI • published to & downloadable from Google Play Store. • UI tests verify correctness.	The application does not, or does not fully contain evidence on the following: • opens & runs on API 28: Android 9.0 (Pie) without file structure & code modification. • text translation, text to speech & localization support. • selection of key phrases. • interactive quiz. • exit application via dialog. • Google map displaying markers. • light & dark mode. • splash screen with image & transition animation. • adaptive launcher icon. • navigation to activities. • visually attractive UI • published to & downloadable from Google Play Store. • UI tests verify correctness.

Code Elegance	Kotlin & XML files thoroughly 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 MVVM, DRY & SOLID. • adhere to an OO architecture, i.e., classes, functions, concise naming & functions assigned to the correct classes. • efficient algorithmic approach. • code formatted Kotlin & XML files.	Kotlin & XML files clearly contain no magic numbers/strings & are stored in their appropriate XML files. Application code clearly 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 MVVM, DRY & SOLID. • adhere to an OO architecture, i.e., classes, functions, concise naming & functions assigned to the correct classes. • efficient algorithmic approach. • code formatted Kotlin & XML files.	Kotlin & XML files contain no magic numbers/strings & are stored in their appropriate XML files. Application code 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 MVVM, DRY & SOLID. • adhere to an OO architecture, i.e., classes, functions, concise naming & functions assigned to the correct classes. • efficient algorithmic approach. • code formatted Kotlin & XML files.	Kotlin & XML files contain frequent magic numbers/strings & are not or are not fully stored in their appropriate XML files. Application code does not or 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 MVVM, DRY & SOLID. • adhere to an OO architecture, i.e., classes, functions, concise naming & functions assigned to the correct classes. • efficient algorithmic approach. • code formatted Kotlin & XML files.

Documentation & Git Usage	README file contains comprehensive evidence of: • URL to application's privacy policy. • wireframes sketched of the application. • step-by-step user guide. • code commented with KDoc & generated with Dokka. • REST API GET endpoints. • URL to application on Google Play Store. Git branches comprehensively named with convention & contain code relating to the feature. Git commit messages comprehensively formatted & reflect the feature changes in concise detail. Continuous integration using GitHub Actions comprehensively setup.	README file contains clear evidence of: • URL to application's privacy policy. • wireframes sketched of the application. • step-by-step user guide. • code commented with KDoc & generated with Dokka. • REST API GET endpoints. • URL to application on Google Play Store. Git branches clearly named with convention & contain code relating to the feature. Git commit messages clearly formatted & reflect the feature changes in substantial detail. Continuous integration using GitHub Actions clearly setup.	README file contains evidence of: • URL to application's privacy policy. • wireframes sketched of the application. • step-by-step user guide. • code commented with KDoc & generated with Dokka. • REST API GET endpoints. • URL to application on Google Play Store. Git branches named with convention & contain code relating to the feature. Git commit messages formatted & reflect the feature changes in detail. Continuous integration using GitHub Actions setup.	README file does not or does not fully contain evidence of: • privacy policy discloses user information collected. • wireframes sketched of the application. • step-by-step user guide. • code commented with KDoc & generated with Dokka. • REST API GET endpoints. • URL to application on Google Play Store. Git branches are not or are not fully named with convention & do not or do not fully contain code relating to the feature. Git commit messages do not or do not fully formatted & reflect the feature changes. Continuous integration using GitHub Actions not or not fully setup.

Project Marking Cover Sheet

Name:

Date:

Learner ID:

Assessor's Name:

Assessor's Signature:

Criteria	Out Of	Weighting	Final Result
Functionality	10	40	
Code Elegance	10	45	
Documentation & Git/GitHub Usage	10	15	
Final Result			/100
This assessment is worth 80% of the final mark for the Mobile Application Development course.			

Feedback: