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**. **Android** topics 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** features & other application development topics independently. In addition, marks will be allocated for code elegance, documentation & **Git/GitHub** usage.

The travelling application will help you sound like a local & adapt to a new culture. You will begin by selecting a country you wish to travel to. For example, if you wish to travel to Italy, you would be provided with all the necessary tools such as text translation & text to speech support, a selection of well-known Italian phrases, an interactive quiz to test your knowledge of Italian culture & a map containing locations of Italy's top-rated tourist attractions. A user of your travelling application must be able to select from at least two countries 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/2KFNPhsK. 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 one language. For example, Canada's main languages are English & French. You would choose either English or 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. Ensure that the **API key** is not publicly exposed in your program files. Alternatively, you can use the **Google Translate API**.
  - 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 well-known phrases. For example, "No worries, mate, she'll be right" is a well-known Australian phrase.
- An interactive quiz for each country.
  - Quiz data can be fetched from an API using **Retrofit** or a **JSON** file stored in the **raw** resource directory.
  - 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 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 an API using **Retrofit** or a **JSON** file stored in the **raw** resource directory.
  - Each data object will be represented with 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 on 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 animated **ImageView**.
  - **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 various features in the application. 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**.
- 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.xml** or **dimens.xml** file & strings must be stored in a **strings.xml**.
- Use of intermediate variables. No method calls as arguments.
- Idiomatic use of control flow, data structures & in-built functions.
- Code adheres to **DRY**, **KISS** & **SOLID**.
  - **Resources:**
    - https://dzone.com/articles/software-design-principles-dry-and-kiss
    - https://digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
- Efficient algorithmic approach.
- **Kotlin** & **XML** files are code formatted.
- No dead or unused code.

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. You will create a webpage which serves your 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.
    - **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://github.com/Kotlin/dokka
      - https://kotlinlang.org/docs/reference/kotlin-doc.html
- Continuous integration using **GitHub Actions**.
  - **YAML** file must be configured for UI tests and generating an APK.
  - **Resources:**
    - https://atlassian.com/continuous-delivery/continuous-integration
    - https://docs.github.com/en/actions
    - 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://freecodecamp.org/news/writing-good-commit-messages-a-practical-guide