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** 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 implement intermediate/advanced **Android** features & other application development topics. 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
Practicals	10%	2, 3	CRA	Cumulative
Project	70%	1, 2, 3	CRA	Cumulative
Presentation	20%	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 has **two** due dates:

- GitHub Gist data & sketched wireframes - **27/09/2021**
- Application and other documentation - **19/11/2021**

Pass Criteria

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

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 & 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/YvRoxlAh. The latest program files in the **main** branch will be used to mark against the **Functionality & Robustness** criterion. Please test your **main** branch application before you submit. Partial marks **are not** given for functionality in other branches. 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 (Features) - Learning Outcomes 1, 2, 3 (40%)

- Application **must** open without file structure modification in **Android Studio**.
- Application **must** run without code modification on a mobile device.
- 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.
  - Display some feedback while the text is being translated.
  - Handle incorrectly formatted input fields. For example, an **EditText** is blank or empty.
- Text to speech support.
  - If a country is not supported, handle gracefully with a **Toast**.
- Selection of three well-known phrases. For example, "No worries, mate, she'll be right" is well-known phrase in Australia.
- An interactive quiz for each country.
  - Quiz data **must** be fetched from a **GitHub Gist**.
  - Quiz topics may include animals, culture, food, drink, geography & sport.
  - Each quiz **must** have at least five questions.
  - Questions are multi-choice & **must** have four answers.
  - Each question **must** have an image.

  - Display appropriate feedback in a **Toast** when a question is answered correctly or incorrectly. If an answer is incorrect, display the correct answer.
  - A quiz **must** be completed within a **3 minute** time limit.
  - At the end the quiz, store the score in a **Room Database** table.
  - Display the highest score in a **TextView**.
- **Localization** support for each country.
- **Google Map** displaying top-rated tourist attractions.
  - Top-rated tourist attraction data **must** be fetched from a **GitHub Gist**.
  - Each data object will represent a marker.
  - The marker's information window **must** display the attraction's name & city/town.
- **Switch** which toggles between light & dark mode.
  - The state (true or false) value of the **Switch must** be stored in a **DataStore**.
  - The mode will be based off the state value of the **Switch**. For example, true equals dark mode & false equals light mode.
- Splash screen which uses a **Lottie** animation.
- Adaptive launcher icon which is displayed in a variety of shapes.
- **BottomNavigationView** which navigates the user to various features in the application. For example, a menu icon for translation support, text to speech support, etc.
- 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.
- Ability to download your application from **Google Play Store** on to a mobile device.
- At least **10** UI tests which verify that your application is functioning correctly.

Code Elegance - Learning Outcomes 1, 3 (40%)

- **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**.
- Idiomatic use of control flow, data structures & in-built functions.
- Code adheres to **DRY**, **KISS** & the **Model-View-ViewModel** architectural pattern.
- Efficient algorithmic approach.
- **Kotlin** & **XML** files are code formatted.
- Unused code & resources.

Documentation & Git/GitHub Usage - Learning Outcomes 2, 3 (20%)

- Provide the following in your repository **README.md** file:
  - URL to your application's privacy policy.
  - Sketched wireframes of your application. The wireframes **must** be sketched/designed using online software.
  - 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**.
  - URL to your application on **Google Play Store**.
- Continuous integration using **GitHub Actions**.
  - **YAML** file **must** be configured, UI tests, code formatting, linting & generating an APK.
- 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.