College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
ID721001: Mobile Application Development
Level 7, Credits 15

Project

Assessment Overview

In this **individual** assessment, you will develop two mobile applications using **React Native** and **Expo**. In addition, marks will be allocated for code elegance, documentation and **Git/GitHub** usage. For Part 3, you will research, prepare and present a **React Native** UI library. The information presented must be in a **README.md** file. Also, you need to provide code examples to accompany the **README.md** file. The main purpose of this assessment is to demonstrate your ability to identify and effectively articulate your findings.

Learning Outcomes

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

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

Assessments

Assessment	Weight	Due Date	Learning Outcomes
Portfolio	100%	21-06-2024 (Friday at 4.59 PM)	1, 2, 3

Conditions of Assessment

You will complete majority of this assessment during your learner-managed time. However, there will be time during class to discuss the requirements and your progress on this assessment. Part 1 will need to be completed by **Friday, 01 September 2023** at **4.59 PM** and Part 2 and 3 will need to be completed by **Friday, 10 November 2023** at **4.59 PM**.

Pass Criteria

This assessment is criterion-referenced (CRA) with a cumulative pass mark of **50%** over all assessments in **ID721001: Mobile Application Development**.

Authenticity

All parts of your submitted assessment **must** be completely your work. Do your best to complete this assessment without using an **AI generative tool**. You need to demonstrate to the course lecturer that you can meet the learning outcome(s) for this assessment.

However, if you get stuck, you can use an **AI generative tool** to help you get unstuck, permitting you to acknowledge that you have used it. In the assessment's repository **README.md** file, please include what prompt(s) you provided to the **AI generative tool** and how you used the response(s) to help you with your work. It also applies to code snippets retrieved from **StackOverflow** and **GitHub**.

Failure to do this may result in a mark of **zero** for this assessment.

Policy on Submissions, Extensions, Resubmissions and Resits

The school's process concerning submissions, extensions, resubmissions and 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.

Submission

You **must** submit all project files via **GitHub Classroom**. Here are the URLs to the repositories you will use for your submission:

- Part 1 - https://classroom.github.com/a/pWdicmvU
- Part 2 - https://classroom.github.com/a/d_u8vXR5

Create a **.gitignore** and add the ignored files in this resource - https://raw.githubusercontent.com/github/gitignore/main/Node
The latest project files in the **master** or **main** branch will be used to mark against the **Functionality** criterion. Please test before you submit. Partial marks **will not** be given for incomplete functionality. 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 and 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 and achieved a **D grade (40-49%)**. The maximum grade awarded for resubmission will be **C-**.

Resits

Resits and reassessments **are not** applicable in **ID721001: Mobile Application Development**.

Instructions

You will need to submit a mobile application and documentation that meet the following requirements:

1 Part 1: Cookbook Application (20%)

Functionality - Learning Outcomes 1, 2, 3 (50%)

- The mobile application needs to run without code or file structure modification in **Visual Studio Code**.
- Usable on a variety of mobile devices, i.e., devices with different screen sizes.
- Free of bugs that significantly effect the usability.
- Food data needs to be fetched using **Axios** from a **GitHub Gist**. You have been provided an example file called **food-data.json**.
- Display **bottom tab navigation** with the following screens:
  - Daily specials
    - This screen will display six random recipes from the **food-data.json** file.
    - Display the random recipes in a **FlatList**.
    - Each recipe item in the **FlatList** needs to display the recipe's name and image. Truncate the recipe's name if it is too long.
    - When a recipe item is pressed, display the recipe's name, image, cuisine, ingredients and instructions in a **ScrollView**.
  - Recipes
    - This screen will display all cuisines from the **food-data.json** file.
    - When a cuisine item is pressed, display all recipes for that cuisine in a **FlatList**.
    - Each recipe item in the **FlatList** needs to display the recipe's name, description and image. Truncate the recipe's name and description if it is too long.
    - When a recipe item is pressed, display the recipe's name, image, cuisine, ingredients and instructions in a **ScrollView**.
  - A heart icon needs to be displayed in the top right corner of the screen. When the heart icon is pressed, the recipe is added to the **Favourites** screen. Persist the favourite recipes using **AsyncStorage**.
  - A plus icon needs to be display next to the heart icon. When the plus icon is pressed, the recipe's ingredients are added to the **Shopping list** screen. Persist the shopping list using **AsyncStorage**.
  - Favourites
    - This screen will display all recipes that have been added to the **Favourites** screen.
    - Display the favourite recipes stored in **AsyncStorage** in a **FlatList**.
    - Ability to delete a favourite recipe from the **FlatList**.
  - Shopping list
    - This screen will display all ingredients from the recipes that have been added to the **Shopping list** screen.
    - Display the shopping list stored in **AsyncStorage** in a **FlatList**.
    - Ensure there are no duplicate ingredients in the **FlatList**.
    - Ability to delete an ingredient from the **FlatList**.
  - Appropriate image used for the splash screen and app icon.
  - Visually attractive UI with a coherent graphical theme and style.

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

- A **Node.js .gitignore** file is used.
- If applicable, a **.env** and **.env.example** file is used.
- Appropriate naming of files, variables, functions and components.
- Idiomatic use of control flow, data structures and in-built functions.
- Efficient algorithmic approach.
- Sufficient modularity.
- Each **component** file **must** have a **JSDoc** header comment located immediately before the **import** statements.
- In-line comments where required. It should be for code that needs further explanation.
- Code is formatted.
- No dead or unused code.

Documentation and Git/GitHub Usage - Learning Outcomes 2, 3 (10%)

- **GitHub** project board to help you organise and prioritise your work.
- Provide the following in your repository **README.md** file:
  - Wireframes of the mobile application's screens. The wireframes can be either hand-drawn or created using a digital tool.
  - How do you setup the environment, i.e., after the repository is cloned?
  - How do you format your code?
  - If applicable, known bugs.
- Use of **Markdown**, i.e., headings, bold text, code blocks, etc.
- Correct spelling and grammar.
- Your **Git commit messages** should:
  - Reflect the context of each functional requirement change.
  - Be formatted using an appropriate naming convention style.

2 Part 2: Traveling Application (40%)

Scenario: The mobile application will help you sound like a local and 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 and text to speech support, a selection of well-known Italian phrases and a map containing locations of Italy's top-rated tourist attractions. A user of your mobile application **must** be able to select from at least **six** countries with at least **one** country per continent **excluding** Antarctica.

Functionality - Learning Outcomes 1, 2, 3 (50%)

- The mobile application needs to run without code or file structure modification in **Visual Studio Code**.
- Usable on a variety of mobile devices, i.e., devices with different screen sizes.
- Free of bugs that significantly effect the usability.
- The mobile application is published to **Google Play Store** or **Apple App Store**.
  - To published to **Google Play Store** or **Apple App Store**, you will need an account. The account's credentials will be privately given to you on **Microsoft Teams**. **Do not** disable any applications published on this account.
- Ability to download the mobile application from **Google Play Store** or **Apple App Store** on to a variety of mobile devices.
- Country data needs to be fetched using **Axios** from a **GitHub Gist**. You have been provided an example file called **country-data.json**. **Note:** This file is incomplete. You will need to add more information to the file.
- Display a list of countries in a **FlatList**. Each item needs to have the country's name and flag. When you select a country, it will navigate to the country's screen containing the following:
  - 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 and French. You would choose either English or French.
    - Use **Axios** and 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 will be privately given to you on **Microsoft Teams**.
    - Display some feedback while the text is being translated.
    - Handle incorrectly formatted input fields. For example, an **TextInput** is blank or empty.
  - Text to speech support.
    - Handle incorrectly formatted input fields. For example, an **TextInput** is blank or empty, or the country is not supported.
  - Selection of two well-known phrases. For example, "No worries, mate, she'll be right" is well-known phrase in Australia.
  - **Google Map** displaying top-rated tourist attractions.
    - Display two tourist attractions per continent.
    - Each data object will represent a marker.
    - The marker's information window needs to display the attraction's name and city/town.
  - Display an image gallery of top-rated tourist attractions. Maximum of four images per attraction.
  - A **WebView** containing the country's Wikipedia page.
  - An interactive quiz for each country.
    - Quiz topics may include animals, culture, food, drink, geography and sport.
    - Each quiz needs to have five questions.
    - Questions are multi-choice and need to have four answers.
    - Each question needs to have an image.
    - Display appropriate feedback in a **TextInput** when a question is answered correctly or incorrectly. If an answer is incorrect, display the correct answer.
    - At the end the quiz, display the score in a **TextInput** and store the score in **Firestore** on **Firebase**.
- Appropriate image used for the splash screen and app icon.
- Visually attractive UI with a coherent graphical theme and style.

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

- A **Node.js .gitignore** file is used.
- If applicable, a **.env** and **.env.example** file is used.
- Appropriate naming of files, variables, functions and components.
- Idiomatic use of control flow, data structures and in-built functions.
- Efficient algorithmic approach.
- Sufficient modularity.
- Each **component** file **must** have a **JSDoc** header comment located immediately before the **import** statements.
- In-line comments where required. It should be for code that needs further explanation.
- Code is formatted.
- No dead or unused code.

Documentation and Git/GitHub Usage - Learning Outcomes 2, 3 (10%)

- **GitHub** project board to help you organise and prioritise your work.
- Provide the following in your repository **README.md** file:
  - Link to the mobile game on **Google Play Store** or **Apple App Store**.
  - Wireframes of the mobile application's screens. The wireframes can be either hand-drawn or created using a digital tool.
  - How do you setup the environment, i.e., after the repository is cloned?
  - If applicable, known bugs.
- Use of **Markdown**, i.e., headings, bold text, code blocks, etc.
- Correct spelling and grammar.
- Your **Git commit messages** should:
  - Reflect the context of each functional requirement change.
  - Be formatted using an appropriate naming convention style.

3 Part 3: Presentation (20%)

List of UI libraries:

- NativeBase
- Teaset
- Material Kit React Native
- React Native Elements
- React Native Paper
- Lottie for React Native
- Reatct Native UI Kitten
- React Native Material Kit
- React Native Material UI
- RNUILib

Documentation - Learning Outcomes 2, 3 (50%)

- Documentation must contain the following sections:
  - Overview - a brief description of what the UI library is.
  - How to install - a description of how to install the UI library.
  - Code examples - provide five code examples of how to use the UI library.
  - References - the information in your documentation is referenced using **APA 7th edition**.
    - **Resource:** https://studentservices.op.ac.nz/learning-support/citingandreferencing
- Use of **Markdown**, i.e., bold text, code blocks, etc.
- Correct spelling and grammar.

Presentation 2, 3 (50%)

- Present your documentation, i.e., **README.md** via a video recording. In addition, you **must**:
  - Upload your presentation to your **OneDrive**.
  - Provide a link to your presentation in your documentation.

Additional Information

- **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.
- You need to provide the wireframes to the course lecturer before you begin development.
- The presentation must not exceed **30 minutes** in length.
- Upload your presentation to **OneDrive**. Email a link to your presentation to grayson.orr@op.ac.nz.