The App

A **Car Rental Company** is expanding its offerings to include weekly rentals of used/new cars. They are seeking a simple proof-of-concept app for clients to use.

A. Requirements

- The app should display **five cars** that the company can rent. Note, you can deal in **"Credit"**, not dollars necessarily.
- Each vehicle should be displayed one at a time; a **"Next"** button is needed to progress through the cars. **Note:** Assume the app only allows for immediate car pickup; there is no concept of "future" bookings.
- The data kept for each car should include *the car name, model, year, a rating on a scale of 1-5, the number of kilometres,* and *the daily rental cost of the car*. You will need to create a model for this; note that the data should not be stored on disk, but instead kept in memory.
- The car details should be shown under each image in the first activity.
- You should include only free images and avoid copyrighted images of vehicles.

1. Renting & Credit Balance

- When the **'Rent'** button is clicked, a new screen displaying further details should be shown, potentially along with the image. The data needs to be passed as a `Parcelable` object. In your report, explain the components of your Intent and the advantages of using `Parcelable` objects on Android.
- Once a car is rented, it should be removed from the available list. If a booking is cancelled, the vehicle should be made available again.
- Each user has a **Credit Balance** of 500 credits, and each car rental should not exceed 400 credits. Enforce a credit limit check for the combined total.
- When renting a car, the credit balance decreases accordingly.
- If the balance is insufficient, an **error message** prevents booking.
- Balance should be shown in the app header or a small widget at the top of the screen.
- The new details should be saved upon pressing **'Save'** (pressing **'Back'** should be considered a cancellation) and should be displayed to the user in the first activity in some way. ***Do not use persistent data for this task.***
- When a car is booked or not booked (e.g., the booking action is cancelled), a `Toast` or `Snackbar` is required to denote this.

2. Favorites

- Users can mark a car as a Favourite with a *heart icon (using an external image or widget), a button, or by long pressing the image.*
- Favourites appear in a separate section on the home screen for quick access.

3. Search & Sort

- Add a **Search bar** to the main screen, allowing users to find cars by *name or model* quickly. (Hint: there is a widget for that)

- Add a **Sort** menu to reorder displayed vehicles by:
    - ➢ Rating (high to low)
    - ➢ Year (newest to oldest)
    - ➢ Cost (low to high)
- Sorting should apply across the in-memory dataset without changing the base data.

4. Dark Mode Toggle

- Add a **Switch or Toggle** at the top of the page to switch between **Light Mode** and **Dark Mode**.
- Styles should adapt accordingly (applies to at least two UI screens).

B. Technical Instructions and Report

- At least **two** of your attributes should utilise a `non-TextView` widget, such as a `RatingBar` for reviews, a `Switch` for different models of the same car name based on year, or a `Slider` to select the number of days (with a minimum of 1 and a maximum of 7).
- This is a good app for UI testing. Note that `RatingBars` are not easily testable with Espresso, so focus on testing text and buttons instead. If you use Espresso, then it is expected that the code will be edited to remove redundancy.
- Error checking should be included for both required and incorrect fields — this could involve verifying required details, checking credit (to ensure it does not exceed the credit limit), etc. This must stop the user from returning to the first activity until the error is fixed or they cancel the booking.
- In Module 5, we examined UX. Please include two user stories and use cases for your app, along with **sketches** of two possible layouts for your app screens, noting the required widgets. Please note that this does not mean "pretty" designs should be made, nor complete prototypes -- it is simply testing whether you can use sketching tools to think about and communicate ideas about your app's UI. Consider different UI elements for each of the two layouts and justify your choices.
- Use at least two styles in two locations -- that is, don't make a style just for one view; use it in multiple locations. This could be for defining *text size, style, colour or something else*. You can use this point in conjunction with the Dark Mode toggle feature mentioned above.
- Discuss your design choices in your report.
- The Challenges, explorations, and takeaways should be included in the report.
- Note that while some concepts from A1 are not explicitly required for A2, they may still be helpful for this task and could be implemented with justification.

Resources

Weeks 1-6 have set you up by isolating different parts of app development needed for this task.

Some further resources that might be useful include:

- For more information on RatingBar, refer to https://developer.android.com/reference/android/widget/RatingBar for those who wish to use it. To constrain the number of stars shown, check the layout_width setting.
- For using the Search bar, see https://developer.android.com/develop/ui/views/search/training/setup.
- For Toggle, see https://developer.android.com/develop/ui/views/components/togglebutton?hl=en
- For sorting, use the built-in functions of Kotlin or create your own custom sorting functions.
- For using a Slider, see https://m3.material.io/components/sliders/overview.
- For information on Toasts, see https://developer.android.com/guide/topics/ui/notifiers/toasts.
- For Snackbars, see https://developer.android.com/develop/ui/views/notifications/snackbar.
- See https://developer.android.com/guide/topics/ui/look-and-feel/themes for guidance on using styles.
- Students may wish to explore the use of Fragments; however, demonstrating communication between fragments and activities is required, along with a written comparison to activity-activity communication. You may use Intents for a feature not listed above (e.g., email confirmation).