**Assignment 2: Rent a Car with Intent**

*(COS30017 – Mobile Computing)*

**Student Name:** [Your Name]  
**Student ID:** [Your ID]  
**Date:** [Submission Date]

Table of Contents

[**1.** **Introduction** 2](#_Toc210460198)

[**2.** **Planning and Research** 2](#_Toc210460199)

[**2.1.** **Objectives** 2](#_Toc210460200)

[**2.2.** **Research Highlights** 2](#_Toc210460201)

[**2.3.** **Development Tools** 2](#_Toc210460202)

[**3.** **Design Decisions** 2](#_Toc210460203)

[**3.1.** **User Interface** 2](#_Toc210460204)

[**3.2.** **Widget Choices Table** 3](#_Toc210460205)

[**3.3.** **Parcelable Usage** 3](#_Toc210460206)

[**4.** **Development Logs** 4](#_Toc210460207)

[**4.1.** **Time Log Table** 4](#_Toc210460208)

[**5.** **Testing** 4](#_Toc210460209)

[**5.1.** **Test Strategy** 4](#_Toc210460210)

[**5.2.** **Test Results Table** 4](#_Toc210460211)

[**6.** **Challenges, Explorations & Takeaways** 5](#_Toc210460212)

[**6.1.** **Challenges** 5](#_Toc210460213)

[**6.2.** **Explorations** 5](#_Toc210460214)

[**6.3.** **Takeaways** 5](#_Toc210460215)

[**7.** **Reflections on Assignment 1 & Changes for Assignment 2** 5](#_Toc210460216)

[**Conclusion** 5](#_Toc210460217)

[**Appendices** 6](#_Toc210460218)

# **Introduction**

This report documents the development of a proof-of-concept car rental mobile app ("RentACar") that leverages Jetpack Compose for a modern, declarative UI approach. The app demonstrates multi-activity usage with intent-based data sharing, incorporating Parcelable objects for efficient data transfer between screens. It allows users to explore a range of five vehicles, examine details individually, rent cars within credit restrictions, save favorites, search by name or model, organize by different criteria, and switch between light and dark themes. All information is handled in-memory without long-term storage, consistent with the task's emphasis on real-time bookings and the straightforward nature of the proof-of-concept

The report elaborates on the planning process, encompassing user stories and use cases, outlines significant design choices with reasons, records development tasks, showcases testing methods and outcomes, addresses difficulties faced and investigations conducted, and assesses insights gained from Assignment 1. Ultimately, it comprises conclusions and a references section for cited materials.

1. **Planning and Research**
   1. **Objectives**

* Develop a two-activity mobile app using Intents and Jetpack Compose for UI.
* Implement Parcelable objects for efficient inter-activity data transfer.
* Integrate dynamic UI components like Slider, Switch, and custom RatingBar.
* Ensure credit limit handling, favorites tracking, search, and sort functionalities.
* Support dark mode toggling with adaptive styles.

## **Research Highlights**

* Investigated Jetpack Compose as the recommended modern UI toolkit for Android, focusing on its declarative nature for better performance and maintainability compared to traditional XML-based views.
* Reviewed Parcelable vs. Serializable for data passing, opting for Parcelable due to its optimization for Android's IPC mechanisms.
* Explored Material Design 3 (M3) components in Compose for sliders, switches, and search bars to ensure a consistent, user-friendly interface.
* Studied Espresso for UI testing, adapting strategies for Compose-specific interactions like button clicks and text assertions.
* Researched in-memory data management with repositories to simulate a simple dataset without disk storage.

## **Development Tools**

* Android Studio Narwhal Feature Drop | 2025.1.2 Patch 2
* Kotlin 2.0.21
* Jetpack Compose
* GitHub Classroom for version control
* Emulated device Pixel 9 Pro API 36.0
  1. **UX Considerations: User Stories and Use Cases**

Two user stories and corresponding use cases were developed to guide the app's design. These focus on core functionalities: browsing/renting cars and managing preferences.

**User Story 1:** As a user with limited credits, I want to browse available cars one at a time and check my balance easily so that I can make informed rental decisions without overwhelming the screen.

* **Use Case:** The user launches the app and sees the first car with details (name, model, year, rating, kilometers, cost). They use "Next" and "Previous" buttons to cycle through cars. The credit balance is displayed in the top bar. If credits are insufficient for a car, the "Rent Now" button is disabled, preventing invalid actions.
* **Preconditions:** App is installed; in-memory car data is loaded.
* **Postconditions:** User navigates cars; balance is visible; proceeds to rental if affordable.

**User Story 2:** As a user in low-light conditions, I want to toggle dark mode and mark favorites for quick access so that I can comfortably use the app and revisit preferred cars.

* **Use Case:** The user toggles the dark mode icon in the top bar, switching themes across activities. They long-press or tap the heart icon on a car to favorite it, which appears in a filtered section on the main screen for quick access.
* **Preconditions:** App is open; cars are available.
* **Postconditions:** Theme changes persist during session; favorites are updated in-memory.

# **Design Decisions**

* 1. **User Interface**
* Main activity displays one car at a time with image, details, and navigation buttons.
* RentalDetails activity handles booking with a slider for rental days and input fields.
* Credit balance shown in the top app bar with a Text composable.

This screen displays the main activity of the app, where users can view one car at a time, browse using the “Next/Previous” buttons, and see their credit balance at the top. Users can search for cars by name or model, sort by rating, year, or cost, and toggle between Light and Dark mode. Favorite cars are listed in a separate section for quick access. The wireframe (left) illustrates layout placement, while the high-fidelity mockup (right) shows the styled app interface.

|  |  |
| --- | --- |
| **A screenshot of a phone  AI-generated content may be incorrect.** | **A screenshot of a car  AI-generated content may be incorrect.** |

Figure 1-Main Activity - Car List and Favorites

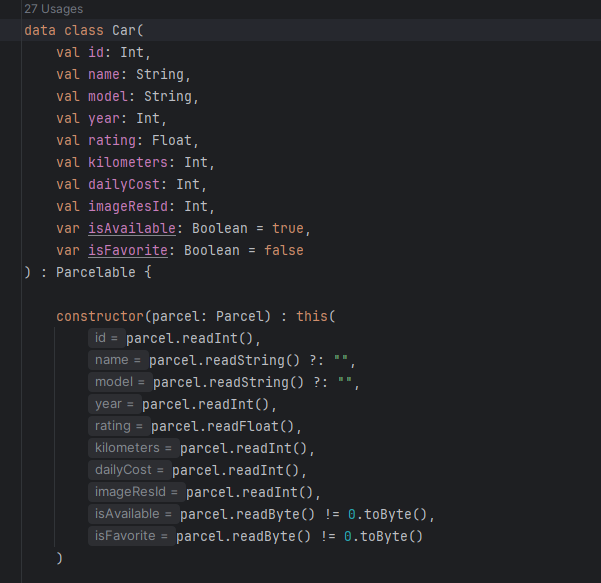
This screen shows the Rental Details Activity where the user sees detailed information about the selected car, including name, year, cost, and features. The slider allows users to select rental duration from 1 to 7 days. The wireframe (left) shows input layout and widget placement, while the high-fidelity mockup (right) demonstrates the fully styled interface.

|  |  |
| --- | --- |
|  |  |
|  |  |

Figure 2 - Rental Details Activity - Car Selection and Rental Duration

## **Widget Choices Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Widget** | **Purpose** | **Justification** | **Locations Used** |
| RatingBar | Display car rating | Custom Compose implementation | Main Activity |
| Switch | Toggle insurance option | User-controlled option | Rental Details |
| Slider | Number of rental days (1–7) | Prevents invalid input | Rental Details |
| Button | Navigation / Rent / Cancel | Standard action trigger | Both Activities |
| TextField | Customer input | Input validation | Rental Details |
| Toast | Booking Success/error | Immediate Feedback | Both Activities |

* 1. **Parcelable Usage**
* The Car class is implemented as a Parcelable object for data transfer.
* 
* Advantages: Faster than Serializable, lower memory overhead, optimized for Android IPC.
* Intent example in MainActivity.kt:

A screen shot of a computer

AI-generated content may be incorrect.

* 1. **Architectural Choices: Jetpack compose vs. Traditional Views**

A key decision was adopting Jetpack Compose over XML-based views for UI construction. This aligns with modern Android development, offering declarative syntax that reduces boilerplate and improves state management. For instance, state like currentCarIndex and isDarkMode is handled reactively with mutableStateOf, automatically triggering recompositions. This choice was justified by Compose's performance benefits (no XML inflation) and better integration with Kotlin, though it required learning new paradigms for widgets like a custom RatingBar composed of Icons.

Two styles were applied: AppTextStyles.carTitleStyle for bold titles (used in both activities) and AppTextStyles.carDetailStyle for subtitles (applied to year, rating, etc.), ensuring consistency and reusability.

# **Development Logs**

## **Time Log Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Task** | **Duration** | **Notes** |
| 2025/09/26 | Project setup, Compose config | 2h | Initialized repository |
| 2025/09/27 | Car model & Repository creation | 3h | Defined Parcelable and data |
| 2025/09/28 | Main activity UI | 4h | Implemented car display |
| 2025/09/29 | Rental activity UI | 3h | Added slider, input validation |
| 2025/09/30 | Favorites, Search, Sort | 4h | Added filtering and sorting |
| 2025/10/01 | Dark mode and styling | 3h | Applied adaptive themes |
| 2025/10/02 | Testing and debugging | 5h | Verified UI and credit logic |

# **Testing**

## **Test Strategy**

* Manual testing for navigation and UI consistency.
* Automated testing with Espresso for text and button interactions.
* Credit validation and error handling.
* Favorites, search, and sort functionality.

## **Test Results Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Input** | **Expected Outcome** | **Actual Outcome** | **Pass/Fail** |
| Rent car under balance | 200 credits | Booking succeeds | Matches expectation | Pass |
| Rent car exceeding balance | 600 credits | Booking blocked, error shown | Matches expectation | Pass |
| Cancel booking | Cancel pressed | Car available, balance unchanged | Matches expectation | Pass |
| Mark favorite | Tap heart icon | Car in favorites list | Matches expectation | Pass |
| Search car by name | "Toyota" | Filtered list shows Toyota | Matches expectation | Pass |
| Sort by rating | "High to Low" | Cars ordered by rating | Matches expectation | Pass |
| Dark mode toggle | Switch enabled | UI changes to dark mode | Matches expectation | Pass |
| Number of rental days | Slider 1-7 | Only valid selection allowed | Matches expectation | Pass |
| Input validation | Empty name/phone | Error prevents confirmation | Matches expectation | Pass |
| Navigation cycle | Next/Previous buttons | Cycles through available cars | Matches Expectation | Pass |

# **Challenges, Explorations & Takeaways**

## **Challenges**

* Handling reactive state in Compose: Initial issues with recomposition when updating favorites or availability led to stale UI; resolved by using mutableStateOf and refresh triggers.
* Parcelable implementation: Ensuring all fields (including booleans for availability/favorites) were correctly parceled without data loss during activity transitions.
* Testing limitations: Espresso struggles with custom Compose components like RatingBar, requiring focus on text assertions and button interactions.
* Credit limit enforcement: Integrating real-time checks across activities without persistent storage required careful repository design.
* Dark mode adaptation: Ensuring custom icons (e.g., heart, sun/moon) tinted correctly in both modes, addressing color contrast for accessibility.

## **Takeaways**

* Jetpack Compose improves productivity with declarative syntax.
* Early state management planning reduces debugging time.
* Espresso testing requires careful widget selection.

# **Reflections on Assignment 1 & Changes for Assignment 2**

* Assignment 1 focused on XML-based single-activity apps.
* Assignment 2 shifts to Compose with multi-activity and advanced features:
  + Intent-based data transfer with Parcelable.
  + Custom Compose widgets (RatingBar, Slider).
  + Favorites, search, and sort.
  + Dark mode with adaptive styling.
* Lessons Learned: Compose simplifies UI development but requires robust state management.

# **Conclusion**

The app demonstrates a functional car rental experience with:

* Efficient data sharing using Parcelable objects.
* Responsive UI with custom Compose components.
* Credit management and error handling.
* Favorites, search, and sort functionality.
* Light and dark mode themes.

# **Appendices**

Appendix A – Acknowledgement of Generative AI Use

No Generative AI tools were used for this task.

Appendix B – Screenshots / Images

App with dark mode enabled

A screenshot of a car

AI-generated content may be incorrect.A screenshot of a car rental application

AI-generated content may be incorrect.

Appendix C – Source Code Link  
[Insert GitHub Classroom repository link]