Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Developing a Trip Planning Application with a Focus on People with Disabilities

Текстова частина до кваліфікаційної роботи
за спеціальністю 121 “Інженерія програмного забезпечення”

Керівник курсової роботи
Андрощук М.В.
_____
(підпис)
“___” _____ 2025 р.

Виконав студент
Орлов С.О.
“___” _____ 2025 р.

Київ 2025

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри мультимедійних систем,
проф., д.ф.-м.н.
_____О. П. Жежерун
(підпис)
«____»_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу

студенту 4-го року БП ІПЗ факультету інформатики
Орлову Станіславу Олеговичу

ТЕМА: Developing a Trip Planning Application with a Focus on People with Disabilities
Зміст ТЧ до курсової роботи:
Анотація
Introduction
Overview Of Accessibility in Available Solutions
Accessibility Barriers in Travel Planning Apps
Review Of Used Technologies
Overview Of Implementation’s Functionality
Conclusions
References

Дата видачі “___” _____ 2024 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

ТЕМА: **Developing a Trip Planning Application with a Focus on People with Disabilities**

Календарний план виконання роботи:

№ п/п	Назва етапу кваліфікаційної роботи	Термін виконання етапу	Примітка
1	Аналіз та узгодження теми роботи	01.09.2024 — 23.02.2025	
2	Огляд технічної документації за темою роботи	16.10.2024 — 04.03.2025	
3	Аналіз проблематики та наявних рішень	18.01.2025 — 21.04.2025	
4	Розробка і тестування	01.03.2025 — 24.05.2025	
6	Захист роботи	02.06.2024 — 04.06.2025	

Студент Орлов С.О.

Керівник Андрощук М.В.

«_____» 2024

ЗМІСТ

АНОТАЦІЯ

Ця робота демонструє етапи розробки організаційної системи для планування подорожей, що враховує потреби людей з інвалідністю, у вигляді Android-застосунку. Описано аналіз наявних подібних рішень на предмет інклюзивності, виявлено та категоризовано їхні бар'єри у забезпеченні інклюзивності, викладено функціонал створеної системи, а також обґрунтовано вибір програмних засобів та технологій, використаних у роботі.

Ключові слова: inclusive software development, Android accessibility tools, mobile application, trip planner, people with disabilities.

INTRODUCTION

Recent data shows a significant growth in the mobile travel applications market. In 2023, about 63% of all global travel bookings were made via mobile platforms (Curry, 2025). This indicates a trend towards increasing the use of mobile devices for travel planning, which is associated with the convenience and accessibility of these services. In addition, in the same year of 2023, the number of downloads of travel apps increased by 13% compared to 2022, and users continue to spend more time in such apps, which increases their engagement (Motta, 2023).

However, not everybody can equally experience those applications. According to modern data, an estimated 1.3 billion people experience significant disability (World Health Organization, 2023). This represents 16% of the world's population, or 1 in 6 of us. This information emphasizes the importance of designing software products with inclusivity in mind.

Accessibility ensures that individuals with limited vision, hearing, motor skills, or cognitive capabilities can interact with digital products effectively, like users without such impairments, and by incorporating accessibility features into mobile applications, developers significantly expand the potential user base and improve the daily experience of people with disabilities (Chang & Tuli, 2021). Moreover, designing with accessibility in mind is an investment in the long-term usability of digital systems. As people age, their visual acuity and motor coordination naturally decline, meaning today's standard user may become tomorrow's accessibility user. Thus, accessible design also serves the needs of everybody's future selves (Milbergs, 2024).

Importantly, accessibility measures do not solely benefit users with permanent disabilities. They also improve the experience for users facing situational limitations — such as using a smartphone under bright sunlight, in noisy environments, or while one hand is occupied (Chang & Tuli, 2021). Designing for these situations leads to more resilient and user-friendly interfaces overall.

These principles and techniques were carefully studied and integrated during the development of the inclusive trip planner presented in this thesis. The decision to develop an inclusive travel planning application is grounded in both societal relevance and practical research value. Travel applications are essential tools for a wide demographic, enabling users to organize their mobility, leisure, and business activities. However, many of these applications remain inaccessible to people with disabilities, limiting their independence and full participation in everyday life.

Despite the critical role travel autonomy plays in fostering social inclusion and the way it can contribute to overall well-being (Friman & Olsson, 2023), travel-related applications remain underrepresented in accessibility research. This thesis addresses that gap by focusing on core functionalities that are most frequently used and most essential: creating events, managing packing lists, and attaching digital tickets. These features reflect common travel behaviours and pose real-world accessibility challenges, such as handling attachments, dynamic content, and complex forms.

By developing these functionalities in full compliance with Web Content Accessibility Guidelines (later abbreviated to as WCAG) (W3C Recommendation, 2024) and Android accessibility guidelines (Google, n.d.-c), this research demonstrates how inclusive design can be effectively applied in a domain where usability and accessibility directly impact the quality of travel experiences.

This thesis's goal is to fill the gap in the currently available resources designed to assist people with disabilities in enhancing their trip planning experience.

The object of the thesis is inclusive mobile development.

The subject of the thesis is various tools and methods to ensure the principles of developing an inclusive Android travel planning application.

To achieve the set goal, the following tasks were completed:

- studied current standards for digital accessibility which should be considered when developing interfaces;
- conducted an analysis of competitors' applications focusing on several types of disabilities (motor, visual, etc.);
- developed an application ensuring compatibility with assistive technologies such as TalkBack and Switch Access.

The scientific novelty of the thesis is:

- for the first time, Android travel planning applications were tested for inclusion gaps;
- for the first time, accessibility gaps in Android travel planning applications were categorized;
- for the first time, Android travel planning app that meets all WCAG and Android accessibility guidelines was developed.

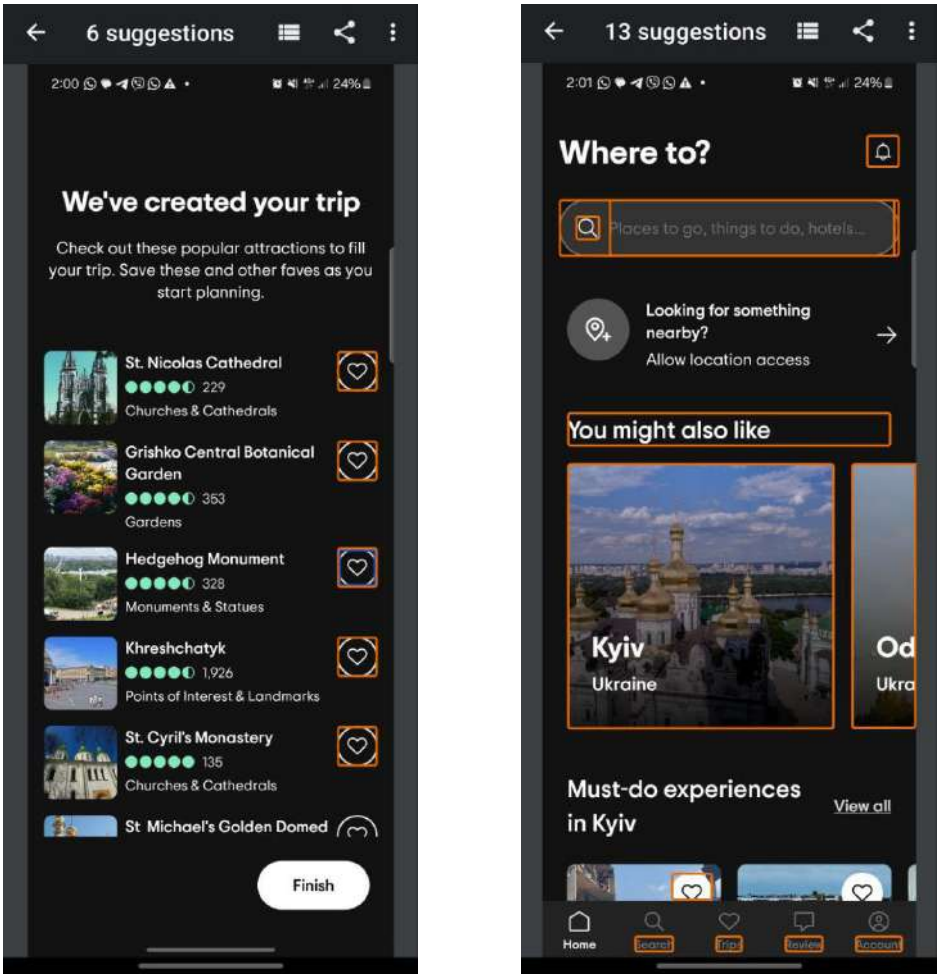
The thesis includes an introduction, 4 sections, general conclusions, and a reference list with 39 titles. The thesis spans 43 pages, 35 of which are of main text, 7 tables, and 17 figures.

1. OVERVIEW OF ACCESSIBILITY IN AVAILABLE SOLUTIONS

This overview consolidates accessibility assessments of seven popular Android apps (Tripadvisor, TripCase, Wanderlog, TripIt, Stippl, Funliday, and Lambus) focusing on inclusivity factors such as screen reader support, visual contrast, touch target size, and adherence to WCAG and Android accessibility guidelines.

1.1. Tripadvisor

Tripadvisor is a comprehensive travel platform and among the most popular ones in the world, offering user-generated reviews, hotel and flight bookings, vacation rentals, and restaurant reservations. It operates in over 40 countries and 20 languages, featuring approximately 1 billion reviews on around 8 million establishments (Tripadvisor, n.d.).



a *b*

Figure 1.1. Tripadvisor's accessibility assessment result (a - onboarding screen, b - home screen)

Issue	Description	Guidelines affected
Duplicate content descriptions	Every heart icon on the onboarding screen has the same content description “Toggle save”, making it hard for screen readers to differentiate buttons	WCAG 2.1 §4.1.2 (Name, Role, Value)
Small touch targets	Heart and notifications icons are smaller than the recommended size, making them harder to interact with	Android accessibility guidelines (Use large, simple controls)
Insufficient colour contrast	On the main page, search bar hint text and navigation bar icons have low colour contrast	WCAG 2.1 §1.4.3 (Contrast Minimum), Android accessibility guidelines (Increase text visibility)

Table 1.1. Tripadvisor's inclusivity concerns

Despite its popularity, the application is partially compliant. Basic accessibility features exist; however, Tripadvisor exhibits common accessibility issues like non-unique content descriptions and insufficient colour contrast. Lack of specificity and small touch targets significantly reduce usability for users with disabilities or relying on assistive technologies.

1.2. TripCase

TripCase is a trip-related information and messaging service that allows users to integrate all their trip-related information into one location, share trip details, and receive updates (TripCase, n.d.).

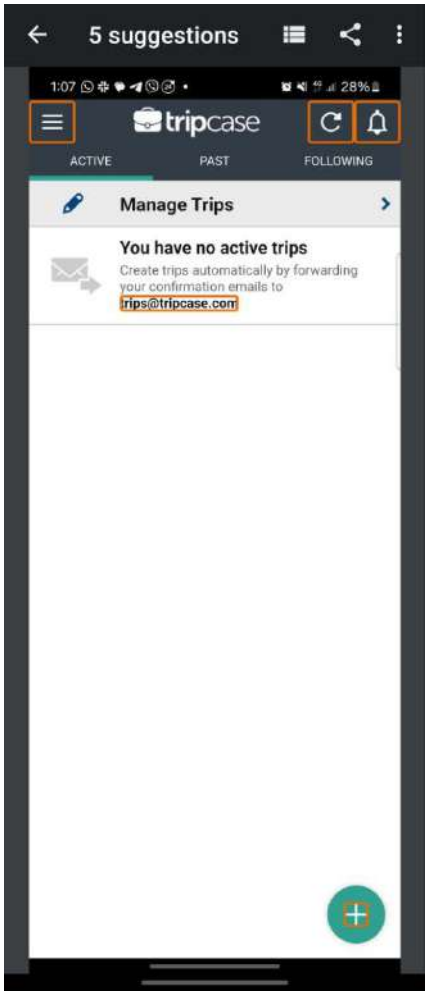


Figure 1.2. TripCase's accessibility assessment result

What is worth noting is that TripCase is using WebView for critical flows like sign up, log in, and password recovery. While not a WCAG violation, it can interfere with TalkBack navigation, particularly with swipe gestures and focus order, creating barriers for screen reader users during essential interactions.

Issue	Description	Guidelines affected
Small touch targets	Menu, notifications, and update buttons are smaller than recommended touch target size. Contact email is interactive but only 16dp in height, making	Android accessibility guidelines (Use large, simple controls)

	it especially difficult for users with motor impairments to tap accurately.	

Table 1.2. TripCase's inclusivity concerns

The application is partially compliant. TripCase's use of small touch targets is a commonly overlooked aspect of applications developed for Android; however, it provides content descriptions for all elements on the screen and enables users to perform all actions with screen reader.

1.3. Wanderlog

Wanderlog is a free travel planning app that allows users to create itineraries, collaborate with friends, organize reservations, and navigate trips. It supports features like route optimization, offline access, and expense tracking (Wanderlog, n.d.).

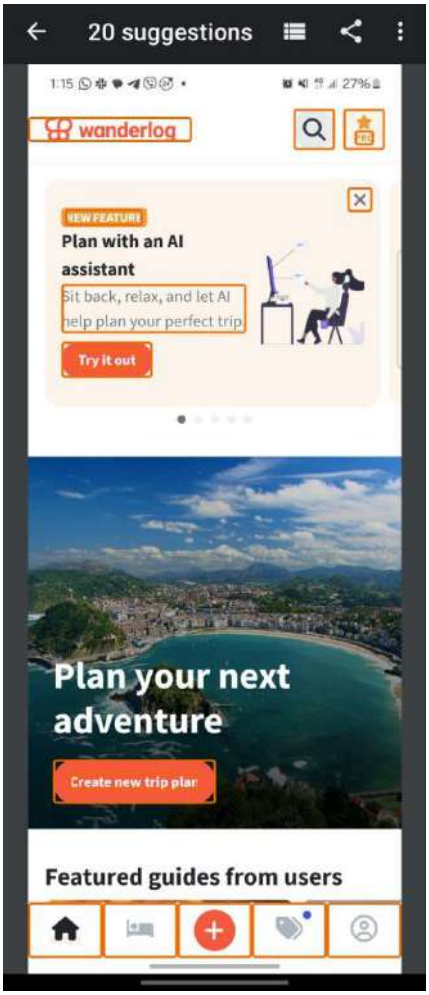


Figure 1.3. Wanderlog's accessibility assessment result

Issue	Description	Guidelines affected
Missing labels and descriptions	Search and bottom navigation bar icons lack labels or readable content descriptions. As a result, screen reader users will not be able to understand the function of these icons.	WCAG 2.1 §4.1.2 (Name, Role, Value)
Insufficient colour contrast	On-screen text and icons have insufficient contrast.	WCAG 2.1 §1.4.3 (Contrast Minimum), Android accessibility guidelines (Increase

		text visibility)
Small touch targets	Some buttons are smaller than the recommended size for touch targets.	Android accessibility guidelines (Use large, simple controls)

Table 1.3. Wanderlog's inclusivity concerns

This application is non-compliant. Critical accessibility features are missing, including descriptive labels and visual clarity. The app poses barriers to both screen reader users and users with motor or visual impairments.

1.4. TripIt

TripIt is a travel organizing app that creates a master itinerary for each trip by compiling travel details from confirmation emails. Users can access their itineraries anytime, even offline (TripIt, n.d.).

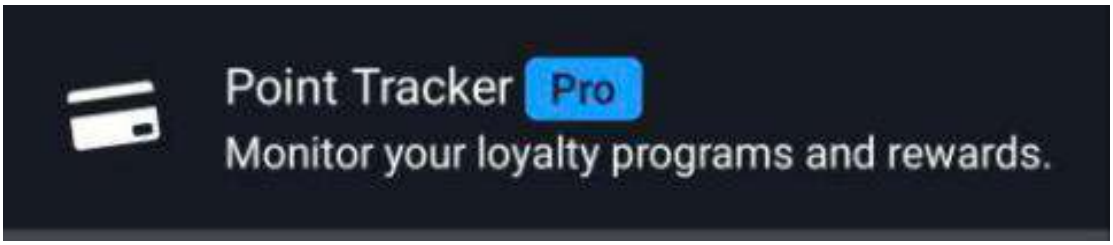


Figure 1.4. Pro feature setting in TripIt

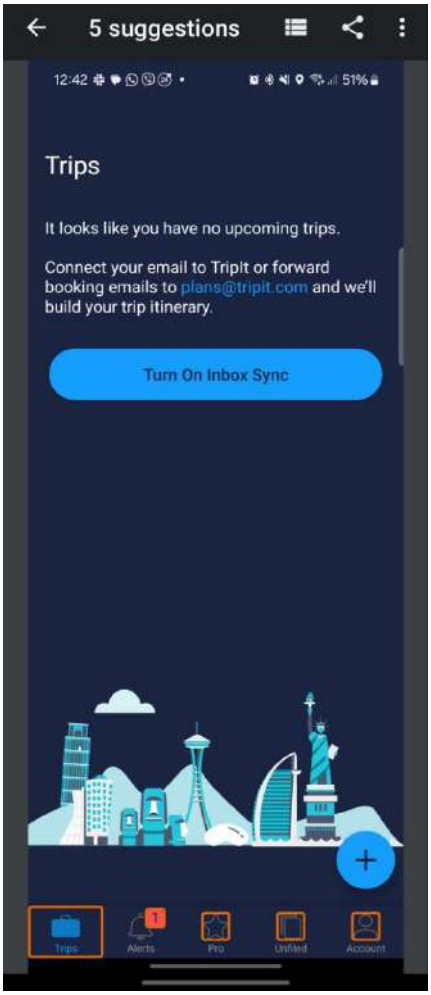


Figure 1.5. TripIt's accessibility assessment result

Issue	Description	Guidelines affected
Insufficient colour contrast	Bottom bar icons lack adequate colour contrast.	WCAG 2.1 §1.4.3 (Contrast Minimum), Android accessibility guidelines (Increase text visibility)
Poor labelling of feature status	In settings of premium features the “Pro” label is read immediately after the feature name, causing confusion.	WCAG 2.1 §3.3.2 (Labels or Instructions)
Text size fixed in dp instead of sp	Badge text size is fixed in dp units rather than	WCAG 2.1 §1.4.4 (Resize Text)

	scalable sp, impairing font scaling.	

Table 1.4. TripIt's inclusivity concerns

This application is mostly compliant. It uses a lot of native solutions for elements like dialogs, buttons, lists, and provides content descriptions for each and every element, making it easy to navigate through the application with the assistance of TalkBack. However, TripIt faces challenges with colour contrast, label clarity, and text scaling, all of which can confuse and reduce accessibility for users.

1.5. Stippl

Stippl is an all-in-one travel planner designed to streamline every aspect of a journey—from planning and budgeting to packing and tracking. It offers features like itinerary creation, expense management, and travel documentation (Stippl, n.d.).

Figure 1.6. Stippl's accessibility assessment result

Issue	Description	Guidelines affected
Insufficient colour contrast	Account page elements including the scratch button, navigation bar text, and user nickname have poor contrast.	WCAG 2.1 §1.4.3 (Contrast Minimum), Android accessibility guidelines (Increase text visibility)
Small touch targets	Followers, followings, and countries sections have small touch targets, settings and share icons are smaller than recommended.	Android accessibility guidelines (Use large, simple controls)
Custom SVG icon rendering	Use of custom SVG rendering instead of built-in recommended methods can cause accessibility issues.	WCAG 2.1 §4.1.2 (Name, Role, Value)

Table 1.5. Stippl's inclusivity concerns

This application is partially compliant. Stippl’s accessibility is compromised by low contrast, small touch targets, and custom icon implementations that may break compatibility with assistive tools.

1.6. Funliday

Funliday is a trip planning app that helps users manage trips, find local attractions, and get directions. It supports itinerary organization, schedule planning, and route management across various platforms (Funliday, n.d.).

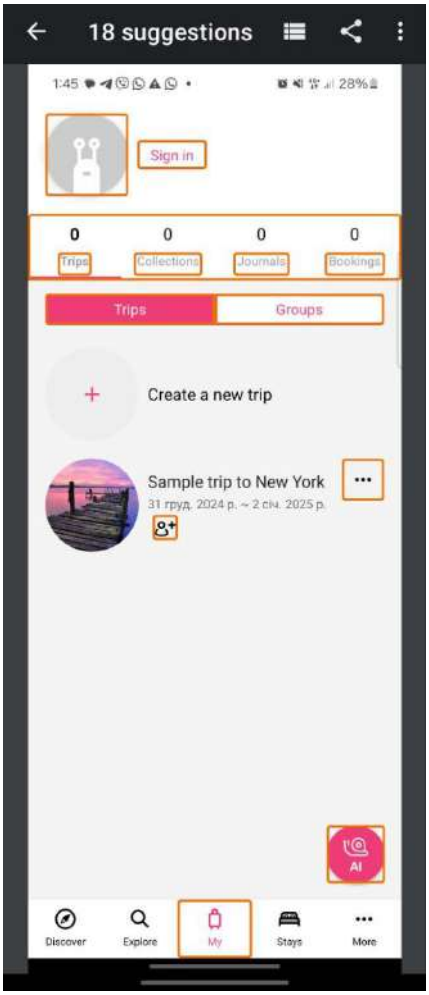


Figure 1.7. Funliday's accessibility assessment result

Issue	Description	Guidelines affected
Missing content descriptions	Multiple icons, including profile pictures, menu buttons, and floating action button lack content descriptions.	WCAG 2.1 §4.1.2 (Name, Role, Value)
Insufficient colour contrast	Text elements have low contrast with their backgrounds.	WCAG 2.1 §1.4.3 (Contrast Minimum), Android accessibility guidelines (Increase text visibility)

Table 1.6. Funliday's inclusivity concerns

This application is non-compliant. Funliday suffers from missing labels and poor contrast, resulting in key interactive elements are not accessible via screen readers and reduced readability for many users. It creates barriers for screen reader users and those with visual impairments.

1.7. Lambus

Lambus is an all-in-one travel platform that allows users to plan and organize trips, including stops, expenses, tickets, photos, and more. It supports both individual and group travel planning (Lambus, n.d.).

Figure 1.8. Lambus's accessibility assessment result

Issue	Description	Guidelines affected
Missing labels and descriptions	No icons have labels or content descriptions, severely limiting	WCAG 2.1 §4.1.2 (Name, Role, Value)

	screen reader usability.	
Small touch targets	Most icons and interactive elements are smaller than the recommended size.	Android accessibility guidelines (Use large, simple controls)
Focus on off-screen elements	Elements not visible on screen are still accessible by TalkBack, causing confusion.	WCAG 2.1 §2.4.3 (Focus Order)
Text size fixed in dp instead of sp	Some text sizes are fixed in dp units rather than scalable sp, impairing font scaling.	WCAG 2.1 §1.4.4 (Resize Text)
Insufficient colour contrast	Some icons have low contrast against their backgrounds.	WCAG 2.1 §1.4.3 (Contrast Minimum), Android accessibility guidelines (Increase text visibility)

Table 1.7. Lambus's inclusivity concerns

This application is non-compliant. Lambus has severe accessibility issues across all key areas, making it exceedingly difficult for users with disabilities to use effectively as it fails basic accessibility guidelines.

2. ACCESSIBILITY BARRIERS IN TRAVEL PLANNING APPS

There is multiple common accessibility strengths observed during assessment. For example, most apps use a standard UI layout, which helps in making navigation more intuitive, or the presence of interactive elements in logical positions matching user expectations. However, some recurring issues were observed that indicate systematic gaps in how inclusivity is approached while designing travel planning applications, leading to barriers for people with visual or motor impairments.

2.1. Visual perception barriers

Visual perception barriers arise when design elements fail to meet visual accessibility standards. These include insufficient colour contrast between text and background, small touch targets, and missing visual cues. Such issues can prevent users with low vision or colour blindness from navigating apps effectively.

For example, Tripadvisor displays insufficient colour contrast in the search bar hint and navigation bar icons. TripIt's bottom navigation icons also lack sufficient contrast, and Stippl shows contrast issues with the "scratch" button, navigation bar text, and user nicknames. Similarly, Funliday and Wanderlog contain various text and icon elements that do not meet contrast guidelines.

Due to the cumulative impact of these visual issues, apps like Lambus and Stippl were deemed unusable for users relying on visual clarity, including those with low vision, colour blindness, and older adults who often experience diminished visual acuity.

2.2. Motor and dexterity barriers

These barriers relate to the physical act of interacting with an app. When interactive elements such as buttons and icons are too small, not spaced adequately, or difficult to target, users with limited dexterity or motor impairments face significant challenges.

Apps such as TripCase and Tripadvisor include icons and buttons that fall below the minimum recommended touch target size (48x48dp), making them difficult to tap accurately. In Stippl, the icons for settings and sharing, along with follower stats sections, are smaller than required. Lambus also has widespread issues with icon sizing.

TripCase and Lambus were considered problematic due to their failure to meet touch target requirements, impacting usability for many. This barrier primarily

affects users with motor impairments, arthritis, tremors, or those who rely on assistive technologies like styluses or mouth sticks.

2.3. Auditory and screen reader barriers

This barrier type involves missing or improper support for screen readers such as TalkBack. When app elements lack content descriptions or have misleading labels, screen reader users are left without essential information to navigate or interact with the UI meaningfully.

In Tripadvisor, all heart icons in a list have the same content description ("Toggle save"), making it difficult to distinguish between them. Wanderlog and Funliday contain multiple unlabelled icons, and Lambus has almost no accessible labels. Additionally, TripCase uses WebViews for critical actions like login, which disrupt TalkBack navigation.

Due to a high number of such issues, Lambus was found to be unusable by screen reader users. These barriers critically impact users who are blind or visually impaired, and those who rely on text-to-speech output for interaction.

2.4. Recommendations for impoving Android applications' inclusivity

1. Consistently provide unique, meaningful content descriptions for all interactive elements.
2. Ensure all text and interactive elements meet minimum contrast ratios (4.5:1 for text, 3:1 for icons and large text).
3. Design touch targets with a minimum size of 48dp × 48dp, with padding where necessary.
4. Exclude non-visible UI elements from accessibility focus to avoid confusion.
5. Use scalable text units (sp) to respect user font size preferences.
6. Prefer native Android vector drawable formats over custom SVG rendering for icons.
7. Avoid WebViews for critical user flows or ensure they are fully accessible via screen readers and keyboard navigation.
8. Perform thorough testing with TalkBack, Accessibility Scanner, and contrast checking tools during development cycles.

3. REVIEW OF USED TECHNOLOGIES

3.1. Android

Android OS is a mobile operating system rooted in Linux that operates on smartphones and tablet devices (Mixon, 2025). The Android environment comprises an OS built on the Linux kernel, a graphical user interface, a web browser, and user-facing applications that can be installed.

Android was made available under the Apache version 2 open-source license, enabling the development of multiple variations for other electronics like gaming systems and digital cameras (Mixon, 2025). While Android is grounded in open-source software, the majority of Android-powered devices come with a set of pre-installed Google apps, including Google Maps, YouTube, Chrome, and Gmail.

Android offers a variety of Accessibility Services, including but not limited to TalkBack (screen reader), Switch Access (physical switch interface), Voice Access (voice command interface), BrailleBack (keyboard for users with low vision) (Google, n.d.-b). All of them assist users in navigating apps through alternative input and output methods and contribute to their positive experience of using their gadgets.

At the same time, developing the mobile trip planner as an Android application ensures reaching the largest possible audience, since Android powers 72 % of all smartphones and other mobile devices worldwide (Statcounter)

3.2. Kotlin

Kotlin is a modern but already mature programming language that was released in 2016 by JetBrains (W3Schools). It is heavily used for development of mobile applications (especially Android apps) by as many as over 60% of professional Android developers (Google, n.d.-e).

One of Kotlin's most compelling advantages is its emphasis on safer code. Thanks to its built-in nullability handling, Kotlin drastically reduces the risk of null pointer exceptions — one of the most common causes of crashes in software (so much in fact that Tony Hoare, who was the creator of null value, later called it a "billion-dollar mistake" (Hoare, 2009)). Applications built with Kotlin are statistically 20% less likely to crash, which directly translates into better user experience and fewer maintenance headaches (Google, n.d.-e).

Kotlin also excels in managing asynchronous tasks through structured concurrency with coroutines. Whether it is handling network requests or updating

databases, coroutines make these operations straightforward and efficient, without the complexity and pitfalls of traditional threading (W3Schools).

3.3. Jetpack Compose

As Android's recommended modern UI toolkit, Jetpack Compose simplifies and accelerates UI development by enabling expressing complex layouts and interactions in a declarative Kotlin-based syntax, which leads to less code, reducing the chance for bugs, and making the UI logic easier to understand, modify, and maintain over time (Google, n.d.-d).

What makes Compose especially powerful is its intuitive nature. Instead of manually updating the UI as the app state changes, it is simply described what the UI should look like and Compose handles the rest. This declarative approach eliminates entire classes of bugs related to state synchronization and leads to more predictable, testable UIs (Google, n.d.-d).

Beyond visual design, Compose empowers developers to create accessible, inclusive applications with built-in support for accessibility best practices, from basic ones (such as providing content descriptions, ensuring colour contrast, and defining proper touch target sizes) to some advanced accessibility features (Google, n.d.-i). For instance, custom accessibility actions, dynamic content descriptions that adapt to state, and clear communication patterns for assistive technologies.

3.4. Material Design 3

Material Design 3 is Google's open-source design system, created to help developers and designers craft beautiful, consistent, and usable digital experiences (Google, n.d.-g).

Accessibility by default is a core design value for Material. Material's accessibility requirements and goals are principles embedded across its components and design guidelines (Google, n.d.-a). This means that out-of-the-box, Material components are structured to support users with a wide range of needs, including those with low vision, blindness, hearing or motor impairments, cognitive challenges, or even temporary limitations like an injured hand. Material's components come with built-in accessibility standards, which lay a solid foundation for inclusive design (Google, n.d.-a).

When we use Jetpack Compose, we benefit from a native implementation of Material Design components, fully optimized for modern declarative UI development. These Compose-based components are not only visually consistent with Material standards but also integrated with Android's accessibility APIs,

ensuring they are correctly interpreted by screen readers and other assistive tools (Google, n.d.-h).

Compose is also supported by Material's tools such as the Material Theme Builder (Evjen, 2021), which helps developers maintain proper colour contrast and visual balance, essential for users with visual impairments.

Using Material Design 3 in combination with Jetpack Compose strengthens ability to build modern Android apps that are inclusive by design.

3.5. Kotlin Coroutines

Kotlinx.Coroutines (stylized in lowercase) is a comprehensive library developed by JetBrains to support asynchronous and concurrent programming using coroutines in Kotlin (Murashkina, 2022).

One of the main advantages of coroutines is that they are lightweight and efficient, allowing developers to launch many coroutines on a single thread without significant overhead. This efficiency is made possible by suspend functions, which can pause their execution without blocking the underlying thread. Instead of holding up system resources, a suspended function simply saves its state and resumes when needed — making it a memory-friendly and scalable approach to handling asynchronous tasks (To, 2022).

The library offers a clean, expressive API, making asynchronous programming more intuitive and less error prone. It simplifies complex scenarios such as network calls, database operations, or background processing, which would otherwise require callbacks or thread management (Google, n.d.-e).

3.6. ViewModel

The ViewModel class plays a significant role in managing UI-related data and business logic in a lifecycle-aware way, making it an essential component in modern Android architecture, especially when using Jetpack Compose (Google, n.d.-j).

At its core, a ViewModel is a state holder for the screen. It exposes state to the UI layer and encapsulates logic tied to that screen, keeping the composables clean and focused solely on rendering. One of its key advantages is that it caches and preserves UI state across configuration changes, such as screen rotations or app backgrounding (Google, n.d.-j). This means there is no need to refetch or recalculate data when navigating between screens or handling lifecycle events, saving both processing time and network or database resources.

In Jetpack Compose, the ViewModel is also the primary mechanism for exposing state to composables. Thanks to Compose's reactive nature, updates to the state held in the ViewModel automatically trigger recomposition of the UI, making the user interface more predictable and responsive (Google, n.d.-j).

While it is technically possible to use a simple class to manage UI data, doing so introduces limitations. For example, navigating between activities or Compose Navigation destinations can result in losing that state, unless saving and restoring is manually implemented using saved instance state. The ViewModel, however, provides a built-in, lifecycle-conscious API to persist data seamlessly, removing that burden from the developer (Google, n.d.-j).

In addition to managing state, ViewModel gives any app a clear separation of concerns by isolating business logic from UI logic. This aligns with the Model-View-ViewModel (commonly abbreviated to MVVM) architectural pattern (Saxena, 2023), where:

- The Model handles data and business rules;
- The ViewModel prepares and exposes that data for the UI;
- The View (in this case, composables) displays it (Google, n.d.-j).

By using ViewModel, it is not only state management or performance that is improving, but also an app architecture by being clean and scalable, thus supporting maintainability and testability across Android project.

3.7. Room

Room is a robust persistence library that is part of Android Jetpack (Google, n.d.-k), a collection of libraries designed to promote best practices, reduce boilerplate, and ensure consistent behaviour across Android versions and devices (Gaur, 2025).

Room acts as a high-level abstraction layer over SQLite, enabling to interact with the database using more fluent, concise, and safer code while still leveraging the full capabilities of SQLite under the hood (Google, n.d.-l).

What makes Room especially beneficial are features such as:

- compile-time verification of SQL queries, which helps catch errors early in development rather than at runtime;
- convenience annotations which eliminate much of the repetitive and error prone code typically associated with database interactions;

- simplified and reliable migration paths, which are crucial for maintaining and evolving the database schema as the app grows (Google, n.d.-l).

Given these advantages, Google strongly recommends using Room instead of interacting with the SQLite APIs directly (Google, n.d.-l). Room provides a modern, efficient, and maintainable approach to local data storage, making it a preferred choice for Android app developers aiming for stability, performance, and clean architecture.

3.8. Kotlin Serialization

When selecting libraries for the Android project, it is recommended to prefer those that rely on code generation (codegen) over reflection. This is because codegen allows tools like R8 and ProGuard to better analyse, optimize, and strip unused code, resulting in smaller, faster, and more efficient applications (Google, n.d.-m). With reflection, it is harder for these tools to determine which parts of the code are actually used at runtime, which can lead to larger APK sizes and missed optimization opportunities.

A common example is serialization. Libraries like Gson, which use reflection to serialize and deserialize objects, often introduce runtime overhead and make it harder to perform code shrinking. In contrast, libraries like Kotlin Serialization generate the necessary code at compile time, which improves performance, ensures better compatibility with optimization tools, and avoids reflection-related issues (Google, n.d.-m).

3.9. Jetpack Navigation

The Android Jetpack Navigation component is a comprehensive framework designed to simplify and standardize in-app navigation. It consists of the Navigation library, the Safe Args Gradle plugin, and supporting tooling integrated into Android Studio. Together, these tools help developers implement various navigation scenarios from basic user interactions like button clicks to more advanced patterns such as bottom navigation, app bars, and navigation drawers (Google, n.d.-n). One of the core strengths of the Navigation component is that it enforces a consistent and predictable navigation experience across the app by following a well-defined set of principles. This not only improves the usability of the application but also helps reduce bugs and inconsistencies in navigation flow.

Importantly, because Google develops both Jetpack Navigation and Jetpack Compose, the Navigation component offers full support for Compose-based applications. This means developers can navigate seamlessly between composables

while still leveraging the robust features of the Navigation component such as type-safe argument passing via Safe Args, back stack management, and lifecycle-aware navigation actions (Google, n.d.-o).

Using the Navigation component in Jetpack Compose projects leads to a unified navigation solution that is officially supported, forward compatible, and tightly integrated with the Android development ecosystem.

3.10. Coil

Coil, which stands for Coroutine Image Loader, is an image loading library designed for Android and Compose Multiplatform (Coil, n.d.). It is known for its speed, achieved through various optimizations such as memory and disk caching, image downsampling, and the ability to automatically pause or cancel requests when appropriate. Despite its powerful features, Coil remains lightweight, relying only on Kotlin, Coroutines, and Okio, and integrates smoothly with Google's R8 code shrinker to keep app sizes small (Coil, n.d.). The library is also easy to use, as its API takes full advantage of Kotlin's language features to provide a simple interface with minimal boilerplate code. Being Kotlin-first, Coil is a modern solution that interoperates well with other contemporary libraries like Compose, Coroutines, Okio, OkHttp, and Ktor (Coil, n.d.). This sets it apart from other image loading libraries such as Glide, which, unlike Coil, is still in beta for Compose support (Google, n.d.-f).

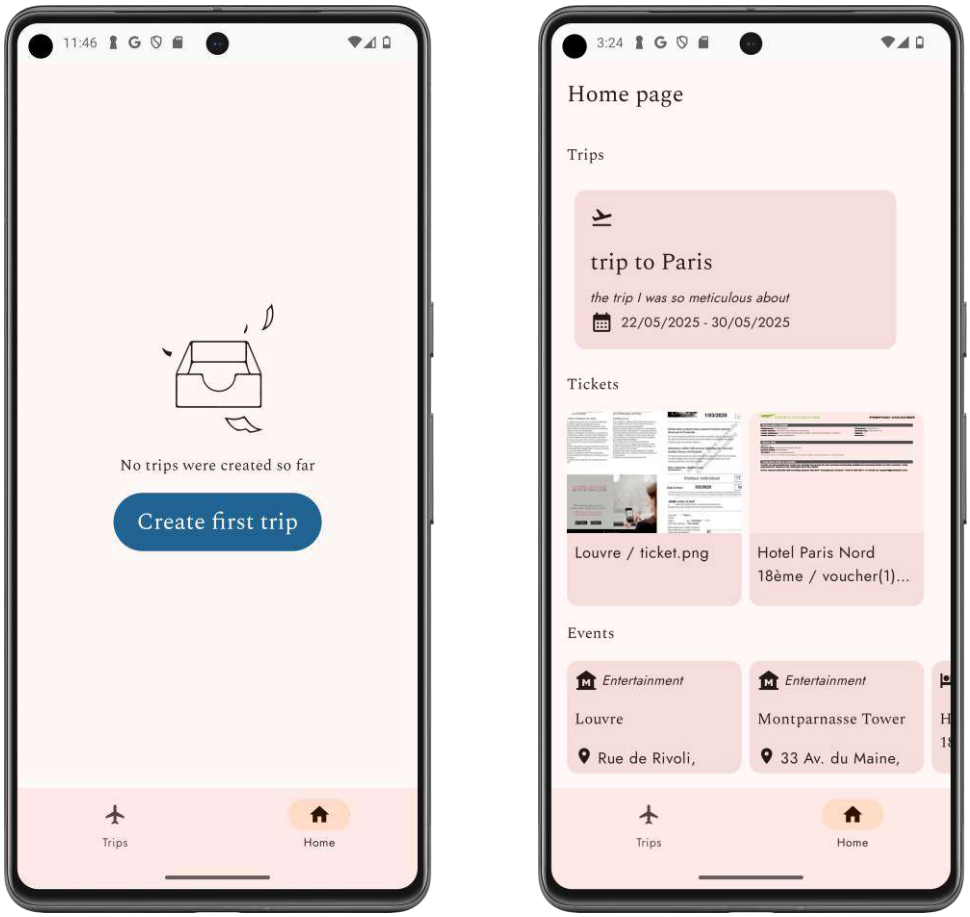
4. OVERVIEW OF IMPLEMENTATION'S FUNCTIONALITY

4.1. Home screen

The home screen of the application dynamically adapts based on whether a trip has been created.

Empty State — when no trips are planned, the screen displays a clear, central message "No trips were created so far" along with a prominent "Create first trip" button. This minimalist design reduces cognitive load and directs the user to the primary action.

Populated State — when trips are created, the home screen transforms into an organized dashboard. It features distinct sections for "Trips," "Tickets," and "Events." Each section lists relevant items. The layout is clean, with clear visual separation between categories and individual items.



a *b*

Figure 4.1. Home screen (a - empty state, b - populated state)

The design of the home screen demonstrates several inclusive considerations:

The clear Call to Action in the Empty State (the "Create first trip" button) is large, centrally located, and uses high-contrast text, making it easy to identify and activate for users with low vision or motor impairments. The clear, concise text avoids ambiguity, which only contributes to a better user experience to those with cognitive disability.

In the populated state distinct sections for "Trips," "Tickets," and "Events" provide a clear visual and logical hierarchy. This helps users with cognitive disabilities or those who may be easily overwhelmed by too much information to process content in manageable chunks. The consistent layout also aids predictability, as both states prioritize essential information, minimizing visual clutter.

Additionally, the screen is also semantically structured for screen readers. An example of this is the fact that "Trips," "Tickets," and "Events" labels are marked as headings using a semantics modifier. It is highly beneficial for users who rely on screen readers. By designating these as headers, screen readers can not only announce them as headings, providing the important context to the end user, but also enhance their navigation speed. Users can navigate directly between headings using specific gestures on their assistive technology. This allows them to quickly jump to the "Tickets" section if they are looking for their booking, as users no longer are required to swipe through all of their content in one section to reach another. This significantly improves efficiency and reduces frustration for users with visual impairments or cognitive disabilities.

4.2. Create trip screen

The "Create Trip" screen is a form-based interface designed for users to input details about their upcoming trip. It includes input fields (e.g. "Title," "Start date," "End date," "Description"), and "Save" button.

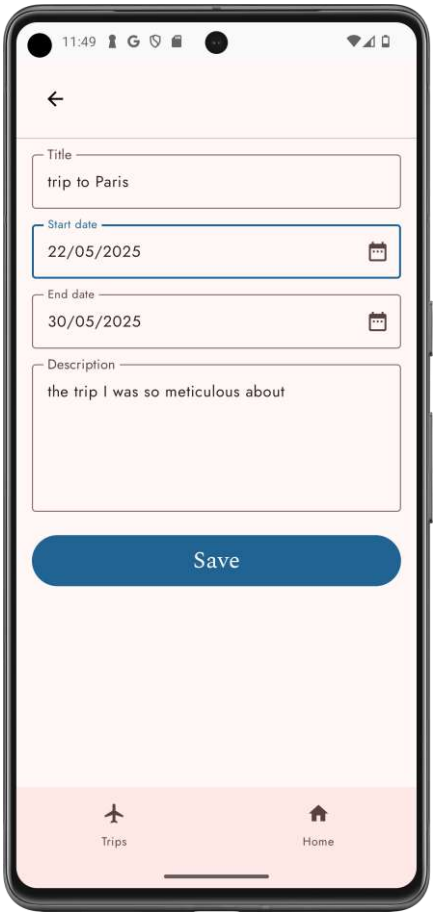


Figure 4.2. Create trip screen

The reliance on Jetpack Compose and Material 3 for this screen inherently provides a durable foundation for inclusivity. Utilizing Material Design components ensures correct application of semantic information to UI elements. As a result, all labels are associated with corresponding input fields, providing context for users with visual impairments. Material 3's colour palettes and typography are designed for good contrast and legibility, benefiting users with low vision or colour blindness, and complex elements like date pickers are designed to be accessible, allowing navigation via screen readers and keyboard inputs, which is crucial for users who cannot easily tap small calendar dates.

4.3. All trips screen

The "All Trips" screen displays a list of all trips created by the user. Each trip is presented as a list item, showing the trip title, a brief description and the start and end dates. On the right side of each trip item, there is a vertical ellipsis icon, indicating an action menu. A plus icon in the top right corner allows the user to add a new trip.

Figure 4.3. All trips screen

The implementation of accessibility actions for the action menu demonstrates a sophisticated understanding of screen reader user needs. The decision to exclude the action menu button (the vertical ellipsis) from the standard navigation flow for screen readers is a critical accessibility enhancement. For screen reader users, having every interactive element announced on list items can slow down navigation and be exhaustive, especially when there is many of them. This can be improved by making "Delete" and "Edit" actions reachable through accessibility actions instead. Additionally, the screen utilizes semantics modifier's property of collection information. When navigating through the screen with a screen reader, user can evaluate how many items there are in the list, and while iterating through the list the index of each element is announced.

4.4. Trip details screen

The "Trip Details" screen provides a comprehensive overview of a selected trip, allowing for further management and planning. At the top, it displays the trip's title, description, and dates. Below this, there are two prominent action buttons: "Edit trip" and "Delete trip," enabling direct modification or removal of the trip. Further down,

the screen features a tabbed navigation system with "Schedule" and "Packing list" options, allowing users to switch between different planning aspects of the trip.

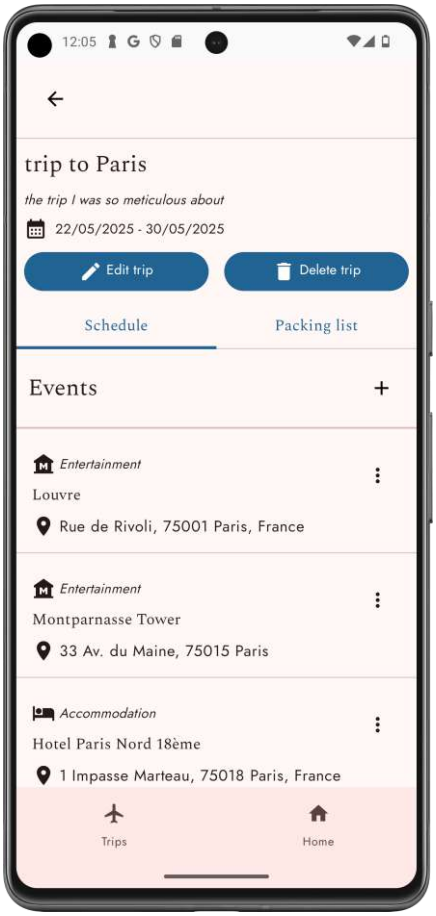


Figure 4.4. Trip details screen

When the "Events" tab is selected, a list of events associated with the trip is displayed. Each event item includes an action menu that follows the same logic that was described on all trips screen, where all functionality from its dropdown menu is duplicated via accessibility actions.

When the "Packing list" tab is selected, the interface adapts to display and manage packing items. This tab offers three distinct modes, controlled by a set of chips: "Check mode," "Edit mode," and "Reorder mode." An input field "Enter item or category" is available for adding new items to the checklist. Items can be grouped under categories, which helps users with cognitive disabilities organize information and navigate the list more easily.

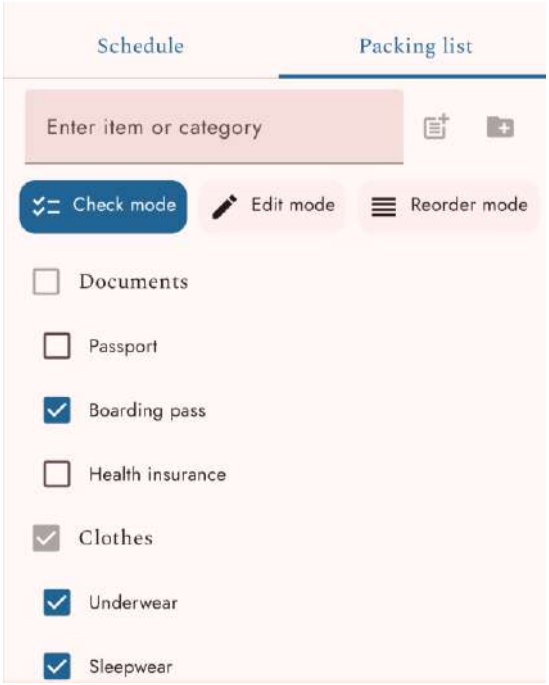


Figure 4.5. Packing list section in check mode

Check mode is the default mode, optimized for marking items as packed. Each item in the list is accompanied by a prominent checkbox.

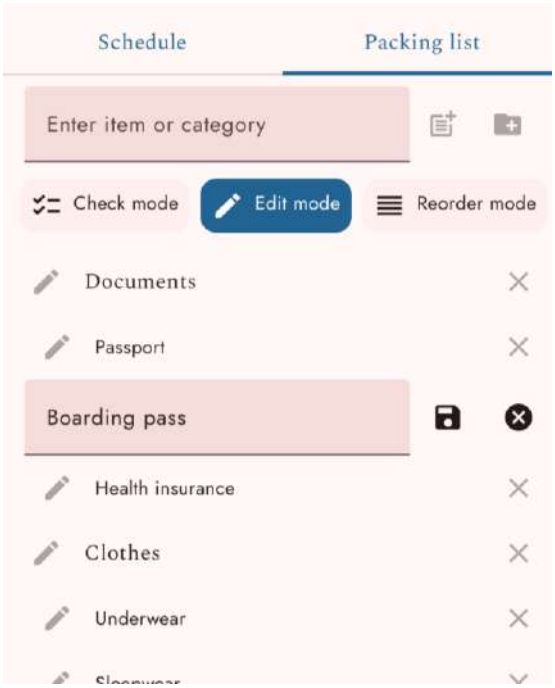


Figure 4.6. Packing list section in edit mode

In Edit mode, the focus shifts to modifying or removing items. Each item displays an edit icon (pencil) next to its name, allowing the user to change the item's text. A delete icon (cross) is also present next to each item for removal.

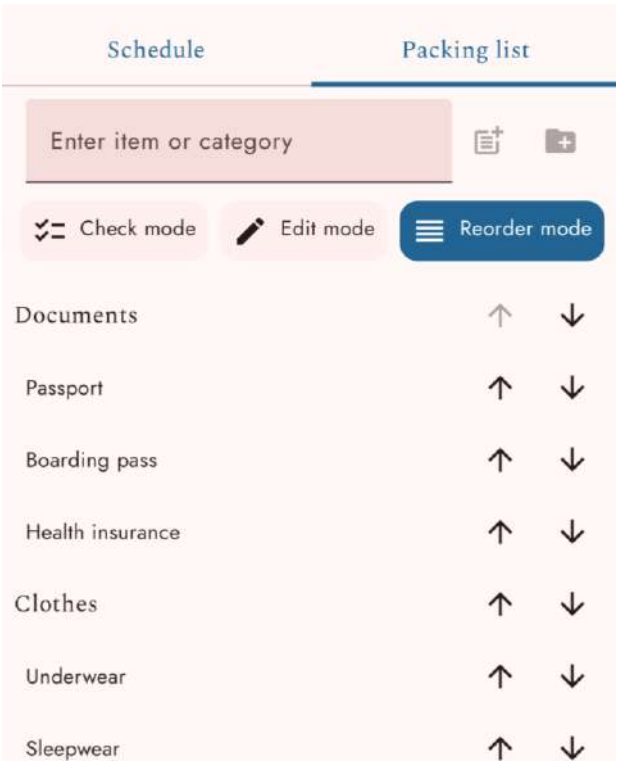


Figure 4.7. Packing list section in reorder mode

Reorder mode enables users to change the order of items within the packing list. Each item is accompanied by up and down arrow icons, allowing users to move items individually within their categories or across the entire list.

By separating functionalities into distinct "Check," "Edit," and "Reorder" modes, the application prevents accidental actions and reduces cognitive overload. Users only see the relevant controls for their current task, which is highly beneficial for individuals with cognitive disabilities, ADHD, or fine motor control issues who might otherwise struggle with a cluttered interface offering all options simultaneously. By separating functionalities into distinct "Check," "Edit," and "Reorder" modes, the application prevents accidental actions and reduces cognitive overload. Users only see the relevant controls for their current task, which is highly beneficial for individuals with cognitive disabilities, ADHD, or fine motor control issues who might otherwise struggle with a cluttered interface offering all options simultaneously. Similar to the "Event type" chips, mode-switching chips are also implemented with appropriate semantic roles, which ensures that screen readers

accurately announce the current mode and allow users to easily switch between modes using standard accessibility gestures.

4.5. Create event screen

The "Create Event" screen is similar in structure to the "Create Trip" screen, featuring input fields for "Title," "Description," and "Address,". The primary distinguishing feature is the "Event type" section, which presents a set of custom "chip" elements (e.g., "Accommodation," "Transport," "Entertainment", "No type") allowing the user to categorize the event. Depending on the chosen type of event, the following input fields will prompt the user to input:

Start Date/Time and End Date/Time for Entertainment category.

Departure Date/Time and Arrival Date/Time for Transport category.

Check-in/Check-out Date with From/To Times for Accommodation category.

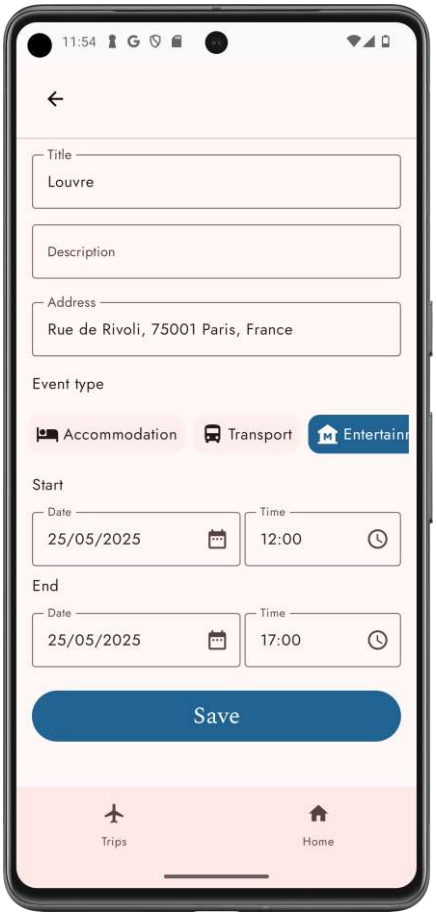


Figure 4.8. Create event screen

While inheriting the strong accessibility foundations from Jetpack Compose and Material 3 as seen in the "Create Trip" screen (e.g., clear labels, touch target

sizes, predictable input fields), the "Event type" chips introduce specific inclusivity benefits. The use of toggleable modifier along with specifying Role.RadioButton for the custom chips is a critical accessibility implementation. This explicitly informs assistive technologies that these chips function as a group of radio buttons. For users relying on screen readers, this means that screen readers will announce each chip as a "radio button" and indicate its selected/unselected state (for example, "Accommodation, radio button, not selected," or "Entertainment, radio button, selected") and understand that these chips form a mutually exclusive selection group like traditional radio buttons.

4.6. Event details screen

The "Event Details" screen presents a detailed view of a specific event within a trip. At the top, it displays the event's type along with its name, followed by the address. Below this primary information, there are four action buttons: "Edit event," "Delete event," "Add image," and "Add PDF." Further down, a "Tickets" section lists attached documents, with a vertical ellipsis icon next to it, allowing users to delete or change the display name of the said document.

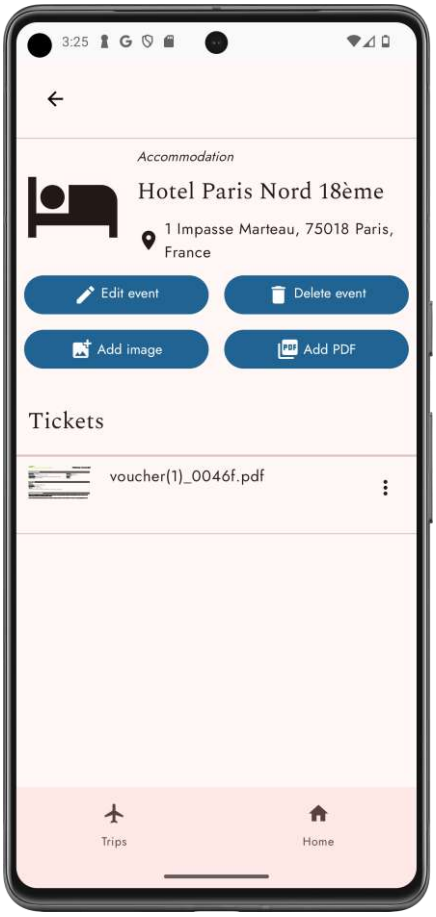


Figure 4.9. Event details screen

CONCLUSIONS

The goal was achieved by addressing the gaps in available mobile applications developed with a view to being of use to people with disabilities while planning their journey, that rarely consider the needs of people with disabilities.

The tasks were accomplished due to the following:

- Android travel planning applications were tested for inclusivity barriers;
- accessibility issues in Android travel planning applications were systemically identified;
- an Android travel planning application that satisfies WCAG and Android accessibility guidelines was developed.

The result of the research was a full-fledged mobile application that enables people with disabilities to conveniently organize their journey and store necessary information in one place. This research is a significant contribution to the development of inclusive software, demonstrating practical ways to implement digital accessibility principles in mobile solutions. The proposed approach contributes to the formation of a barrier-free digital environment and can be used as a guideline for developers who seek to create applications that are accessible to all users regardless of their physical capabilities.

REFERENCES

Chang, C., & Tuli, S. (Writers). (2021). *Introduction to Accessibility on Android* [Motion Picture]. YouTube. Retrieved from https://www.youtube.com/watch/rtyjbUxUmG8

Coil. (n.d.). *Overview.* Retrieved from Coil: https://coil-kt.github.io/coil/

Curry, D. (2025, January 28). *Travel App Report 2024.* Retrieved from Business of Apps: https://www.businessofapps.com/data/travel-app-report/

Evjen, Y. (2021, October 27). *Introducing Material Theme Builder.* Retrieved from Material Design Blog: https://m3.material.io/blog/material-theme-builder

Friman, M., & Olsson, L. E. (2023). Are we leaving some people behind? Travel autonomy, perceived accessibility, and well-being among people experiencing mental and physical difficulties. *Transportation Research Part F: Traffic Psychology and Behaviour, 98*, pp. 243-253. doi:https://doi.org/10.1016/j.trf.2023.08.009.

Funliday. (n.d.). Retrieved from Funliday: www.funliday.com

Gaur, A. (2025, January 28). *Room Database in Android.* Retrieved from Medium: https://medium.com/@anandgaur2207/room-database-in-android-d5f279d4648a

Google. (n.d.-a). *Accessibility overview – Material Design 3.* Retrieved from Material Design: https://m3.material.io/foundations/overview/principles

Google. (n.d.-b). *Android accessibility overview.* Retrieved from Google Support: https://support.google.com/accessibility/android/answer/6006564

Google. (n.d.-c). *Build accessible apps*. Retrieved from Android Developers: https://developer.android.com/guide/topics/ui/accessibility

Google. (n.d.-d). *Build better apps faster with Jetpack Compose.* Retrieved from Android Developers: https://developer.android.com/compose

Google. (n.d.-e). *Develop Android apps with Kotlin.* Retrieved from Android Developers: https://developer.android.com/kotlin

Google. (n.d.-f). *Loading images.* Retrieved from Android Developers: https://developer.android.com/develop/ui/compose/graphics/images/loading

Google. (n.d.-g). *Material Design 3.* Retrieved from Material Design: https://m3.material.io/

Google. (n.d.-h). *Material Design 3 in Compose.* Retrieved from Accessibility overview – Material Design 3: https://developer.android.com/develop/ui/compose/designsystems/material3

Google. (n.d.-i). *Semantics.* Retrieved from Android Developers: https://developer.android.com/develop/ui/compose/accessibility/semantics

Google. (n.d.-j). *ViewModel overview.* Retrieved from Android Developers: https://developer.android.com/topic/libraries/architecture/viewmodel

Google. (n.d.-k). *Android Jetpack*. Retrieved from Android Developers: https://developer.android.com/jetpack

Google. (n.d.-l). *Save data in a local database using Room.* Retrieved from Android Developers: https://developer.android.com/training/data-storage/room

Google. (n.d.-m). *Choose libraries wisely.* Retrieved from Android Developers: https://developer.android.com/build/choose-libraries-wisely

Google. (n.d.-n). *Navigation.* Retrieved from Android Developers: https://developer.android.com/guide/navigation

Google. (n.d.-o). *Navigation with Compose.* Retrieved from Android Developers: https://developer.android.com/develop/ui/compose/navigation

Hoare, T. (2009, August 25). *Null References: The Billion Dollar Mistake.* Retrieved from InfoQ: https://www.infoq.com/presentations/Null-References-The-Billion-Dollar-Mistake-Tony-Hoare/

Lambus. (n.d.). Retrieved from Lambus: lambus.com

Milbergs, K. (2024, November 25). *Most Common Disabilities in the World (2024 Updated Statistics)*. Retrieved from Accessibly: https://accessiblyapp.com/blog/most-common-disabilities-world/

Mixon, E. (2025, October 15). *Definition: Android OS.* Retrieved from TechTarget: https://www.techtarget.com/searchmobilecomputing/definition/Android-OS

Motta, M. (2023, August 17). *Travel apps' 2023 stats and growth in five charts.* Retrieved from Adjust: https://www.adjust.com/blog/travel-apps-2023

Murashkina, N. (2022, February 16). *Coroutines guide.* Retrieved from Kotlin Docs: https://kotlinlang.org/docs/coroutines-guide.html

Saxena, S. (2023, November 1). *Introduction to Model View View Model (MVVM).* Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/introduction-to-model-view-view-model-mvvm/

Statcounter. (n.d.). *Mobile Operating System Market Share Worldwide - February 2025.* Retrieved February 28, 2025, from Statcounter Global Stats: https://gs.statcounter.com/os-market-share/mobile/worldwide

Stippl. (n.d.). Retrieved from Stippl: stippl.io

To, Z. (2022, May 12). *RxKotlin vs Coroutines.* Retrieved from Medium: https://ze9786.medium.com/rxkotlin-vs-coroutines-47fc5358e1fc

Tripadvisor. (n.d.). Retrieved from Tripadvisor: https://www.tripadvisor.com/

TripCase. (n.d.). Retrieved from TripCase: tripcase.com

TripIt. (n.d.). Retrieved from TripIt: tripit.com

W3C Recommendation. (2024, December 12). *Web Content Accessibility Guidelines (WCAG) 2.2.* Retrieved from World Wide Web Consortium (W3C): https://www.w3.org/TR/2024/REC-WCAG22-20241212/

W3Schools. (n.d.). *Kotlin Introduction.* Retrieved from W3Schools: https://www.w3schools.com/kotlin/kotlin_intro.php

Wanderlog. (n.d.). Retrieved from Wanderlog: wanderlog.com

World Health Organization. (2023, March 7). *Disability.* Retrieved from World Health Organization: https://www.who.int/news-room/fact-sheets/detail/disability-and-health