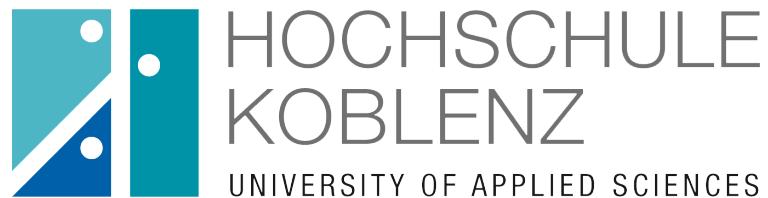


Hochschule Koblenz  
Fachbereich Ingenieurwesen  
Elektro- und Informationstechnik



# Studienarbeit

**Entwicklung einer Urlaubtracking App für Android**

Ayoub Saidi

543574

**Betreuer:** Prof. Dr. Markus Kampmann

Abgabedatum: 16.04.2024

## Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet habe. Diese Arbeit wurde in gleicher oder ähnlicher Fassung noch keiner anderen Prüfungsbehörde vorgelegt. Ich versichere, dass die elektronische Version der Arbeit mit der gedruckten Version der Arbeit inhaltlich übereinstimmt. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Mir ist bewusst, dass ein Nichteinhalten der Regeln guter wissenschaftlicher Praxis das Nichtbestehen der Prüfungsleistung zur Folge hat.



---

Koblenz, den 16.04.2024

---

Ayoub Saidi

# Inhaltsverzeichnis

<b>Selbstständigkeitserklärung</b>	<b>I</b>
<b>Inhaltsverzeichnis</b>	<b>II</b>
<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Hintergrund der Arbeit . . . . .	1
1.2 Problemstellung . . . . .	1
1.3 Lösungsansatz . . . . .	2
<b>2 Urlaubtracking Apps</b>	<b>3</b>
2.1 Bestehende Urlaubtracking Apps . . . . .	3
2.2 Beitrag der Künstlichen Intelligenz zum Urlaubstracking . . . . .	4
<b>3 Anforderungsanalyse</b>	<b>7</b>
3.1 Funktionale Anforderungen . . . . .	7
3.2 Nicht Funktionale Anforderungen . . . . .	8
<b>4 Konzeption der App</b>	<b>9</b>
4.1 Gesamt Architektur der App . . . . .	9
4.2 Aktivitätsdiagramm . . . . .	10
4.3 Sequenzdiagramme . . . . .	12
<b>5 Implementierung der App</b>	<b>17</b>
5.1 Entwicklungsumgebung . . . . .	17
5.2 Entwicklungsprozesses . . . . .	19
5.2.1 Projektmanagement Methodik . . . . .	19
5.2.2 Name der App . . . . .	20
5.2.3 Auswahl externer APIs . . . . .	20
5.2.4 Implementierung des Planvacation Fragment . . . . .	21
5.2.5 Implementierung des Voicetranslation Fragment: . . . . .	26
5.2.6 Implementierung des Summary Fragment . . . . .	28
5.3 Benutzeroberfläche . . . . .	34
<b>6 Ergebnisse</b>	<b>37</b>
<b>7 Zusammenfassung</b>	<b>38</b>
7.1 Zusammenfassung Der Arbeit . . . . .	38
7.2 Ausblick und zukünftige Entwicklungen . . . . .	39
<b>Literatur</b>	<b>40</b>

## Abbildungsverzeichnis

1	Aktivitätsdiagramm . . . . .	11
2	Sequenzdiagramm Zielort Bestimmung . . . . .	12
3	Sequenzdiagramm Urlaubvorschlag . . . . .	13
4	Sequenzdiagramm Ausgaben . . . . .	14
5	Sequenzdiagramm Sprachübersetzung . . . . .	15
6	Sequenzdiagramm Zusammenfassung . . . . .	16
7	Konstruktion der JSON-Anfrage . . . . .	21
8	Erstellung der POST-Anfrage . . . . .	22
9	Asynchroner Aufruf der CHAT GPT API . . . . .	23
10	onCreateView-Methode im MapFragment . . . . .	24
11	Funktion getAddressFromLatLng . . . . .	25
12	Methode stopRecording/startRecording . . . . .	26
13	Methode transcribeAudio . . . . .	27
14	Locationservice Initialisierung und Setup . . . . .	28
15	startLocationUpdates Methode . . . . .	29
16	Methode isPlaceOfInterest . . . . .	30
17	Methode onServiceConnected . . . . .	31
18	Methode onServiceDisconnected . . . . .	31
19	Methode getLastAddedImageWithLocation . . . . .	32
20	Methode getLocationNameFromImage . . . . .	33
21	Übersicht der Testfälle . . . . .	37

# 1 Einleitung

## 1.1 Hintergrund der Arbeit

Digitale Technologien werden in unserem Alltag immer wichtiger, besonders in einer zunehmend vernetzten Welt. Früher bereiteten sich Reisende traditionell mit gedruckten Reiseführern, handschriftlichen Notizen und Papierfotos aus dem Urlaub vor. Diese Praktiken gaben der Reiseplanung eine persönliche und fast rituelle Dimension, die mit der Vorfreude auf das bevorstehende Erlebnis verbunden war. Jedoch hat sich dies in den letzten Jahren stark gewandelt.

Das Web und mobile Geräte haben neu bestimmt, wie wir Reisen denken, planen und erleben. Online-Karten, Buchungsportale, Bewertungsseiten und sozialen Netzwerke ersetzen oder ergänzen alte Tools. Spezielle Reise-Apps bieten jetzt maßgeschneiderte Lösungen für fast jeden Teil des Reiseerlebnisses: Von der Inspirationssuche über die Planung bis zur Dokumentierung der Reise in Echtzeit.

Die digitale Evolution hat die Reiseplanung nicht nur bequemer und effizienter gemacht, sondern auch neue Möglichkeiten geschaffen, um das Reisen persönlicher und unvergesslicher zu gestalten. Die Nutzung einer breiten Auswahl an Informationen und Werkzeugen ist möglich, kann aber auch überwältigend sein. Es spielt eine wichtige Rolle, diese Daten und Ressourcen sinnvoll zu verwenden und in einer nahtlosen, benutzerfreundlichen Lösung zu integrieren, die den Bedürfnissen des Kunden gerecht wird.

## 1.2 Problemstellung

In der gegenwärtigen Ära der Digitalisierung, obwohl es einen Überfluss an Hilfsmitteln für Reisende gibt, tritt die wesentliche Herausforderung hervor: Die existierenden Anwendungen schaffen es oft nicht, eine wirklich integrierte, auf den Einzelnen zugeschnittene und interaktive Reiseerfahrung zu gewährleisten. Die gängigen Lösungen auf dem Markt neigen dazu, sich nur auf spezifische Segmente der Reiseplanung zu konzentrieren – beispielsweise auf das Buchen von Flügen, das Reservieren von Unterkünften oder das Zusammenstellen von Tagesausflügen. Dabei verlieren sie aus dem Blick, was eine Reise in ihrem Kern ausmacht: eine ganzheitliche Erfahrung, die sowohl gut geplante Elemente als auch Raum für spontane Abenteuer und persönliche Entdeckungen umfasst.

Es existiert ein spürbarer Mangel an einer ganzheitlichen Plattform, die nicht nur die unterschiedlichen Phasen einer Reise – angefangen bei der anfänglichen Planung über die eigentliche Durchführung bis hin zur nachträglichen Dokumentation – nahtlos miteinander verwebt, sondern den Nutzern auch ermöglicht, ihre Reisen individuell zu gestalten und dabei unvergessliche Momente zu erleben.

Die eigentliche Herausforderung liegt somit nicht ausschließlich in der Bereitstellung von Werkzeugen zur Abdeckung logistischer Aspekte der Reiseplanung. Vielmehr geht es darum, ein umfassendes System zu entwickeln, das die einzigartigen Vorlieben und Interessen jedes Reisenden in den Vordergrund rückt. Ein solches System müsste in der Lage sein, sich flexibel an verändernde Präferenzen und Umstände der Nutzer anzupassen, um nicht nur eine reibungslose Planung und Durchführung zu gewährleisten, sondern auch eine detaillierte Dokumentation der Reiseerlebnisse zu ermöglichen. Dadurch könnten Reisende ihre wertvollen Erinnerungen nicht nur festhalten, sondern auch auf eine Weise konservieren, die ihre persönlichen Geschichten und Entdeckungen widerspiegelt.

Ein ideales Reiseunterstützungssystem würde somit weit über die grundlegenden Funktionen hinausgehen und eine Plattform bieten, die es den Nutzern erlaubt, ihre Reisen tiefgreifend zu personalisieren und zu interagieren. Dies umfasst eine sorgfältige Berücksichtigung der vielfältigen Wünsche und Bedürfnisse, die Reisende vor, während und nach ihren Abenteuern haben. Durch die Schaffung eines solchen umfassenden und flexiblen Systems könnten Nutzer nicht nur ihre Reisen effizienter planen und durchführen, sondern auch eine reichere, bedeutungsvollere Erfahrung genießen, die ihre individuellen Geschichten und Erlebnisse in den Mittelpunkt stellt.

### 1.3 Lösungsansatz

Unser Ansatz zur Bewältigung dieser Herausforderungen basiert auf der Entwicklung einer umfassenden Urlaubstracking-App, die modernste Technologien und personalisierte Datenanalyse nutzt, um eine maßgeschneiderte Reiseerfahrung zu bieten. Die App integriert verschiedene Funktionen zur Reiseplanung, Echtzeit-Empfehlungen und persönlichen Dokumentation, unterstützt durch eine intuitive Benutzeroberfläche, die den Nutzern volle Kontrolle über ihre Reiseerfahrung gibt.

Durch die Nutzung von Künstlicher Intelligenz (KI) zur Analyse von Nutzerdaten und Präferenzen kann die App individuell zugeschnittene Vorschläge für Aktivitäten, Sehenswürdigkeiten und gastronomische Erlebnisse bieten. Zudem erkennt und dokumentiert die App automatisch besuchte Orte und erstellt eine personalisierte Zusammenfassung der Reise, inklusive Fotos, Notizen und besonderen Momenten. Diese ganzheitliche Herangehensweise zielt darauf ab, nicht nur die Planung und Durchführung von Reisen zu vereinfachen, sondern auch die Erstellung eines wertvollen und persönlichen Reiseberichts zu ermöglichen.

In dem kommenden Kapitel analysieren wir bestehende Urlaubstracking-Apps, um deren Funktionsumfang und Limitationen zu identifizieren. Dabei konzentrieren wir uns auf die Herausforderungen, die diese Apps nicht bewältigen können, und erörtern die Möglichkeiten, wie Künstliche Intelligenz (KI) zur Überwindung dieser Grenzen eingesetzt werden kann.

## 2 Urlaubtracking Apps

### 2.1 Bestehende Urlaubtracking Apps

In unserer heutigen, von Technologie durchdrungenen Welt, in der digitale Werkzeuge fast jeden Bereich unseres Lebens beeinflussen, verwundert es kaum, dass auch das Reisen durch technologische Neuerungen eine Aufwertung erfährt. Urlaubs-Tracker-Apps haben sich als unersetzbliche Begleiter für den modernen Globetrotter erwiesen. Sie bieten alles Mögliche, von der Haushaltsführung über die Übersetzung von Fremdsprachen bis hin zur Möglichkeit, ein digitales Reisetagebuch zu führen. Diese Apps machen nicht nur die Planung und Durchführung unserer Abenteuer einfacher, sondern ermöglichen es uns auch, unsere Erfahrungen auf frische und interaktive Weise zu dokumentieren und zu teilen.

Apps zur Ausgabenverfolgung, wie beispielsweise Splitwise, haben die Art und Weise, wie wir auf Reisen mit Finanzen umgehen, grundlegend verändert. Sie bieten eine detaillierte Aufschlüsselung unserer Ausgaben, die weit mehr als nur die einfache Aufteilung der Kosten zwischen Freunden oder Familienmitgliedern ermöglicht. Mit diesen Apps können wir unsere Ausgaben in Echtzeit erfassen, sie verschiedenen Kategorien zuordnen und so ein detailliertes Budget zusammenstellen. Dies hilft uns, den Überblick zu behalten und Überausgaben zu vermeiden. Die nahtlose Integration von Zahlungsdiensten wie PayPal vereinfacht darüber hinaus das Begleichen gemeinsamer Rechnungen erheblich. Obwohl diese Apps in puncto Organisation und finanzieller Transparenz viele Vorteile bieten, müssen Nutzer auch potenzielle Nachteile wie Datenschutzbedenken und die mögliche Abhängigkeit von einer ständigen Internetverbindung berücksichtigen.[1]

Technologische Fortschritte wie die, die Google Translate bietet, haben die Art, wie wir mit Sprachbarrieren umgehen, wenn wir ins Ausland reisen, revolutioniert. Die Möglichkeit, Texte durch das bloße Richten der Smartphone-Kamera auf ein Schild oder ein Dokument in Echtzeit zu übersetzen, hat es uns erheblich leichter gemacht, uns in Ländern zurechtzufinden, deren Sprache wir nicht sprechen. Moderne Übersetzungsapps setzen auf hochentwickelte Algorithmen des maschinellen Lernens, um die Präzision ihrer Übersetzungen stetig zu verbessern. Trotz dieser Fortschritte gibt es immer noch Herausforderungen, wie die Abhängigkeit von einer stabilen Internetverbindung und gelegentliche Ungenauigkeiten in den Übersetzungen, die in wichtigen Kommunikationssituationen zu Schwierigkeiten führen können.[2]

Digitale Reisetagebuch-Apps wie FindPenguins und Traverous ermöglichen es Reisenden, ihre Erlebnisse auf innovative Weise festzuhalten. Im Gegensatz zu traditionellen Tagebüchern bieten diese digitalen Plattformen Funktionen wie Geotagging, das automatische Erstellen von Routen und die Integration von Multimedia-Elementen. Diese Funktionen bereichern die Dokumentation der Reiseerfahrungen erheblich und bieten eine dynamische Plattform zur Erstellung einer digitalen Chronik der Reise. Die Herausforderungen bei digitalen Reisetagebüchern liegen oft in den kostenpflichtigen Abonnements für erweiterte Funktionen und in Datenschutzbedenken hinsichtlich der gespeicherten Inhalte.

Die präzise Vorhersage von Wetterbedingungen spielt eine entscheidende Rolle bei der Planung von Aktivitäten auf Reisen. Dank moderner Wetter-Apps erhalten Reisende umfassende Informationen über die Wetterbedingungen an ihren Zielen, darunter stündliche Prognosen, Warnhinweise bei extremen Wetterverhältnissen und sogar Angaben zur Luftqualität. Diese Apps bedienen sich komplexer Algorithmen und Satellitendaten, um möglichst präzise Auskünfte zu geben. Dennoch gibt es Herausforderungen, wie zum Beispiel die Zuverlässigkeit der Wettervorhersagen in weniger gut untersuchten Gebieten und das Problem, dass die Fülle an Daten überwältigend sein kann.

Für viele Reisende sind Navigations-Apps zu einem unverzichtbaren Werkzeug geworden, besonders wenn sie sich in unbekannten Gegenden bewegen. Apps wie Google Maps liefern nicht nur detaillierte Anweisungen für Autofahrer, Fußgänger und Nutzer des öffentlichen Verkehrs, sondern bieten auch wertvolle Informationen über Sehenswürdigkeiten, Restaurants und Unterkünfte. Besonders praktisch ist die Möglichkeit, Kartenmaterial für die Offline-Nutzung herunterzuladen, was in Gebieten mit schlechter Internetanbindung von großem Nutzen sein kann. Trotzdem können die Aktualität und die Genauigkeit der Karten in weniger bekannten oder besuchten Destinationen manchmal zu Problemen führen.

Zusammenfassend bieten Urlaubtracking-Apps eine breite Palette an Dienstleistungen, die von der finanziellen Organisation bis hin zur Überwindung von Sprachbarrieren reichen. Während sie unbestreitbare Vorteile bieten, um die Reiseerfahrung zu verbessern, ist es wichtig, auch potenzielle Nachteile wie Datenschutzbedenken und die Abhängigkeit von technologischer Infrastruktur in Betracht zu ziehen.

## 2.2 Beitrag der Künstlichen Intelligenz zum Urlaubstracking

Die Einführung Künstlicher Intelligenz (KI) in den Tourismus markiert einen Meilenstein in unserem Reiseverhalten und der Planung unserer Urlaubserlebnisse. Durch KI-Technologien sind nicht nur die Effizienz und die Benutzerfreundlichkeit bei der Reiseplanung gestiegen, sondern es wurden auch maßgeschneiderte und anpassungsfähige Reiseerlebnisse möglich gemacht, die früher kaum vorstellbar waren.

Die automatisierte Planung von Reiserouten ist ein Bereich, in dem KI bahnbrechende Verbesserungen ermöglicht hat. Algorithmen, die lernfähig sind, erstellen Reiserouten nicht nur auf Grundlage logistischer Aspekte wie Wetterverhältnisse und Verkehrslage, sondern nehmen auch die persönlichen Vorlieben und Interessen der Reisenden in den Blick. Diese Algorithmen schöpfen aus einem breiten Spektrum von Datenquellen und berücksichtigen dabei sowohl Bewertungen von Sehenswürdigkeiten als auch historische Wetterdaten und aktuelle Verkehrsinformationen. Das Ergebnis sind individuell zugeschnittene Reisepläne, die genau auf die Bedürfnisse der Nutzer abgestimmt sind und es ihnen ermöglichen, ihre Reisezeit optimal zu nutzen.

In der Welt des Reisens sind Echtzeit-Informationen unverzichtbar. KI-Systeme leisten hier einen entscheidenden Beitrag, indem sie ständig Daten von Fluggesellschaften, meteorologischen Diensten und Verkehrsbehörden analysieren. So können Reisende zeitnah über Flugverspätungen, plötzliche Wetterumschwünge oder Staus informiert werden. Diese Fähigkeit, nahezu in Echtzeit auf Veränderungen zu reagieren, ermöglicht es Reisenden, ihre Pläne flexibel anzupassen und mögliche Unannehmlichkeiten auf ein Minimum zu reduzieren.[3]

Chatbots und virtuelle Assistenten, angetrieben durch fortschrittliche KI, haben die Interaktion zwischen Reiseanbietern und Kunden revolutioniert. Diese Technologien bieten nicht nur Unterstützung bei der Beantwortung von Fragen und der Vornahme von Buchungen, sondern sind auch fähig, basierend auf dem Verhalten und den Vorlieben der Nutzer personalisierte Empfehlungen abzugeben. Von der Empfehlung eines verborgenen Juwels unter den Restaurants bis hin zur Vorstellung einer wenig bekannten kulturellen Veranstaltung – KI-Assistenten bereichern das Reiseerlebnis durch maßgeschneiderte Vorschläge.[4]

Personalisierte Empfehlungen durch KI reichen weit über die Auswahl von Restaurants und Unterkünften hinaus. KI-Systeme nutzen komplexe Algorithmen, um das bisherige Such- und Buchungsverhalten der Nutzer zu analysieren und daraus Vorlieben abzuleiten. Diese Informationen werden genutzt, um individuell zugeschnittene Empfehlungen für Reiseziele, Aktivitäten und sogar Reisezeiten zu generieren, was den Planungsprozess vereinfacht und gleichzeitig die Vorfreude auf die Reise steigert.[5]

Reisetagebuch-Apps, die mit Künstlicher Intelligenz (KI) angereichert sind, bieten eine innovative Möglichkeit für Reisende, ihre Erlebnisse aufzuzeichnen. Diese Apps nutzen KI, um automatische Einträge über besuchte Orte, Wetterbedingungen und durchgeführte Aktivitäten zu erstellen. Sie bieten personalisierte Empfehlungen für Sehenswürdigkeiten und Restaurants, versehen Fotos automatisch mit Tags und Beschreibungen und verarbeiten Sprachaufzeichnungen sowie Texte effizient. Außerdem können sie, basierend auf den Vorlieben des Nutzers und dessen früheren Reisen, individuell zugeschnittene Reisevorschläge machen. Während diese Funktionen die Reisedokumentation erheblich bereichern und personalisieren, hängt ihre Effektivität letztendlich von der Präzision der KI-Analyse und der Relevanz der zugrunde liegenden Daten ab.

Bessere Sicherheit und Gesundheitsüberwachung durch KI tragen entscheidend dazu bei, das Wohlbefinden der Reisenden zu sichern. KI-Systeme überwachen globale Sicherheits- und Gesundheitsmeldungen in Echtzeit und können Reisende proaktiv über potenzielle Risiken in ihrem Zielgebiet informieren. Ob es um Ausbrüche von Krankheiten, politische Unruhen oder Naturkatastrophen geht die frühzeitige Warnung durch KI-Technologien ermöglicht es Reisenden, informierte Entscheidungen zu treffen und ihre Pläne entsprechend anzupassen.

Die Integration von Künstlicher Intelligenz (KI) in Urlaubstracking-Apps markiert tatsächlich eine revolutionäre Entwicklung, durch die Reisen nicht nur effizienter und persönlicher, sondern auch sicherer werden. Dank der fortlaufenden Verbesserung dieser Technologien dürfen wir uns auf eine Zukunft freuen, die noch stärker auf individuelle Bedürfnisse zugeschnittene Reiseerlebnisse bietet und Reisenden eine noch nie dagewesene Flexibilität ermöglicht.

### 3 Anforderungsanalyse

Die Anforderungsanalyse ist ein kritischer Schritt in der Entwicklung von Softwareanwendungen. Sie definiert klar und präzise, was das System tun soll (funktionale Anforderungen) und wie gut es dies tun soll (nicht-funktionale Anforderungen). Für unsere Urlaubstracking-App umfasst diese Analyse eine detaillierte Untersuchung der Nutzerbedürfnisse und des gewünschten Systemverhaltens.

#### 3.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben die spezifischen Funktionen und Prozesse der App. Für jede Funktion werden die Erwartungen und das Verhalten aus Nutzersicht detailliert beschrieben.

- **Zielort Bestimmung:**

- **Nutzerinteraktion:** Die App muss eine benutzerfreundliche Schnittstelle bereitstellen, durch die Nutzer ihren Zielort für den Urlaub auswählen können. Dies sollte sowohl durch Eingabe des Ortsnamens als auch durch Auswahl auf einer interaktiven Karte möglich sein. Ein weiteres Feature ist die Integration einer Auswahlmöglichkeit für Start- und Enddatum des Urlaubs

- **Systemreaktion:** Sobald Nutzer ihren Zielort samt Start- und Enddatum des Urlaubs festgelegt haben, speichert das System diese Daten, um individuell zugeschnittene Urlaubsvorschläge anzubieten. Die gespeicherten Daten werden verwendet, um relevante Informationen zum Zielort bereitzustellen, wie beispielsweise bevorstehende lokale Ereignisse und maßgeschneiderte Aktivitäten.

- **Urlaubsvorschläge Generieren:**

- **Nutzerinteraktion:** Basierend auf dem gewählten Zielort und den Reisedaten sollen Nutzer personalisierte Urlaubsvorschläge erhalten.

- **Systemreaktion:** Das System soll eine Liste von Aktivitäten und Sehenswürdigkeiten präsentieren, die auf den Interessen und Präferenzen des Nutzers basieren.

- **Ausgabenverwaltung:**

- **Nutzerinteraktion:** Nutzer sollen in der Lage sein, Ausgaben zu erfassen, indem sie Details wie Betrag, Datum, Kategorie und Zweck der Ausgabe angeben.

- **Systemreaktion:** Das System soll diese Ausgaben in einem visuellen Chart darstellen, der eine Analyse der täglichen Ausgaben ermöglicht.

- **Sprachübersetzung:**

- **Nutzerinteraktion:** Nutzer sollen Zugang zu einer Echtzeit-Übersetzungsfunktion haben, die es ihnen ermöglicht, gesprochene Sprache sofort zu übersetzen, was die Kommunikation in fremden Ländern erleichtert.

- **Systemreaktion:** Die App soll akustische Eingaben erfassen, sie zur Verarbeitung an eine Übersetzungs-API senden und das übersetzte Ergebnis in Echtzeit ausgeben.

- **Tägliche Zusammenfassung:**

- **Nutzerinteraktion:** Am Ende jedes Tages sollen Nutzer eine automatisch erstellte Zusammenfassung ihres Urlaubstages erhalten, die besuchte Orte, zurückgelegte Wege, aufgenommene Fotos und Ausgaben umfasst.

- **Systemreaktion:** Das System soll die Zusammenfassung generieren, indem es Daten von Google Maps für die Routennachverfolgung, das Foto-Speichersystem für Bilduploads und das Ausgaben-Chart-System integriert.

## 3.2 Nicht Funktionale Anforderungen

Während funktionale Anforderungen das "Was" definieren, beschreiben nicht-funktionale Anforderungen das "Wie" der App-Leistung, einschließlich Geschwindigkeit, Zuverlässigkeit, Sicherheit und Benutzererfahrung.

- **Performance:**

Die App muss schnell und reaktionsfähig sein, mit minimalen Ladezeiten für das Abrufen von Informationen und der Generierung von Urlaubsvorschlägen. Die Performance-Metriken sollten spezifiziert werden.

- **Benutzerfreundlichkeit:**

Ein Hauptaugenmerk liegt auf einer intuitiven Benutzeroberfläche, die auch für Erstnutzer leicht zu navigieren ist. Dies umfasst klare Menüführungen, verständliche Icons und eine logische Anordnung von Funktionen.

- **Zuverlässigkeit:**

Die App muss eine hohe Verfügbarkeit aufweisen und korrekt funktionieren, auch unter hoher Last oder bei schlechter Internetverbindung. Datenintegrität, besonders bei der Ausgabenverfolgung und -speicherung, ist entscheidend.

- **Skalierbarkeit:**

Das System muss in der Lage sein, mit einer wachsenden Anzahl von Nutzern und Daten zu skalieren, ohne dass es zu einem Verlust der Servicequalität kommt.

- **Interoperabilität:**

Die App sollte gut mit externen APIs und Diensten (wie Google Maps und Übersetzungs-APIs) zusammenarbeiten und flexibel genug sein, um zukünftige Integrationen zu unterstützen.

## 4 Konzeption der App

### 4.1 Gesamt Architektur der App

Die Architektur dieser App ist speziell für Android-Geräte entwickelt und nutzt Android Studio, um eine effiziente und effektive Entwicklungsplattform bereitzustellen. Die Entwicklung teilt sich in Frontend und Backend, wobei XML für das Design der Benutzeroberfläche und Java für die Backend-Logik verwendet werden.

- **Frontend:**

Das Frontend der App ist in XML (Extensible Markup Language) gestaltet. XML wird verwendet, um die Layouts der Benutzeroberfläche zu definieren, einschließlich Bildschirmelemente wie Buttons, Textfelder und Kartenansichten. Dies ermöglicht eine klare Trennung zwischen der Gestaltung der Benutzeroberfläche und der Geschäftslogik der App. Die Hauptelemente des Frontends umfassen:

- Interaktive Karten und Ortsauswahl für die Bestimmung des Zielorts, inklusive eines Kalenders für Start- und Enddatum.
- Dynamische Anzeige von Urlaubsvorschlägen, angepasst an die Eingaben und Vorlieben des Nutzers.
- Ausgabenverwaltung, die es den Nutzern ermöglicht, ihre finanziellen Aufwendungen während der Reise zu verfolgen.
- Echtzeit-Sprachübersetzung, um die Kommunikation in fremden Sprachen zu erleichtern.
- Generierung einer täglichen Zusammenfassung der Reiseaktivitäten und Ausgaben.

- **Backend:**

Das Backend ist in Java programmiert und handhabt die gesamte Geschäftslogik der App. Java wird für die Datenverarbeitung, das Management der Benutzerinteraktionen und die Kommunikation mit externen APIs verwendet. Die Hauptfunktionen des Backends umfassen:

- Verarbeitung der Zielortauswahl und Speicherung der Reisedaten.
- Anbindung an externe APIs wie ChatGPT für personalisierte Inhalte und Google Maps für geografische Daten.
- Management von Nutzerausgaben und deren Darstellung in einer übersichtlichen Form.
- Umsetzung der Sprachübersetzung durch Integration entsprechender Übersetzungs-APIs.

- Erstellung von täglichen Zusammenfassungen, indem Daten von verschiedenen Aktivitäten und Standorten des Tages aggregiert werden.

- **Externe Dienste:**

- ChatGPT API: Bietet personalisierte Urlaubsvorschläge basierend auf den Präferenzen und Interessen des Nutzers.
- Google Maps API: Ermöglicht die Nutzung geographischer Daten, Routenplanung und Standortbestimmung.

## 4.2 Aktivitätsdiagramm

Das Diagramm zeigt die Schritte zur Planung eines Urlaubs mit unserer Reise-App. Nach dem Start der App wählt der Benutzer ein Ziel durch Eingabe oder Auswahl auf der Karte und legt das Datum fest. Basierend auf diesen Informationen schlägt die App einen Reiseplan vor, der gespeichert und später betrachtet werden kann. Zusätzlich bietet die App eine Funktion für Sprachübersetzung, die hilft, Sprachbarrieren zu überwinden. Nach Eingabe der Ausgaben erstellt die App eine übersichtliche Zusammenfassung des Urlaubs, die der Benutzer bearbeiten und nach Wunsch teilen kann

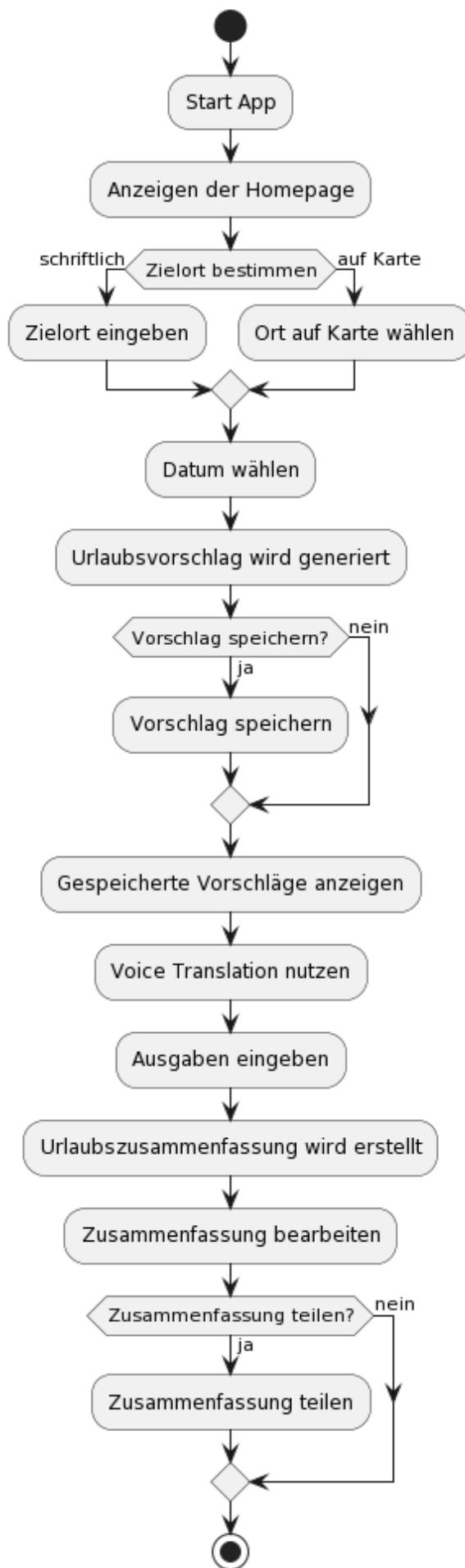


Abbildung 1: Aktivitätsdiagramm

### 4.3 Sequenzdiagramme

- Zielort bestimmen:**

Das angezeigte Sequenzdiagramm zeigt die Schritte, die ein Benutzer und das System durchlaufen, um ein Ziel in der App festzulegen. Der Benutzer hat die Möglichkeit, entweder das Ziel direkt schriftlich einzugeben oder es auf einer interaktiven Karte auszuwählen. Nachdem er das Ziel eingegeben hat, wird eine Anfrage an das Backend-System gesendet, das geeignete Vorschläge generiert und sie zurück an die Benutzeroberfläche sendet. Bei der Auswahl auf der Karte wird der gewählte Standort bestätigt und vom Backend gespeichert. Schließlich erhält der Benutzer eine Bestätigung der erfolgreichen Speicherung seines gewählten Ziels. Dieser Prozess demonstriert die effiziente Interaktion zwischen Frontend und Backend der App, um ein benutzerfreundliches Erlebnis zu bieten.

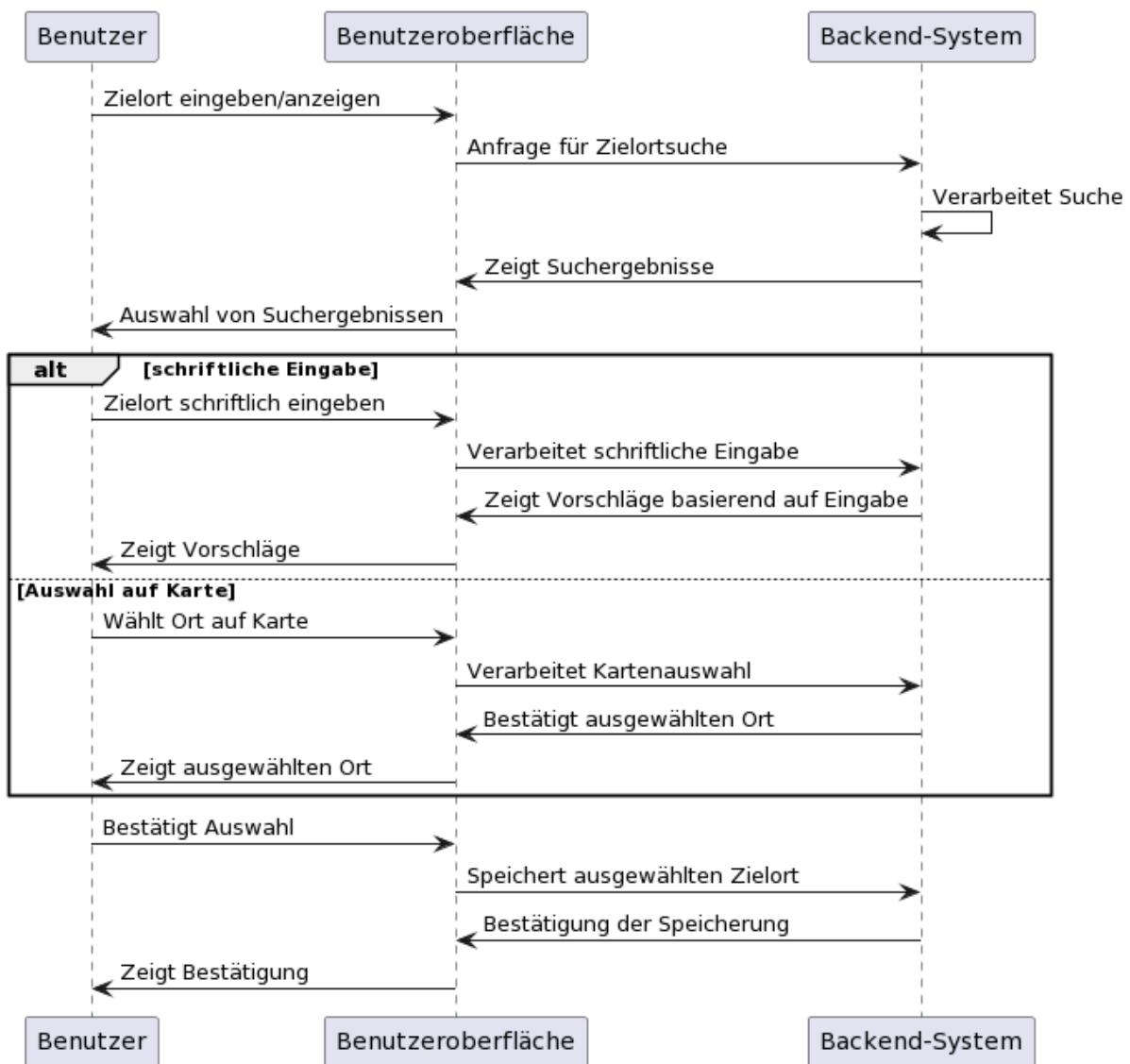
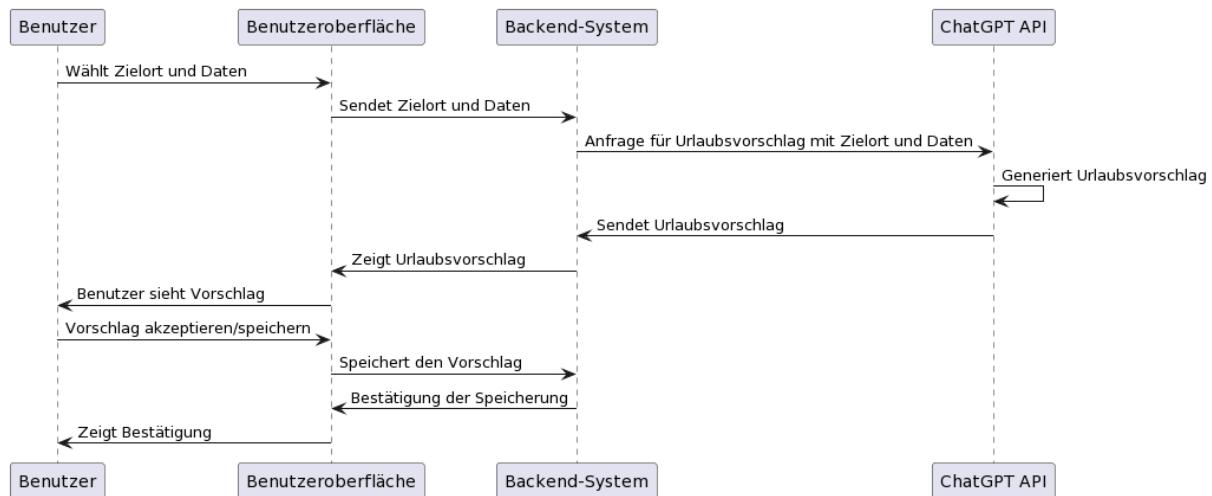


Abbildung 2: Sequenzdiagramm Zielort Bestimmung

- **Urlaubsvorschlag generieren:**

Das Sequenzdiagramm zeigt die Schritte bei der Erstellung eines personalisierten Urlaubsvorschlags in unserer App. Der Benutzer gibt zuerst das gewünschte Reiseziel und die Reisedaten ein. Diese Informationen werden an das Backend-System gesendet, das dann eine Anfrage an die ChatGPT-API sendet, um einen benutzerdefinierten Vorschlag zu erhalten. Sobald der Urlaubsvorschlag erstellt ist, zeigt ihn das Backend auf der Benutzeroberfläche an, wo der Benutzer die Möglichkeit hat, ihn zu akzeptieren und zu speichern. Schließlich wird dem Benutzer eine Bestätigung der erfolgreichen Speicherung angezeigt.



**Abbildung 3:** Sequenzdiagramm Urlaubsvorschlag

- Ausgaben verfolgen:**

Das Diagramm veranschaulicht den Prozess der Erfassung und Darstellung von Ausgaben in der App. Der Benutzer trägt die Details der Ausgaben über ein Formular ein, welches Datum, Zweck, Kategorie und Betrag umfasst. Nach der Dateneingabe sendet die Benutzeroberfläche die Informationen an das Backend-System, das die Daten verarbeitet und speichert. Anschließend wird ein aktualisiertes Chart von der Chart-Komponente generiert, das die neuen Ausgaben reflektiert und dem Benutzer zur Ansicht bereitgestellt wird. Dies ermöglicht dem Benutzer, seine Finanzen effektiv zu verwalten und den Überblick über seine Ausgaben zu behalten.

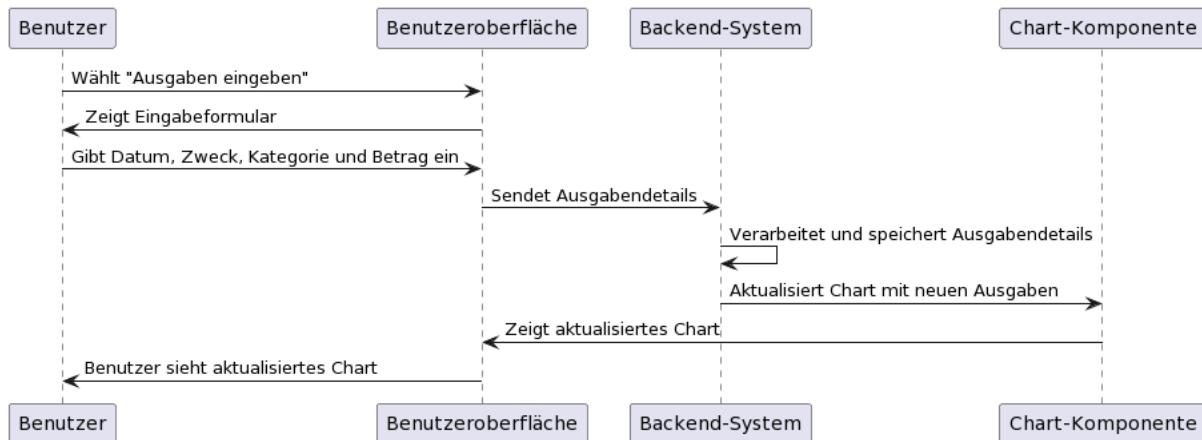
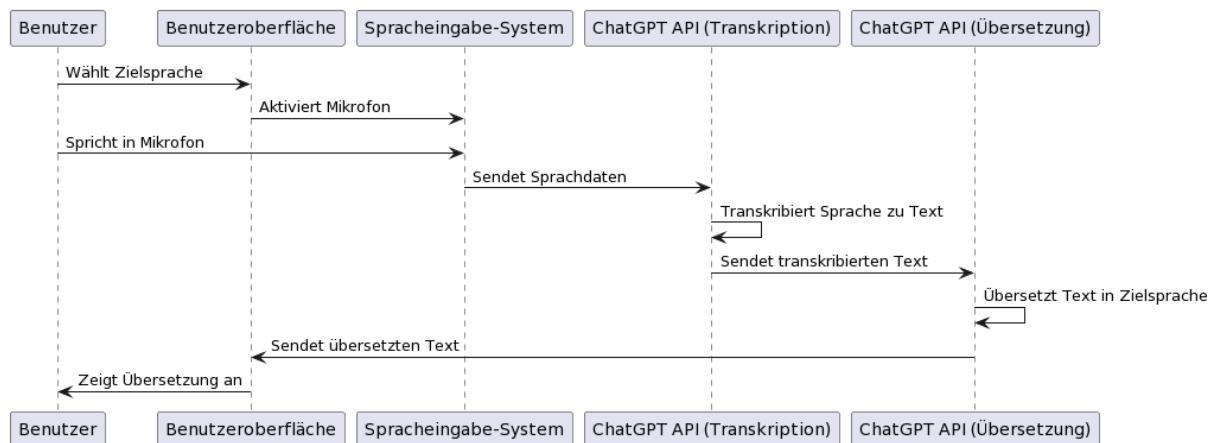


Abbildung 4: Sequenzdiagramm Ausgaben

- Sprachübersetzung:**

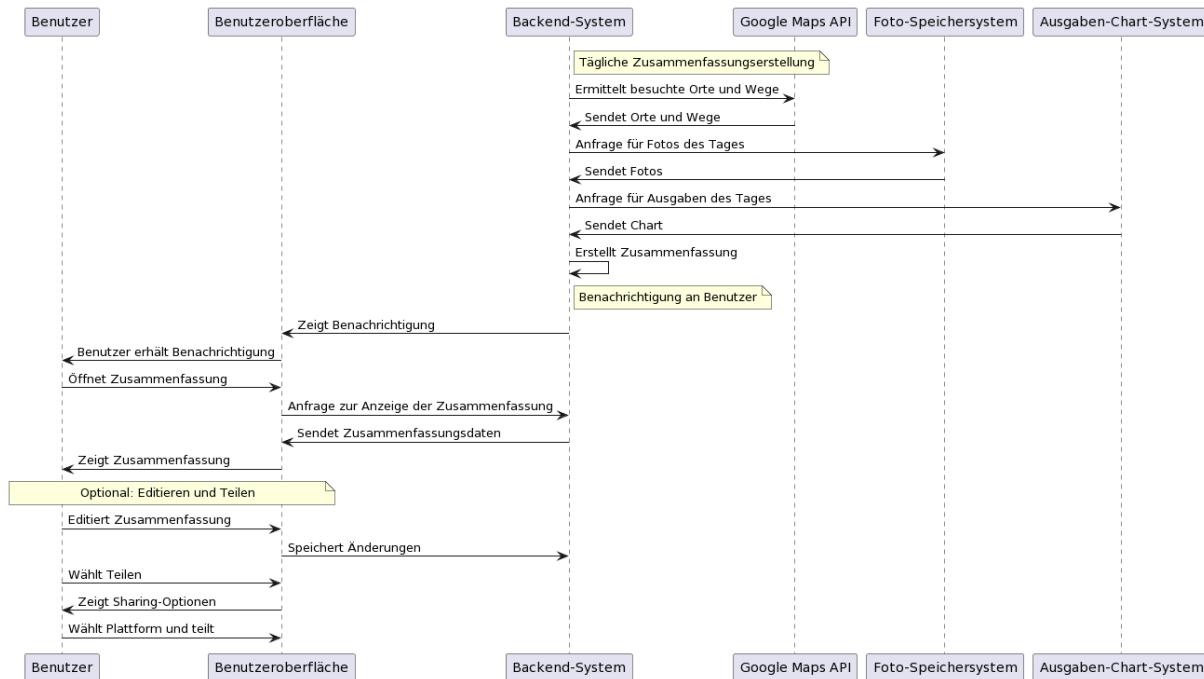
Das Diagramm zeigt, wie die App die Sprachaufnahme und Übersetzung durchführt. Zuerst wählt der Benutzer über die Benutzeroberfläche die gewünschte Sprache aus, bevor er das Mikrofon aktiviert, um die Spracheingabe zu beginnen. Die Benutzeroberfläche sendet die Audiodaten an das Spracheingabe-System, das sie zur Transkription an die ChatGPT API weiterleitet. Die API transkribiert Audiodaten in Text und sendet sie an die ChatGPT API zur Übersetzung. Dort wird der Text in die gewünschte Sprache übersetzt und zurück an die Benutzeroberfläche gesendet, die die Übersetzung zeigt.



**Abbildung 5:** Sequenzdiagramm Sprachübersetzung

- **Tageszusammenfassung erstellen:**

Die App erstellt eine individuelle Tageszusammenfassung für den Benutzer, wie im Sequenzdiagramm dargestellt. Das Backend-System sammelt zunächst Daten von verschiedenen Quellen. Dazu gehören Google Maps API, Fotos des Tages aus einem internen Fotospeichersystem und die täglichen Ausgaben aus dem Ausgaben-Chart-System. Der Benutzer erhält eine Benachrichtigung nach der Zusammenstellung dieser Informationen. Dann hat der Benutzer die Möglichkeit, die Zusammenfassung zu öffnen, zu bearbeiten, zu speichern und nach Wunsch direkt zu teilen.



**Abbildung 6:** Sequenzdiagramm Zusammenfassung

## 5 Implementierung der App

### 5.1 Entwicklungsumgebung

Die Entwicklung unserer Urlaub-Tracking-App stützt sich auf zwei tragende Säulen der modernen Android-Entwicklung: Java als Programmiersprache und Android Studio als IDE(Integrated Development Environment)

- **JAVA:**[6]

Entwickelt im Jahr 1995 von James Gosling bei Sun Microsystems, startete Java ursprünglich als das Projekt OAK im Bereich des interaktiven Fernsehens und entwickelte sich zu einer objektorientierten Programmiersprache, die plattformunabhängig ist. Sie ist vor allem für ihr Prinzip des "write once, run anywhere"(WORA) bekannt. Diese Maxime erlaubt es Programmierern, einen Code zu erstellen, der auf jeder Plattform mit einer installierten Java Virtual Machine (JVM) läuft, was Java zu einer bevorzugten Sprache für die Entwicklung von vielfältigen Anwendungen, Unternehmenssoftware und Smartphone-Betriebssystemen macht. Die Syntax von Java weist Ähnlichkeiten mit der von C++ auf, was Programmierern den Einstieg erleichtert. Als Reaktion auf die wachsende Vernetzung von Geräten und als Teil der digitalen Welle der 1990er Jahre erlangte Java schnell eine Schlüsselposition in der Welt der Softwareentwicklung. Mehr als zwei Jahrzehnte später bleibt Java wegen seiner Flexibilität, der starken Orientierung an Objekten und der Fähigkeit zur plattformübergreifenden Nutzung eine der weltweit führenden Programmiersprachen.

Durch seinen konsequent objektorientierten Ansatz erleichtert Java die Modellierung komplexer Systeme, fördert die Wiederverwendbarkeit von Code und unterstützt die Erstellung modularer Anwendungsarchitekturen. Diese Merkmale sind besonders wertvoll bei der Entwicklung umfangreicher Anwendungen, wie zum Beispiel einer Urlaub-Tracking-App, da sie die Pflege und das Skalieren des Projekts erleichtern. Ein wesentlicher Vorteil von Java ist auch seine Plattformunabhängigkeit, die durch die Java Virtual Machine (JVM) gewährleistet wird und die Entwicklung von Anwendungen, die plattformübergreifend funktionieren, stark vereinfacht. Zudem profitiert Java von einer umfassenden und aktiven globalen Entwicklergemeinschaft, einer Vielzahl von Frameworks und reichhaltiger Dokumentation, die zusammen die Bewältigung von Entwicklungsproblemen erleichtern.

Obwohl Java viele Vorteile bietet, gibt es auch Herausforderungen, die berücksichtigt werden müssen. Die automatische Speicherverwaltung und die Garbage Collection, obwohl hilfreich für die Verwaltung des Speichers, können in bestimmten Situationen zu unvorhersehbaren Verzögerungen führen. Dies kann besonders in mobilen Umgebungen zu Schwierigkeiten führen. Weiterhin kann die Performance von Java-Anwendungen im Vergleich zu Sprachen wie C++, die direkt auf der Hardware laufen, durch die Notwendigkeit, Bytecode über die JVM zu interpretieren, beeinträchtigt sein.

- **Android Studio:**[7]

Seit seiner Ankündigung im Mai 2013 und der offiziellen Veröffentlichung der Version 1.0 am 8. Dezember 2014 hat Android Studio die Entwicklung von Android-Apps maßgeblich geprägt. Als von Google offiziell unterstützte Entwicklungsumgebung für sein Betriebssystem hat Android Studio den Weg für eine neue Generation der App-Entwicklung geebnet. Die Einführung der ersten Preview-Version von 1.3 im Mai 2015 verstärkte diese Entwicklung weiter, indem das bisher genutzte Android SDK vollständig in Android Studio integriert wurde, was die Entwicklungsprozesse weiter vereinheitlichte und verbesserte.

Android Studio, das auf JetBrains IntelliJ IDEA basiert, nutzt Java als Hauptprogrammiersprache, bietet aber auch Unterstützung für andere Sprachen, vor allem für Kotlin, das sich großer Beliebtheit erfreut. Die Wahl, IntelliJ IDEA als Grundlage zu nutzen, stellt Entwicklern eine leistungsfähige und benutzerfreundliche Plattform für die Erstellung von Android-Apps zur Verfügung. Ein zentrales Merkmal von Android Studio ist das auf Gradle basierende Build-Management-Tool. Gradle bietet eine anpassungsfähige und effektive Möglichkeit, Builds zu konfigurieren, die den vielfältigen Anforderungen der modernen App-Entwicklung gerecht wird.

Android Studio zeichnet sich durch eine breite Palette an Entwicklungswerkzeugen aus, die von einem effizienten Code-Editor, über fortschrittliche Debugging-Tools, bis hin zu einem vielseitigen Build-System reichen. Entwicklern wird zusätzlich durch Tools wie einen visuellen Layout-Editor, einen APK Analyzer, Profiling-Tools und einen umfangreichen Emulator unterstützt. Die problemlose Verknüpfung mit Google-Diensten, einschließlich der Google Cloud Platform und Firebase, vereinfacht die Integration von Features wie Datenbankmanagement, Authentifizierung und Cloud-Speicher. Trotz der regelmäßigen Updates, die Entwicklern die neuesten Funktionen und bewährte Methoden zur Verfügung stellen, kann die hohe Ressourcenanforderung von Android Studio, besonders auf älteren oder weniger leistungsfähigen Systemen, zu Einbußen in der Performance führen. Zudem könnte die Fülle an Funktionen und Konfigurationsmöglichkeiten für Neulinge in der App-Entwicklung zunächst eine Herausforderung darstellen und eine gewisse Einarbeitungszeit erfordern.

Die Verwendung von Java in Kombination mit Android Studio stellt eine starke Basis für die Entwicklung unserer Urlaub-Tracking-App dar. Diese Kombination erlaubt eine effiziente Handhabung des gesamten Entwicklungsprozesses und legt gleichzeitig den Grundstein für eine hohe Qualität der Anwendung, Benutzerfreundlichkeit und die Möglichkeit zur zukünftigen Erweiterung.

## 5.2 Entwicklungsprozesses

### 5.2.1 Projektmanagement Methodik

Zur Entwicklung unserer Urlaubstracking-App wurde das Wasserfallmodell, eine klassische Methode im Projektmanagement, verwendet. Dieses Modell mit seiner linearen und sequenziellen Herangehensweise erwies sich als ideal für die strukturierte und gut definierte Projektsteuerung. Durch die Unterteilung des Entwicklungsprozesses in fünf klar definierte Phasen konnte das Projekt effizient und systematisch vorangetrieben werden.

- **Phase 1: Anforderungsanalyse** Wir haben uns in der Anfangsphase mit den spezifischen Anforderungen der Urlaubstracking-App auseinandergesetzt und diese definiert. Die Planung umfasste eine detaillierte Planung, die alle notwendigen Funktionen, die geplante Benutzererfahrung und die technische Machbarkeit der App umfasste.
- **Phase 2: Entwurf** Auf der Grundlage der festgelegten Anforderungen wurde ein konkretes Lösungskonzept für die App entwickelt. In dieser Phase wurden die notwendigen Technologien und Tools ausgewählt, um die geplante Architektur und Benutzeroberfläche der App detailliert darzustellen.
- **Phase 3: Implementierung** Die App wurde auf der Grundlage des festgelegten Entwurfs entwickelt. Diese Phase umfasste die Programmierung der Kernfunktionalitäten, die Gestaltung der Benutzeroberfläche und die Integration der verschiedenen App-Komponenten. Die Aufteilung in kleinere, überschaubare Teilaufgaben ermöglichte regelmäßige Fortschritte und die effektive Steuerung der Entwicklung.
- **Phase 4: Testen** Nach der Implementierung wurde eine umfassende Testphase durchgeführt, um sicherzustellen, dass die App den definierten Anforderungen entsprach und keine Fehler aufweist. Diese Maßnahme ermöglichte es Probleme frühzeitig zu erkennen und zu beheben.
- **Phase 5: Einsatz und Wartung** Nach dem Abschluss der Testphase wurde die App für den internen Gebrauch eingeführt.

Durch die Anwendung des Wasserfallmodells konnte der Entwicklungsprozess klar strukturiert und schrittweise eine funktionale und benutzerfreundliche Urlaubstracking-App realisiert werden.

### 5.2.2 Name der App

Die Urlaubs-Tracking-App "TripSquirrel" wurde genannt. Der Name leitet sich von "Tri-päb, was Reise bedeutet, und SSquirrel", was für seine Agilität und Sammelfreude bekannt ist. Die Hauptfunktionen der App sind das Sammeln und Organisieren von Reiseinformationen sowie die flexible Anpassung an die Bedürfnisse der Nutzer. TripSquirrel funktioniert wie ein Eichhörnchen, das Vorräte für später aufbewahrt. Es ermöglicht den Nutzern, ihre Reisepläne, Erfahrungen und Erinnerungen zu speichern und jederzeit darauf zuzugreifen, was die Planung und Durchführung von Reisen erleichtert.

### 5.2.3 Auswahl externer APIs

Bei der Entwicklung der Urlaubstracking-App war die Integration externer APIs entscheidend, um leistungsfähige und benutzerfreundliche Funktionen anzubieten. Zwei Schlüsseltechnologien, die Google Maps API und die OpenAI's ChatGPT API, spielten eine zentrale Rolle in der Realisierung unserer Kernfunktionen: Standortverfolgung, Wegdarstellung und die Generierung personalisierter Urlaubsvorschläge.

- Google Maps API: Die Entscheidung, die Google Maps API zu nutzen, beruhte auf der Notwendigkeit, unseren Nutzern eine präzise und interaktive Standortverfolgung sowie Wegdarstellungen anzubieten. Diese API ermöglicht es uns, auf einer Karte nicht nur die aktuellen Standorte unserer Nutzer in Echtzeit zu präsentieren, sondern auch die Wege zwischen verschiedenen Orten visuell ansprechend darzustellen. Dies ist besonders wertvoll für die tägliche Zusammenfassung, in der die Nutzer sehen können, welche Strecken sie während ihres Urlaubs zurückgelegt haben. Die intuitive Schnittstelle und die umfangreichen Funktionen der Google Maps API ermöglichen es uns, komplexe Informationen einfach und verständlich darzustellen.
- ChatGPT API von OpenAI: Wir haben uns für die Integration der ChatGPT-API von OpenAI entschieden, um unseren Nutzern personalisierte und intelligente Urlaubsvorschläge zu bieten. Diese Entscheidung wurde durch die Fähigkeit der API motiviert, natürliche Sprache zu verstehen und darauf aufbauend informative, kreative und auf den Nutzer zugeschnittene Antworten zu generieren. Die ChatGPT API ermöglicht es unserer App, die Präferenzen und Interessen der Nutzer zu analysieren und ihnen individuell angepasste Vorschläge für Urlaubsziele, Aktivitäten und Erlebnisse zu unterbreiten. Die Einbindung dieser KI-getriebenen Empfehlungen unterscheidet unsere App von traditionellen Urlaubsplanungs-Tools und bietet einen echten Mehrwert für die Nutzererfahrung.

#### 5.2.4 Implementierung des Planvacation Fragment

Das PlanVacationFragment ermöglicht es den Nutzern, ihre Urlaubsplanung innerhalb der App zu organisieren. Die Benutzer können Ziele angeben, Daten auswählen und persönliche Präferenzen eintragen.

##### - Verwendete Bibliotheken:

OkHttp ist eine effiziente HTTP-Client-Bibliothek, die für Android und Java-Anwendungen optimiert ist. Sie bietet eine zuverlässige Plattform für Netzwerkanfragen und ist besonders geeignet für die Integration von REST-basierten API-Diensten wie der ChatGPT API:

- Netzwerkinteraktionen: OkHttp verwaltet Netzwerkverbindungen effizient und ermöglicht Features wie Verbindungspooling, GZIP-Kompression und Caching. Für Die App bedeutet dies eine zuverlässige und schnelle Kommunikation mit externen Diensten, die für das Abrufen von Urlaubsvorschlägen genutzt wird.
- Asynchrone Anfragen: Durch den Einsatz asynchroner Callbacks kann OkHttp die UI nicht blockieren. Dies ist besonders wichtig, wenn Die App komplexe Anfragen verarbeitet.

##### - Nutzung der ChatGPT API mit OkHttp3:

###### 1- Konstruktion der JSON-Anfrage

```
void callAPI(String question) {  
  
    JSONObject jsonBody = new JSONObject();  
    try {  
        jsonBody.put("name: "model", "value: "gpt-3.5-turbo-instruct");  
        jsonBody.put("name: "prompt", "question");  
        jsonBody.put("name: "max_tokens", "value: 4000);  
        jsonBody.put("name: "temperature", "value: 0);  
    } catch (JSONException e) {  
        throw new RuntimeException(e);  
    }  
}
```

Abbildung 7: Konstruktion der JSON-Anfrage

Ein JSONObject wird initialisiert, um die Datenstruktur für die API-Anfrage aufzubauen. JSON (JavaScript Object Notation) ist ein weit verbreitetes Format zum Austausch von Daten zwischen Server und Client und wird von den meisten APIs inklusive ChatGPT für Anfragen und Antworten verwendet.

- **model**: Gibt das spezifische Modell der GPT-3.5-Familie an, das für die Anfrage verwendet werden soll. gpt-3.5-turbo-instruct ist eine spezifische Konfiguration, die darauf ausgerichtet ist, Anweisungen zu folgen und relevante Antworten zu generieren.
- **prompt**: Dies ist die tatsächliche Frage oder Aufforderung, die an die API gesendet wird. Der Inhalt dieser Variable wird basierend auf Benutzereingaben dynamisch erstellt und soll die API anleiten, spezifische Informationen oder Vorschläge zu generieren.
- **max-token**: Begrenzt die Länge der Antwort. In diesem Fall erlaubt 4000 Tokens eine ausführliche Antwort, was für detaillierte Reisevorschläge nützlich sein kann.
- **temperature**: Steuert die Vorhersehbarkeit der Antworten. Ein Wert von 0 sorgt für konsistente und zuverlässige Antworten, indem er die Wahrscheinlichkeit von zufälligen Antworten minimiert.

## 2- Erstellung der POST-Anfrage

```
RequestBody requestBody = RequestBody.create(jsonBody.toString(), JSON);
Request request = new Request.Builder()
    .url(API.API_URL + "completions")
    .header(name: "Authorization", value: "Bearer " + API.API)
    .post(requestBody)
    .build();
```

Abbildung 8: Erstellung der POST-Anfrage

- **RequestBody erstellen**: Das zuvor erstellte JSON-Objekt wird in einen String umgewandelt und als Body einer POST-Anfrage verwendet. RequestBody.create() wird genutzt, um diesen String in ein Format umzuwandeln, das OkHttp für den Versand nutzen kann.
- **Request aufbauen**: Ein Request-Objekt wird mit dem URL-Endpunkt der API, dem erforderlichen Authentifizierungstoken (Bearer Token), und dem Body der Anfrage konfiguriert. Die Verwendung eines Bearer Tokens ist eine gängige Methode für die Authentifizierung bei APIs, die sichert, dass Anfragen von autorisierten Nutzern stammen.
- **POST-Methode**: Dieser HTTP-Verb wird verwendet, um Daten an den Server zu senden, typisch für das Erstellen oder Aktualisieren von Inhalten.

### 3- Asynchroner Aufruf der API

```

client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(@NotNull Call call, @NotNull IOException e) {
        runOnUiThread(() -> progressBar.setVisibility(View.INVISIBLE));
    }

    @Override
    public void onResponse(@NotNull Call call, @NotNull Response response) throws IOException {
        if (response.isSuccessful()) {
            runOnUiThread(() -> {
                floatingActionButton.setVisibility(View.VISIBLE);
                destinationEditText.setVisibility(View.GONE);
                datePicker.setVisibility(View.GONE);
                purposeEditText.setVisibility(View.GONE);
                sendBtn.setText("try again");
                progressBar.setVisibility(View.INVISIBLE);
            });
        }
    }
});

```

Abbildung 9: Asynchroner Aufruf der CHAT GPT API

- **Asynchroner Aufruf:** client.newCall(request).enqueue() initiiert einen Netzwerkauftrag auf einem Hintergrund-Thread. Dies ist entscheidend, um die Benutzeroberfläche (UI) der App reaktionsfähig zu halten, da langwierige Netzwerkanfragen die Haupt-UI-Thread nicht blockieren.
- **Callback Handling:** Die Callback-Schnittstelle ermöglicht es, auf die Antwort der API zu reagieren oder Fehler zu behandeln, ohne die App zu blockieren.
- **onFailure-Methode:** Diese Methode wird aufgerufen, wenn ein Fehler bei der Netzwerkanfrage auftritt. Der Code in onFailure verbirgt die Fortschrittsanzeige (progressBar), um dem Benutzer zu signalisieren, dass der Ladevorgang abgeschlossen ist, auch wenn dabei ein Fehler aufgetreten ist. Diese Rückmeldung ist wichtig, damit der Benutzer weiß, dass etwas schiefgelaufen ist und möglicherweise weitere Schritte erforderlich sind.
- **onResponse-Methode:** Wenn die Anfrage erfolgreich ist, also eine Antwort vom Server erfolgreich empfangen wurde, führt die onResponse-Methode mehrere Aktionen aus:  
Die Schaltfläche für weitere Aktionen (floatingActionButton) wird sichtbar, was dem Benutzer zusätzliche Interaktionsmöglichkeiten bietet. Gleichzeitig werden Eingabeelemente wie das destinationEditText, der datePicker und das purposeEditText ausgeblendet, da keine weiteren Eingaben mehr benötigt werden. Der Text des Senden-Buttons wird zu "try again" geändert, was dem Benutzer die Option gibt, die Eingabe bei Bedarf zu wiederholen. Abschließend wird die Fortschrittsanzeige unsichtbar gemacht, um zu signalisieren, dass der Prozess beendet ist.

## - Implementierung des Google Maps API:

### 1- Initialisierung und Setup der Kartenansicht:

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_map, container, attachToRoot: false);

    mapView = view.findViewById(R.id.mapView);
    mapView.onCreate(savedInstanceState);
    mapView.getMapAsync(callback: this);
    Places.initialize(requireContext(), apiKey |"YOUR_GOOGLE_PLACES_API_KEY");
    placesClient = Places.createClient(requireContext());
    SupportMapFragment mapFragment = (SupportMapFragment) getChildFragmentManager().findFragmentById(R.id.mapView);
    if (mapFragment != null) {
        mapFragment.getMapAsync(callback: this);
    }

    return view;
}
```

Abbildung 10: onCreateView-Methode im MapFragment

Die onCreateView-Methode in der MapFragment kümmert sich um die Initialisierung und das Setup der Kartenansicht sowie um die Einrichtung der Google Places API. Dies umfasst das Laden der Karte, das Bereitstellen eines API-Clients für Ortsabfragen und das Einrichten notwendiger Callbacks, um auf Karteneignisse zu reagieren.

- **LayoutInflater:** Dieses Objekt wird verwendet, um die XML-Layoutdatei in tatsächliche View-Objekte umzuwandeln.
- **MapView:** Dies ist eine spezielle View für Karten. Durch Aufruf von onCreate wird es initialisiert, was notwendig ist, um den Lebenszyklus der View korrekt zu verwalten. getMapAsync(this) registriert das Fragment als Callback, sodass es benachrichtigt wird, wenn die Google Map bereit ist.
- **Places API:** Die Initialisierung von Places erfolgt durch Places.initialize, was essentiell ist, um den API-Key festzulegen und die Nutzung der Places-Dienste zu ermöglichen. createClient wird genutzt, um einen PlacesClient zu erstellen, der dann verwendet wird, um Anfragen an die Places API zu stellen.
- **SupportMapFragment:** Dies ist eine spezielle Art von Fragment, die speziell für die Anzeige einer Google Map konzipiert ist. In diesem Fall wird geprüft, ob bereits ein MapFragment im Fragmentmanager des Containers vorhanden ist. Dies könnte der Fall sein, wenn das Fragment bereits früher einmal initialisiert wurde und dann durch eine Fragment-Transaktion wiederhergestellt wird.

## 2- die geografische Daten verarbeiten:

```

private void getAddressFromLatLng(LatLng latLng) {
    Geocoder geocoder = new Geocoder(requireContext(), Locale.getDefault());
    try {
        List<Address> addresses = geocoder.getFromLocation(latLng.latitude, latLng.longitude, maxResults: 1);
        if (addresses != null && addresses.size() > 0) {
            Address address = addresses.get(0);
            StringBuilder locationNameBuilder = new StringBuilder();
            if (address.getCountryName() != null) {
                if (locationNameBuilder.length() > 0) {
                    locationNameBuilder.append(", ");
                }
                locationNameBuilder.append(address.getCountryName());
            }
            if (address.getLocality() != null) {
                if (locationNameBuilder.length() > 0) {
                    locationNameBuilder.append(", ");
                }
                locationNameBuilder.append(address.getLocality());
            }
            String countryName = locationNameBuilder.toString();
            findNearbyPlaces(latLng);
            savePlaceDetails(countryName, latLng.latitude, latLng.longitude);
            returnToButtonFragment(countryName, latLng.latitude, latLng.longitude);
            Toast.makeText(requireContext(), text: "Selected Country: " + countryName, Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(requireContext(), text: "Error: Unable to get location name", Toast.LENGTH_SHORT).show();
        }
    }
}

```

**Abbildung 11:** Funktion getAddressFromLatLng

Diese Methode ist besonders nützlich in Anwendungen, die geografische Daten verarbeiten, wie etwa in Reise-Apps, Lieferdiensten oder sozialen Netzwerken, die Standortangaben benötigen. Durch das Anzeigen des Landesnamens kann die App dem Benutzer einen schnellen Überblick geben, wo sich der ausgewählte Punkt befindet, was die Benutzererfahrung verbessert und hilfreich sein kann, um den Kontext oder die Gültigkeit eines ausgewählten Standorts zu bestätigen.

- **Geocoder Initialisierung:** Ein Geocoder-Objekt wird erstellt, um Adresseinformationen basierend auf Längen- und Breitengraden zu erhalten. Es nutzt den Kontext der Anwendung und die Standard-Locale, um lokalisierte Ergebnisse zu liefern.
- **Adressabfrage:** Die Methode getFromLocation des Geocoder-Objekts wird aufgerufen, um eine Liste von Adressen zu erhalten, die der angegebenen geografischen Position entsprechen. Es wird nur das erste Ergebnis angefordert.
- **Adressdaten Verarbeitung:** Die erste Adresse aus der Liste wird untersucht. Der Ländername (getCountryName) und die Ortschaft (getLocality) werden in einen StringBuilder eingefügt, um eine zusammengesetzte Beschreibung der Position zu erstellen.

### 5.2.5 Implementierung des Voicetranslation Fragment:

Das TraductVoiceFragment ermöglicht es Benutzern, Audio aufzunehmen und diese Aufnahmen in Text umzuwandeln, der dann in eine gewählte Sprache übersetzt wird. Durch die Verwendung von MediaRecorder für die Audioaufnahme und OkHttpClient für Netzwerkanfragen zu Übersetzungsservices, bietet das Fragment eine nahtlose Integration von Spracheingabe- und Übersetzungsfunktionalitäten.

#### 1- Aufnahme starten und stoppen:

```
private void startRecording() {
    try {
        mediaRecorder.prepare();
        mediaRecorder.start();
        Toast.makeText(getApplicationContext(), "Recording started", Toast.LENGTH_SHORT).show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

1 usage

private void stopRecording() {
    try {
        mediaRecorder.stop();
        new TranscriptionTask().execute();
    } catch (RuntimeException stopException) {
        // Handle the exception if needed
    } finally {
        mediaRecorder.reset();
        mediaRecorder.release();
        mediaRecorder = null;
        Toast.makeText(getApplicationContext(), "Recording stopped", Toast.LENGTH_SHORT).show();
        mediaRecorder = new MediaRecorder();
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AAC);
        mediaRecorder.setOutputFile(outputFile);
    }
}
```

Abbildung 12: Methode stopRecording/startRecording

- **mediaRecorder.start():** Dies startet die tatsächliche Aufnahme. Ab diesem Moment beginnt das Gerät, Audio vom definierten Audioeingang (z.B. Mikrofon) zu erfassen und in die konfigurierte Datei zu schreiben.
- **mediaRecorder.stop():** Diese Methode stoppt die Aufnahme, die durch start() begonnen wurde. Nach dem Aufruf dieser Methode werden keine weiteren Audio-Daten mehr geschrieben.
- **Nachbearbeitung:** Nach dem Stoppen der Aufnahme wird ein TranscriptionTask gestartet, der das aufgenommene Audio transkribiert. Dies geschieht asynchron, um die Benutzeroberfläche nicht zu blockieren.

## 2- Audiodatei transkribieren:

```

private String transcribeAudio(String apiKey, String filePath, String model) {
    OkHttpClient client = new OkHttpClient();
    RequestBody requestBody = new MultipartBody.Builder()
        .setType(MultipartBody.FORM)
        .addFormDataPart("file", filename: "openai.mp4", RequestBody.create(MediaType.parse("audio/mp4")))
        .addFormDataPart("model", model)
        .build();
    Request request = new Request.Builder()
        .url(API.API_URL + "audio/translations")
        .header("Authorization", value: "Bearer " + apiKey)
        .post(requestBody)
        .build();
    try (Response response = client.newCall(request).execute()) {
        if (response.isSuccessful()) {
            JSONObject jsonObject = null;
            String result = "";
            try {
                jsonObject = new JSONObject(response.body().string());
                result = jsonObject.getString("text");
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
            return result;
        } else {
            Log.e(tag: "Error: ", msg: response.code() + " " + response.message());
        }
    } catch (IOException e) {
        Log.e(tag: "Error executing request", e.getMessage());
    }
}

```

**Abbildung 13:** Methode transcribeAudio

- **OkHttpClient:** Diese Klasse wird verwendet, um Netzwerkanfragen zu stellen. Sie ist bekannt für ihre Effizienz und Fähigkeit, große Datenmengen zu handhaben, was sie ideal für den Datei-Upload und API-Interaktionen macht.
- **MultipartBody:** Das ist ein spezieller Body-Typ, der für das Versenden von Dateien und Daten in mehreren Teilen (multipart/form-data) verwendet wird.
- **Request Builder:** Erstellt einen HTTP POST-Request mit der URL für die Transkriptions-API, fügt den vorbereiteten RequestBody und den erforderlichen Authentifizierungskopf hinzu.
- **header:** Fügt den Authentifizierungstoken hinzu, der notwendig ist, um die API zu nutzen.
- **Response Handling:** Nach dem Absenden des Requests wird die Antwort überprüft. Bei einem erfolgreichen Response wird die Antwort ausgelesen und der Text zurückgegeben.
- **JSONObject:** Parsen des JSON-Antwortkörpers, um den transkribierten Text zu extrahieren.

### 5.2.6 Implementierung des Summary Fragment

- **locationtracking service:** Der LocationTrackingService ist ein IntentService, der für die Verfolgung und Speicherung von Ortsdaten und der Interaktion mit der Google Places API konzipiert ist.

#### 1-Initialisierung und Setup:

```
public LocationTrackingService() { super( name: "LocationTrackingService" ); }

no usages

public static synchronized LocationTrackingService getInstance() {
    if (instance == null) {
        instance = new LocationTrackingService();
    }
    return instance;
}

@Override
public void onCreate() {
    super.onCreate();
    // Initialize Places API with your API key
    Places.initialize(getApplicationContext(), apiKey: "AIzaSyBA9JG7t21GrUWOy-rnmjCenGhCzJWicSk");

    placesClient = Places.createClient( context: this );
    fusedLocationClient = LocationServices.getFusedLocationProviderClient( context: this );

    startLocationUpdates();
}
```

**Abbildung 14:** Locationservice Initialisierung und Setup

- **Konstruktor:** Der Konstruktor von LocationTrackingService ruft den Konstruktor der übergeordneten Klasse IntentService auf und gibt ihm den Namen des Services. Dies ist wichtig, da Android Services parallel ausgeführt werden und der Name zur Identifikation im System verwendet wird.

- **Singleton-Muster:** getInstance implementiert das Singleton-Muster. Dies stellt sicher, dass zu jeder Zeit höchstens eine Instanz des LocationTrackingService existiert, was hilfreich ist, um Ressourcenkonflikte zu vermeiden und den Zustand des Services konsistent zu halten. Die Verwendung eines synchronisierten Zugriffsmechanismus gewährleistet, dass die Erstellung der Instanz threadsicher erfolgt.

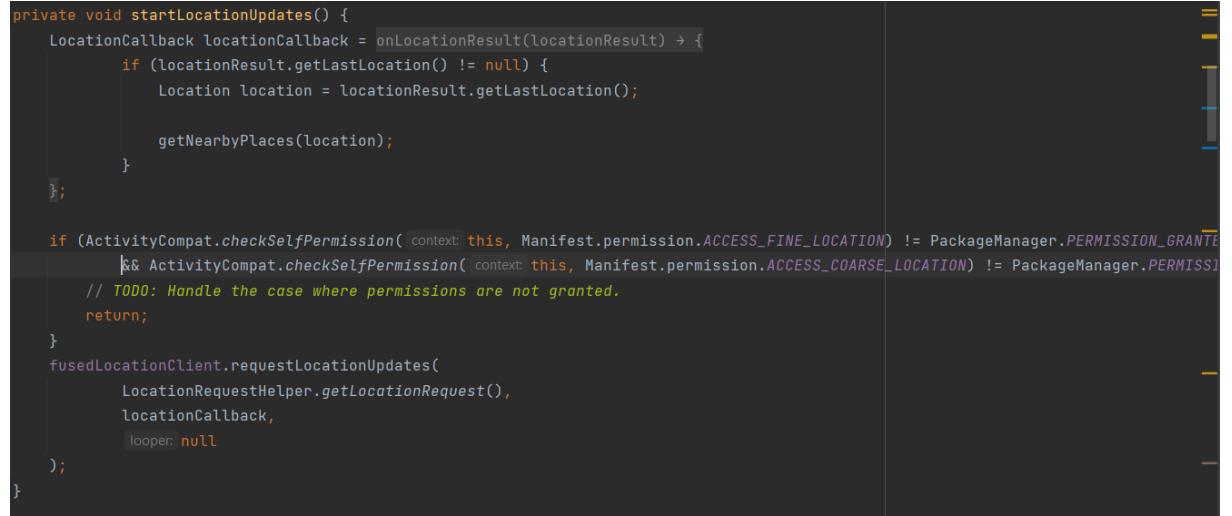
- **Places API Initialisierung:** Im onCreate-Methodenaufruf wird die Google Places API mit einem spezifischen API-Schlüssel initialisiert. Diese Initialisierung ist notwendig, um auf die Places-Dienste zugreifen zu können, die Standortbezogene Daten und Funktionalitäten bereitstellen.

- **Places Client:** Wird erstellt, um Anfragen an die Google Places API zu senden. Dieser Client wird verwendet, um Orte in der Nähe und andere relevante Ortsinformationen zu erfragen.

- **Fused Location Provider Client:** Ist ein Bestandteil der Google Play Services und bietet eine Schnittstelle für Standortdienste. Der Client wird verwendet, um Standortaktualisierungen zu erhalten, was essentiell für standortbezogene Funktionen innerhalb des Services ist.

- **Start der Standortaktualisierungen:** Die Methode startLocationUpdates wird aufgerufen, um regelmäßige Standortaktualisierungen anzufordern. Dies ist kritisch, um die aktuellen Standortinformationen kontinuierlich zu erhalten und zu verarbeiten.

## 2-Standortupdates verwalten:



```
private void startLocationUpdates() {
    LocationCallback locationCallback = onLocationResult(locationResult) -> {
        if (locationResult.getLastLocation() != null) {
            Location location = locationResult.getLastLocation();

            getNearbyPlaces(location);
        }
    };

    if (ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
        & ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
        // TODO: Handle the case where permissions are not granted.
        return;
    }
    fusedLocationClient.requestLocationUpdates(
        LocationRequestHelper.getLocationRequest(),
        locationCallback,
        looper: null
    );
}
```

Abbildung 15: startLocationUpdates Methode

- **Location Callback:** Diese innere Klasse wird definiert, um auf Standortupdates zu reagieren. Bei jedem Update wird überprüft, ob eine gültige letzte Position vorhanden ist, und falls ja, wird die Methode getNearbyPlaces aufgerufen.
- **Berechtigungsprüfung:** Es wird sichergestellt, dass die notwendigen Berechtigungen vorhanden sind, bevor Standortupdates angefordert werden. Dies ist wichtig, um die Privatsphäre der Nutzer zu respektieren und rechtliche Anforderungen zu erfüllen.
- **Anforderung von Standortupdates:** Der fusedLocationClient wird verwendet, um regelmäßige Updates des Standorts zu beantragen, was die Grundlage für die Ortsbasierten Dienste des Services bildet.

### 3- Festlegung der interessanten Orte:

```
private boolean isPlaceOfInterest(Place place) {  
    List<Place.Type> interestingPlaceTypes = Arrays.asList(  
        Place.Type.RESTAURANT,  
        Place.Type.ART_GALLERY,  
        Place.Type.AMUSEMENT_PARK,  
        Place.Type.LODGING, // Hotel  
        Place.Type.MUSEUM,  
        Place.Type.HINDU_TEMPLE,  
        Place.Type.AMUSEMENT_PARK  
    );  
    for (Place.Type placeType : place.getTypes()) {  
        if (interestingPlaceTypes.contains(placeType)) {  
            return true;  
        }  
    }  
    return false;  
}
```

Abbildung 16: Methode isPlaceOfInterest

- Die Google Maps API bietet eine umfangreiche Liste von Ortskategorien (Place.Type), die es Entwicklern ermöglichen, spezifische Arten von Standorten zu identifizieren.
- In der App wird die Funktion isPlaceOfInterest verwendet, um aus der Vielzahl möglicher Orte diejenigen herauszufiltern, die für Urlauber während ihrer Reise besonders relevant sind. .
- In der App werden ausschließlich die Orte erfasst und verarbeitet, die durch die Funktion isPlaceOfInterest als relevant definiert wurden. Diese Auswahl stützt sich auf bestimmte Kategorien der Google Maps API, darunter Restaurants, Kunstgalerien, Vergnügungsparks, Unterkunftsmöglichkeiten, Museen und religiöse Einrichtungen.

- **Image labeling service:** Der ImageLabelingService ist darauf spezialisiert Bilder in der Galerie des Benutzers zu überwachen und sie basierend auf ihrem Standort zu kategorisieren.

### 1-ServiceConnection zum Binden an LocationTrackingService::

```
private ServiceConnection serviceConnection = new ServiceConnection() {  
  
    1 usage  
    @Override  
    public void onServiceConnected(ComponentName className, IBinder service)  
        no usages  
        LocationTrackingService.LocalBinder binder = (LocationTrackingService.LocalBinder) service;  
        locationTrackingService = binder.getService();  
        isBound = true;};
```

Abbildung 17: Methode onServiceConnected

- **onServiceConnected:** Diese Methode wird aufgerufen, wenn eine Verbindung zum Service erfolgreich hergestellt wurde. Innerhalb dieser Methode wird der IBinder, der vom gebundenen Service zurückgegeben wird (hier als service bezeichnet), verwendet, um eine Instanz des LocationTrackingService zu erhalten. .

- **locationTrackingService.addLocationUpdateListener:** Registriert einen Listener im LocationTrackingService, der benachrichtigt wird, wenn es eine Standortaktualisierung gibt.

```
public void onServiceDisconnected(ComponentName arg0) {  
    isBound = false;  
}  
};
```

Abbildung 18: Methode onServiceDisconnected

- **onServiceDisconnected:** Diese Methode wird aufgerufen, wenn die Verbindung zum Service unerwartet getrennt wurde (z.B. wenn der Service abgestürzt ist oder beendet wurde). Die isBound-Flagge wird auf false gesetzt, um anzudeuten, dass der Service nicht mehr gebunden ist.

## 2-Ermittlung des letzten hinzugefügten Bildes mit Standort:

```

private void getLastAddedImageWithLocation() {
    ContentResolver contentResolver = getContentResolver();
    String[] projection = {
        MediaStore.Files.FileColumns._ID,
        MediaStore.Images.Media.DATE_TAKEN
    };
    Cursor cursor = contentResolver.query(
        MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
        projection,
        null,
        null,
        MediaStore.Images.Media.DATE_ADDED + " DESC");
}

if (cursor != null && cursor.moveToFirst()) {
    int idColumnIndex = cursor.getColumnIndex(MediaStore.Files.FileColumns._ID);
    long imageId = cursor.getLong(idColumnIndex);
    cursor.close();
    Uri imageUri = Uri.withAppendedPath(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, Long.toString(imageId));
    if (!isImageProcessed(imageUri.toString())) {
        new GetLocationNameAsyncTask().execute(imageUri);
    }
}

```

**Abbildung 19:** Methode getLastAddedImageWithLocation

- **getContentResolver().query():** erstellt eine Abfrage, die auf die URI der externen Inhalte der MediaStore.Images zugreift. Dies gibt uns Zugriff auf die Mediendaten (Bilder), die auf dem Gerät gespeichert sind.
- **MediaStore.Images.Media:** Dies ist der Sortierparameter, der sicherstellt, dass die Ergebnisse nach dem Hinzufügedatum absteigend sortiert werden. Das heißt, das zuletzt hinzugefügte Bild wird als erstes zurückgegeben.
- Die if-Bedingung überprüft, ob der Cursor nicht null ist und ob er zum ersten Datensatz bewegt werden kann, was bedeutet, dass es mindestens ein Bild in der Galerie gibt.
- Dann wird die Uri des Bildes mit Uri.withAppendedPath() konstruiert, indem die Basis-URI der MediaStore-Images und die ID des zuletzt hinzugefügten Bildes zusammengefügt werden.
- **isImageProcessed():** überprüft, ob das Bild bereits verarbeitet wurde, um doppelte Verarbeitungen zu vermeiden.
- Wenn das Bild noch nicht verarbeitet wurde, wird ein neuer AsyncTask (hier GetLocationNameAsyncTask) gestartet, der die URI des Bildes übernimmt, um die Standortinformationen im Hintergrund abzurufen.

### 3- Standortermittlung aus einem Bild:

```

private String getLocationNameFromImage(Uri imageUri) {
    InputStream stream = getContentResolver().openInputStream(imageUri);
    if (stream == null) {
        return null;
    }
    ExifInterface exifInterface = new ExifInterface(stream);
    double[] latLong = exifInterface.getLatLong();
    if (latLong != null) {
        return getLocationNameFromCoordinates(latLong[0], latLong[1]);
    } else {
        long startTime = System.currentTimeMillis();
        if(isBound)
        {
            if (lastKnownLocation != null) {
                return getLocationNameFromCoordinates(lastKnownLocation.getLatitude(), lastKnownLocation.getLongitude());
            }
            else {

                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    catch (IOException e) {
        e.printStackTrace();
        return null;
    }
}

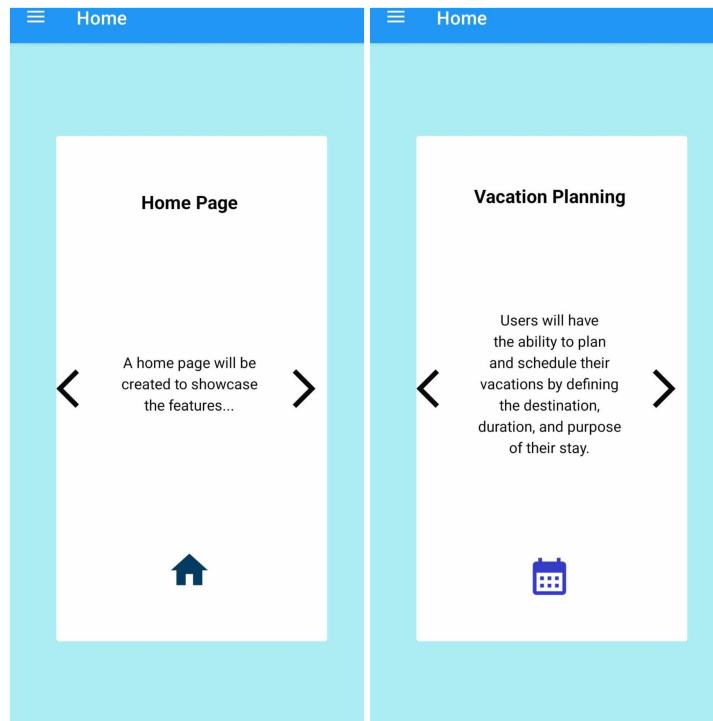
```

Abbildung 20: Methode getLocationNameFromImage

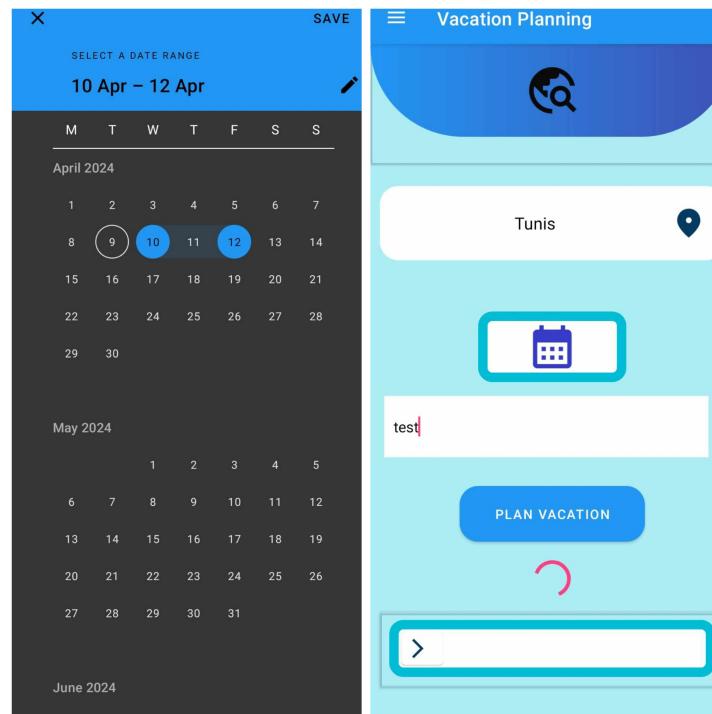
- **InputStream und ContentResolver:** Diese Komponenten werden verwendet, um auf Daten einer Bilddatei zuzugreifen, die durch einen URI lokalisiert ist. Der InputStream ermöglicht das Lesen der Bilddaten, während der ContentResolver hilft, auf Inhalte zuzugreifen, die durch andere Apps bereitgestellt werden.
- **ExifInterface:** Diese Klasse wird genutzt, um auf die EXIF-Metadaten eines Bildes zuzugreifen. EXIF-Metadaten enthalten Details wie Kamerainformationen, Bearbeitungsinformationen und geografische Koordinaten des Ortes, an dem das Foto aufgenommen wurde.
- **LatLong Koordinaten Extraktion:** Aus den EXIF-Daten des Bildes werden geografische Breiten- und Längengradinformationen extrahiert, falls verfügbar. Diese Informationen sind nützlich, um den tatsächlichen Standort zu identifizieren.
- **Standortnamenbestimmung aus Koordinaten:** Sobald die Koordinaten verfügbar sind, wird eine weitere Methode (getLocationNameFromCoordinates) aufgerufen, um den eigentlichen Namen des Standortes aus diesen Koordinaten zu ermitteln.
- **Fallback auf letzte bekannte Position:** Wenn keine Koordinaten aus den EXIF-Daten erhältlich sind, versucht die Methode, die letzte bekannte Position des Gerätes zu verwenden. Falls diese verfügbar ist, wird ebenfalls versucht, daraus den Standortnamen zu ermitteln.

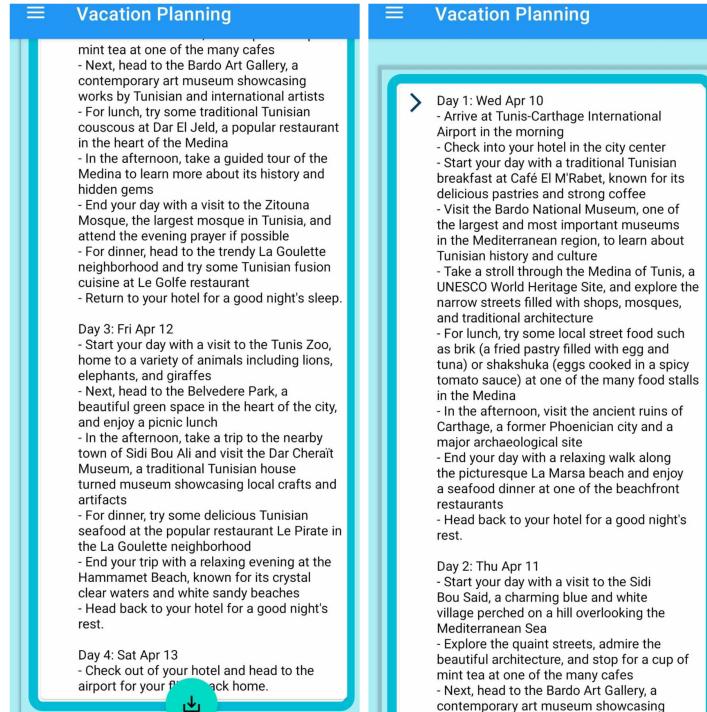
### 5.3 Benutzeroberfläche

- Homepage:

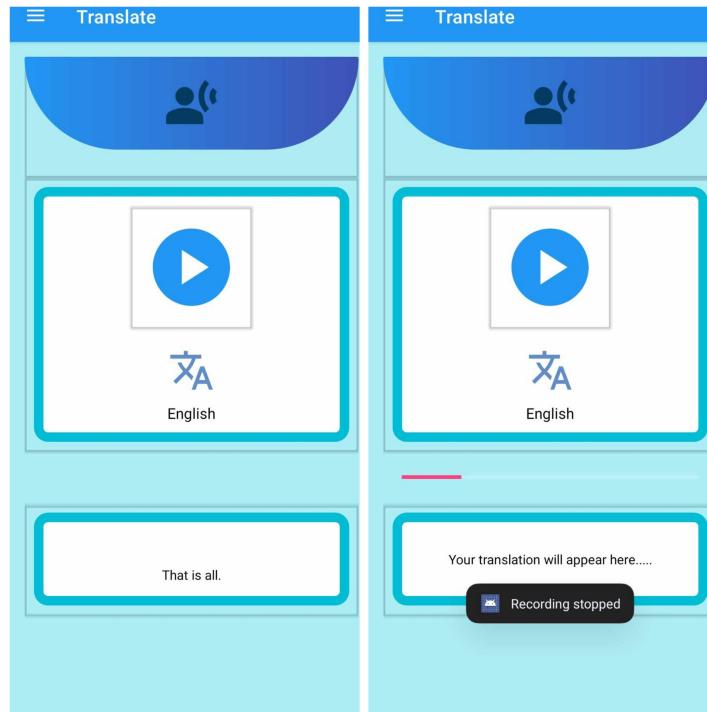


- Vacation planning:

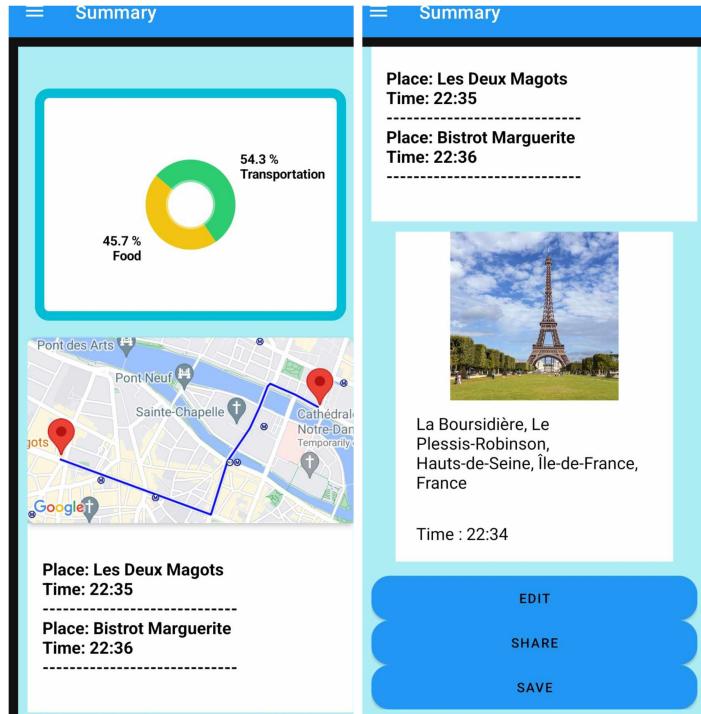




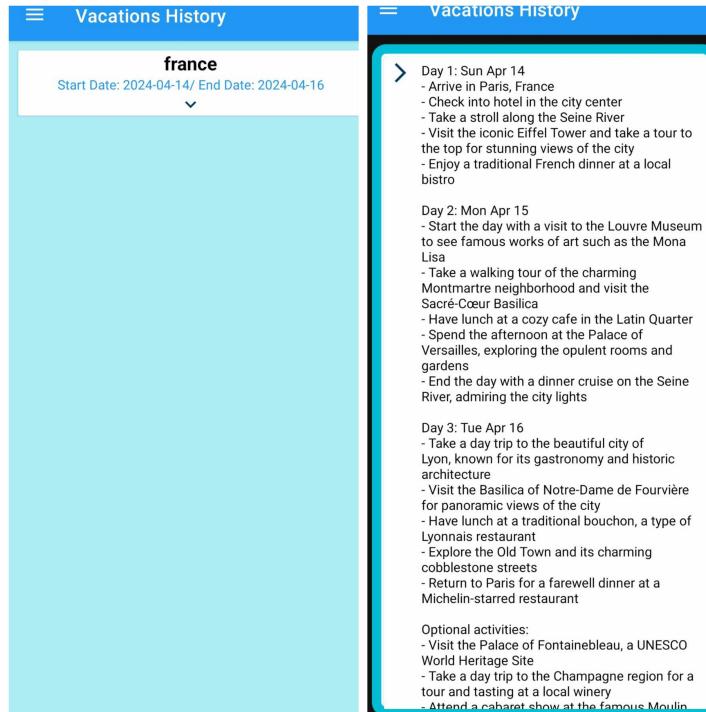
- Voice translation:



- Summary:



- Vacations History:



## 6 Ergebnisse

Testfall-ID	Ziel	Voraussetzung	Schritte	Erwartetes Ergebnis	Ergebnis
TC_UI_1	Überprüfung der Funktionalität zur Eingabe und Auswahl eines Urlaubsorts.	Die App ist geöffnet und bereit für die Interaktion.	1- Der Benutzer navigiert zum Urlaubsplanungs-Feature. 2- Der Benutzer gibt einen Zielort schriftlich ein oder wählt ihn auf der Karte aus.	Der ausgewählte Ort wird korrekt angezeigt und für den Planungsvorgang übernommen.	Der Benutzer konnte den Ort erfolgreich eingeben und auswählen.
TC_UI_2	Sicherstellen, dass die App die Auswahl von Start- und Enddatum für die Urlaubsplanung ermöglicht.	Der Benutzer hat bereits einen Zielort ausgewählt.	Der Benutzer gibt das Start- und Enddatum der Reise ein.	Die App akzeptiert und speichert den gewählten Zeitraum.	Der Zeitraum wurde korrekt eingetragen und für die Reiseplanung verwendet.
TC_UI_3	Prüfung der Generierung von Urlaubsvorschlägen.	Ort und Zeitraum sind spezifiziert.	Die App verarbeitet die Eingaben und generiert Vorschläge.	Die App bietet eine Liste von Aktivitäten und Orten, die auf den eingegebenen Daten basieren.	Der Benutzer erhält relevante Vorschläge für den Urlaubsort.
TC_UI_4	Überprüfen der Funktion zur Eingabe und Kategorisierung von Ausgaben.	Die App ist im Bereich der Ausgabenverfolgung	Der Benutzer gibt Ausgaben mit Datum, Kategorie und Betrag ein.	Die Ausgaben werden korrekt erfasst und in einer Übersicht angezeigt.	Die Ausgaben wurden erfolgreich hinzugefügt und korrekt kategorisiert.
TC_UI_5	Sicherstellung der korrekten Funktionsweise der Voice Translation.	Der Benutzer ist bereit, die Sprachübersetzung zu nutzen.	1-Der Benutzer wählt die Zielsprache aus. 2-Der Benutzer spricht in das Mikrofon. 3-Die App übersetzt die gesprochenen Wörter in die gewählte Sprache.	Die Übersetzung ist akkurat.	Die Spracheingabe wurde korrekt übersetzt und dem Benutzer angezeigt.
TC_UI_6	Test der automatischen Erstellung einer täglichen Zusammenfassung des Urlaubs.	Der Benutzer muss der App Zugriff auf Fotos und Standortdaten gewähren.	Der Benutzer überprüft die generierten Zusammenfassungen für jeden Tag.	Die App stellt eine vollständige Zusammenfassung der Orte, Routen, Ausgaben und Fotos des Tages bereit.	Die Zusammenfassung wird täglich erstellt, jedoch treten bei einigen Android-Versionen Fehler bei der Darstellung von Fotos und Routen auf.

Abbildung 21: Übersicht der Testfälle

## 7 Zusammenfassung

### 7.1 Zusammenfassung Der Arbeit

Die vorliegende Arbeit beschreibt die Entwicklung und Implementierung unsere Urlaub-tracking App, die für die Überwachung von Urlaubsreisen verwendet wird. Diese App zielt darauf ab, Reisenden eine einfache und benutzerfreundliche Plattform zu bieten, um ihre Reisen zu planen, zu überwachen und zu reflektieren. „TripSquirrel“ wurde entwickelt, um der wachsenden Nachfrage nach personalisierten Reiseerlebnissen gerecht zu werden und es den Benutzern ermöglicht, ihre Reiseaktivitäten effektiv zu dokumentieren und zu verwalten.

„TripSquirrel“ ist eine fortschrittliche Anwendung, die Benutzern ein umfassendes Reiseerlebnis bietet. Sie kombiniert fortschrittliche Funktionen, um Reisenden dabei zu helfen, ihre Abenteuer zu planen und zu erleben. Die Personalisierung von Urlaubsvorschlägen ist eine der Hauptfunktionen der App. Maßgeschneiderte Reisepläne werden basierend auf den individuellen Präferenzen der Benutzer erstellt. Dadurch erhalten Benutzer Ideen, die ihren Vorlieben entsprechen. Ein weiteres Merkmal ist die Sprachübersetzung in Echtzeit. Ohne Sprachbarrieren können Reisende sich nahtlos mit Einheimischen verstständigen. Fortschrittliche Technologien ermöglichen es der Anwendung, in den vier internationalen Sprachen zu sprechen. TripSquirrel kann auch automatische Reisezusammenfassungen erstellen. Geografische Daten, fotografische Einträge und persönliche Erfahrungen werden in diesen Berichten kombiniert. Dadurch entsteht ein vollständiges Bild der Reise, das den Benutzern hilft, ihre Erfahrungen zu dokumentieren und zu teilen. Eine weitere nützliche Funktion ist die visuelle Darstellung der Reiseausgaben. Benutzer können mithilfe von Kuchendiagrammen leicht sehen, wie viel sie für Unterkunft, Verpflegung, Aktivitäten und andere Ausgabenkategorien ausgegeben haben. Dies verbessert die Finanztransparenz und ermöglicht eine bessere Budgetplanung. Für die Erstellung kontextbezogener Inhalte und Vorschläge verwendet die Anwendung APIs wie OpenAIs GPT-3 und Google Maps für geografische Dienste.

Die Testphase der „TripSquirrel“-App zeigte, dass die Urlaubsvorschläge umfangreich sind und die Anzeige der Ausgaben präzise waren. Die Benutzer bewunderten insbesondere die einfache Handhabung der Sprachübersetzung und den Mehrwert der täglichen Zusammenfassungen. Jedoch wurden aufgrund der hohen Sicherheitsanforderungen auch technische Probleme festgestellt, insbesondere bei der Speicherung von Fotos auf älteren Android-Versionen und bei Geräten der Marke Xiaomi. Diese Probleme traten insbesondere auf älteren Betriebssystemen und bestimmten Sicherheitseinstellungen auf. Weitere Tests im Android Studio Emulator zeigten Probleme bei der Ortssuche durch Google Maps. Diese Probleme könnten Netzwerkprobleme oder Emulatoreinschränkungen sein. Um eine breitere Kompatibilität der App zu gewährleisten, benötigen diese Punkte zusätzliche Untersuchungen.

## 7.2 Ausblick und zukünftige Entwicklungen

Es gibt zahlreiche Möglichkeiten, die Funktionalität der App zu erweitern und zu verbessern, angesichts der ständig steigenden Erwartungen moderner Reisender und der rasanten technologischen Entwicklungen. Diese Verbesserungen zielen darauf ab, die Benutzererfahrung zu verbessern, die Informationen genauer zu machen und die App einem größeren Publikum zugänglicher zu machen.

Die Integration von AR-Technologien ist eine faszinierende Möglichkeit, die App zu erweitern. AR ermöglicht es Nutzern, virtuelle Reiseziele zu planen, bevor sie sie tatsächlich besuchen. Die Kamera des Smartphones ermöglicht es, Hotels, Sehenswürdigkeiten und Restaurants in Echtzeit zu visualisieren, was insbesondere in der Planungsphase von Reisen eine wertvolle Hilfe darstellt. AR-Technologie kann auch verwendet werden, um interaktive Wegbeschreibungen und Navigationshilfen zu erstellen, was es erheblich erleichtert, sich in unbekannten Städten zurechtzufinden.

Durch die Verwendung von Machine Learning (ML) Technologien ist es möglich, die Reisevorschläge von „TripSquirrel“ erheblich zu personalisieren. Die App kann Aktivitäten und Ziele basierend auf Bewertungen, vergangenen Reisen und persönlichen Präferenzen vorschlagen. Darüber hinaus könnte ML die Genauigkeit der Reisekostenvorhersage basierend auf dynamischen Variablen wie Nutzerverhalten, Saison und Währungsschwankungen verbessern.

„TripSquirrel“ könnte durch die Erweiterung der Kompatibilität auf zusätzliche Betriebssysteme wie iOS und Windows Mobile einem noch größeren Publikum zugänglich gemacht werden. Darüber hinaus wäre die Schaffung einer Webversion der Anwendung eine bedeutende Ergänzung, um Benutzern den Zugriff auf ihre Daten auf nicht-mobilen Geräten zu ermöglichen. Die nahtlose Übertragung von Reiseplänen von einem Gerät auf ein anderes wäre durch die Plattformübergreifende Synchronisation möglich.

Durch die Einführung einer Community-Plattform, auf der Nutzer ihre Erfahrungen teilen, Reisetipps austauschen und sogar gemeinsame Aktivitäten planen können, könnte ein sozialer Aspekt hinzugefügt werden. Die App könnte durch Funktionen wie Gruppenchats, Foto-Sharing und Bewertungssysteme erweitert werden, was sie zu einem Planungstool und einem sozialen Netzwerk für Reiseliebhaber macht.

Die Fortsetzung von „TripSquirrel“ hat das Ziel, die Art und Weise, wie wir reisen, grundlegend zu verändern. Die App kann weiterhin optimiert werden, um den sich ändernden Bedürfnissen der globalen Reisegemeinschaft gerecht zu werden, indem neue Technologien und Nutzerfeedback einbezogen werden. Zukünftige Erweiterungen werden TripSquirrel als führende digitale Reiseplanungslösung etablieren und gleichzeitig die Funktionalität verbessern.

## Literatur

- [1] Laura Knops. Besser informiert im urlaub: Apps, die das reisen erleichtern. <https://www.merkur.de/reise/ausland-hilfe-geld-packen-planen-uebersetzen-handy-apps-reisen-urlaub-reisende-zr-92639164.html>, November 2023. Zugegriffen: 15. März 2024.
- [2] JULIA HUBERT. Reisetagebuch-apps: Die 3 besten kostenlosen apps im test. <https://www.merkur.de/reise/ausland-hilfe-geld-packen-planen-uebersetzen-handy-apps-reisen-urlaub-reisende-zr-92639164.html>. Zugegriffen: 16.03.2024.
- [3] Online Redaktion. Künstliche intelligenz im tourismus – ein „schlafender riese“? <https://gastgewerbe-magazin.de/kuenstliche-intelligenz-im-tourismus-ein-schlafender-riese-41188>, March 2022. Zugegriffen: 17.03.2024.
- [4] Konstanze Nastarowitz. Wenn chatgpt die reise plant. <https://www.tagesschau.de/wirtschaft/verbraucher/kuenstliche-intelligenz-tourismus-reise-100.html>, 2023. Zugegriffen: 28.02.2024.
- [5] Luisa Ziegler. Künstliche intelligenz: So verändert sie das reisen. <https://www.rnd.de/reise/wie-kuenstliche-intelligenz-das-reisen-veraendert-nachhaltiger-tourismus-dank-ki-IVBJCI4ZIBITJZQJCXD3SBIZE.html>, March 2023. Zugegriffen: 29.02.2024.
- [6] Was ist java? <https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-java-programming-language/>:text=Java Zugegriffen: 15.03.2024.
- [7] Stephan Augsten. Was ist android studio? <https://www.dev-insider.de/was-ist-android-studio-a-605428/>, 2017. Zugegriffen: 15.03.2024.