Introduction

YouSightSeen App is a mobile application for automatic tourist route generation based on user preferences. The application helps tourists quickly register via Google, complete a preference survey, view a map with their current location, and use flexible functionality for building and editing routes through interesting city locations.

Glossary

- **GPS** — Global Positioning System, used to determine user location.
- **Point of Interest (POI)** — an object on the map of interest to tourists (parks, architecture, museums, etc.).
- **Route** — optimal sequence of points of interest suggested by the application.
- **Map** — interactive application layer displaying user location and routes.

Actors

- **User**
    - Role: end user of the application.
    - Goals: register, specify interests, build and edit routes, get descriptions of points of interest, use navigation.
    - Responsibilities: correctly input data, follow application recommendations.

Functional Requirements

Strategic Use Cases

Use Case UC-S-1: Registration and Onboarding

- **Actors:** User, Google Authentication Service, System.
- **Goals:** Quick registration and collection of personal data for personalized service.
- **Preconditions:** Application is installed, user has a Google account.
- **Trigger Condition:** First application launch.
- **Extensions:** UC-S-2 (Route Building).

Main Success Scenario:

1. User opens the application.
2. Application displays Google registration window.
3. User completes authorization through Google Authentication Service.
4. Token is received, user profile is created in the database.
5. Preference survey screen is displayed with questions about:

    - Preferred types of attractions (parks, architecture, museums, historical sites, entertainment venues, natural landmarks, cultural sites)
    - Activity level (walking intensity)
    - Interest in dining establishments during the route

6. User completes the questionnaire and clicks "Save".
7. System saves user preferences to the profile.
8. Yandex MapKit initializes and displays the main map with user's current location.
9. Registration completes successfully.

Alternative Scenarios:

Alternative Scenario "Authentication Failed":

Trigger Condition: Google Authentication Service returns an error

1. System displays message "Authentication error. Please try again" and returns to registration screen.
2. User can retry authentication or close the application.

Alternative Scenario "Incomplete Questionnaire":

Trigger Condition: User attempts to skip questions

1. System displays warning "Please complete the questionnaire to receive personalized recommendations" and highlights unanswered fields.
2. User completes remaining questions.
3. Process continues from step 6.

Alternative Scenario "Internet Connection Lost":

Trigger Condition: Connection is lost during authorization

1. System shows message "No internet connection. Please check your connection and try again" and waits for connection restoration.
2. User can retry when connection is available.

Use Case UC-S-2A: Route Creation

Actors: User, Yandex MapKit, OpenRouteService, Geoapify, System.

Goals: Build an initial route based on user parameters and preferences.

Preconditions:

- User is authorized (UC-1 completed)
- User preferences are filled
- Geolocation is available via GPS

- Internet connection is available for external services

Trigger Condition: "Create Route" button is pressed.

Extensions: UC-S-2B (Route Editing and Filtering)

Main Success Scenario:

1. User views the map (Yandex MapKit) with their current location.
2. User presses "Create Route" button.
3. System opens route parameters window with input fields:
    - Desired walk duration
    - Maximum number of places to visit
    - Need to include dining establishments
    - Optional: preferred starting point (default — current location)
4. User fills parameters and presses "Create Route".
5. System validates entered data.
6. System loads user preferences from profile.
7. System requests Geoapify for list of points of interest in radius corresponding to specified time and attraction types from user preferences.
8. System filters received POIs based on:
    - Type of attractions (matching preferences)
    - Presence of dining establishments (if required)
9. System sends filtered points to OpenRouteService to calculate optimal route:
    - Starting point (current location or user-selected)
    - Destination points (selected POIs)
    - Time constraint
    - Travel mode (pedestrian)
10. OpenRouteService returns optimized route with minimum travel time.
11. System displays route on map (Yandex MapKit):
    - Route line connecting all points
    - Point of interest markers
    - Estimated time at each point
    - Total route time
12. User reviews the route.
13. User can click any point of interest to view details:
    - Name and category

- Attraction description
- Photos (from Geoapify)
- Recommended visit duration

14. Route creation is complete; user can proceed to edit route (UC-S-2B) or accept as is.

Alternative Scenarios:

Alternative Scenario "Insufficient Points of Interest":

Trigger Condition: Geoapify doesn't return sufficient POIs at step 7

1. System displays message "Could not find enough places matching your preferences and parameters".
2. System offers options:
    - Increase walk duration
    - Increase maximum number of places
    - Broaden interest categories
    - Manually select points on map
3. User selects manual point selection.
4. Yandex MapKit displays map with all available points of interest.
5. User manually selects desired locations on map.
6. System sends selected points to OpenRouteService.
7. Process continues from step 10.

Alternative Scenario "Invalid Parameters":

Trigger Condition: Invalid data entered at step 5

1. System validates parameters and detects issues:
    - Duration is too short (less than minimum required)
    - Number of places equals zero or exceeds reasonable limit
    - Conflicting parameters (too many places for available time)
2. System displays validation error message with specific description.
3. System highlights problematic fields in red.
4. User corrects parameters.
5. Process continues from step 5.

Alternative Scenario "GPS Unavailable":

Trigger Condition: Cannot determine user location at step 1

1. System displays message "Geolocation services unavailable".
2. System asks user to:

    - Enable GPS/location services
    - Manually set starting point on map
3. User enables geolocation or selects starting point manually on Yandex MapKit.
4. Process continues from step 2.

Use Case UC-S-2B: Route Editing and Filtering

Actors: User, Yandex MapKit, OpenRouteService, System.

Goals: Customize and refine an existing route using filters and manual adjustments.

Preconditions:

- User has generated an initial route (UC-S-2A completed)
- Route is displayed on map
- Internet connection is available for route recalculation

Trigger Condition: User opens side editing menu or presses "Edit Route" button.

Main Success Scenario:

1. User views the generated route on map.
2. User opens side editing menu.
3. User modifies the route using available options:
    - Removes uninteresting points from list
    - Adds new points (from system suggestions or manually selects on map)
    - Changes time spent at specific locations
    - Changes visit order via drag-and-drop
    - Modifies dining preferences
4. User confirms changes with "Recalculate Route" button.
5. System sends updated point set to OpenRouteService.
6. OpenRouteService recalculates route with new parameters.
7. Yandex MapKit displays updated route on map with:
    - New route line
    - Updated markers
    - Recalculated time estimates
    - Updated total route time
8. User reviews the updated route.
9. User can continue editing (return to step 3) or accept final route.

10. User accepts final route version.
11. System saves route to user profile.
12. User can start navigation along the route.

Alternative Scenarios:

Alternative Scenario "Cannot Build Valid Route After Editing":

Trigger Condition: User modifications create impossible conditions at steps 3-6

1. OpenRouteService cannot build route or system detects:
    - Too many points for specified time
    - Selected points are too far apart
    - No valid path exists between selected points
2. System displays warning message with problem description.
3. System offers options:
    - Return to previous valid route
    - Automatically adjust parameters (increase duration)
    - Continue manual editing
4. User selects option.
5. Process continues based on user choice.

Alternative Scenario "Internet Connection Lost During Editing":

Trigger Condition: Network connection lost at step 5-6

1. System detects network error during recalculation request.
2. System displays message "Connection lost. Changes saved locally. Route will recalculate when connection is restored."
3. System saves user's editing changes locally.
4. User can continue editing offline or wait for connection.
5. When connection is restored, system automatically sends recalculation request.
6. Process continues from step 6.

Non-Functional Requirements

Environment

Server Side:

- Programming Language: Go
- Framework: ECHO
- API: REST API, HTTP/HTTPS protocols

- Database: Relational DB PostgreSQL

Client Side:

- Platform: Android
- Programming Language: Java
- Map SDK: Yandex MapKit SDK
- Minimum Android Version: 8.0 (API Level 26)

External Services:

- Google API — user authorization
- Yandex MapKit — map display, geolocation, route visualization
- OpenRouteService API — optimal route building
- Geoapify API — point of interest (POI) information retrieval

Infrastructure:

- Stable internet connection required for external API operation
- GPS support for user location determination
- Load balancer for request distribution
- Service replication under high load

Performance

- **Interface Response Time:** up to 1 second per user interaction operation.
- **Route Building Time:** maximum 5 seconds (including Geoapify and OpenRouteService requests).
- **Map Loading Time:** no more than 2 seconds with stable internet.
- **Throughput:** system should support up to 1000 simultaneous users without performance degradation.
- **Scalability:** horizontal scaling capability with increasing user numbers.
- **Request Optimization:** caching Geoapify data for frequently requested areas, minimizing repeated OpenRouteService requests.

Reliability

- **System Availability:** uptime of at least 99.5% (allowable downtime — no more than 3.65 hours per month).
- **Failure Handling:**
    - Logging all server errors for subsequent analysis
    - Using load balancer to distribute requests between servers
    - Automatic restart of failed services
    - Backup servers for critical components

- **Failure Recovery:**
    - Automatic connection restoration with external APIs during temporary outages
    - Saving user state for work continuation after failure
    - Database backup every 24 hours
- **Data Protection:**
    - Encryption of user personal data
    - Secure data transmission via HTTPS
    - Compliance with personal data protection requirements (GDPR)
- **External Service Unavailability Handling:**
    - When Geoapify unavailable — use cached data or local POI database
    - When OpenRouteService unavailable — notify user and offer simplified route building
    - When Yandex MapKit unavailable — display error message and inability to continue

Extensibility

- **Scalability:**
    - Horizontal scaling of server side with increased load
    - Database replication for read load distribution
    - Ability to add additional application servers
- **Modularity:**
    - Ability to replace or add new cartographic services besides Yandex MapKit
    - Ability to integrate additional POI sources besides Geoapify
    - Support for alternative routing services when needed
- **Functional Extensions:**
    - Adding new types of points of interest and attraction categories
    - Integration with public transport systems for combined routes
    - Multi-language interface support
    - Adding social functions (route sharing, ratings, reviews)
    - Weather service integration
- **Compatibility:**
    - API versioning for backward compatibility
    - Client application update capability without mandatory reinstallation
    - Compatibility with various Android versions (from 8.0 and higher)