GeoScene

SWE Final Project

Itay Bouganim, Sahar Vaya, Lior Hassan

Itay Bouganim

UI Development, Algorithms Development, AR and Maps Integration

Sahar Vaya

Algorithms Development, Design, Testing and Documentation

Lior Hassan

Design, APIs Integration, Testing and Documentation

Group Members

Project Mentors

Prof. Ohad Ben-Shahar

Professional Mentor, Project Idea and Vision

Founding and acting director of the Interdisciplinary Computational Vision Laboratory, Computer Science Dep.

Dr. Achiya Elyasaf

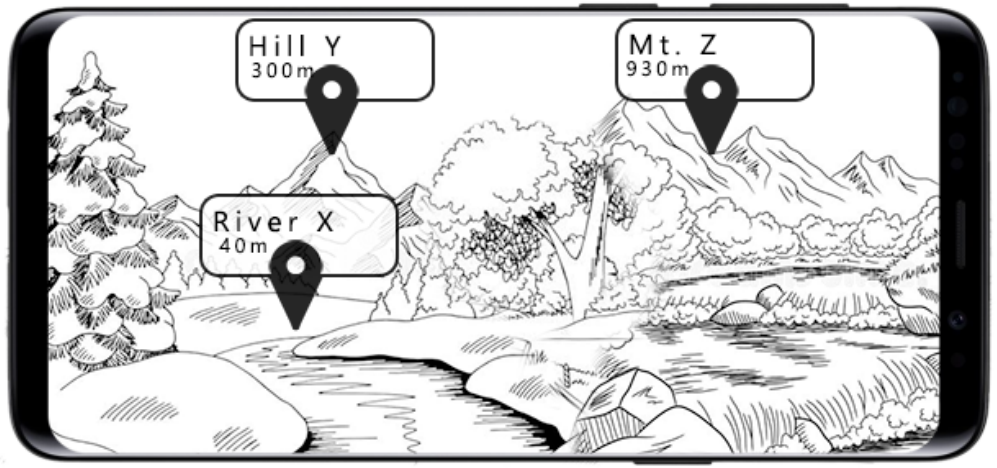
Academic Mentor

Department of Software and Information Systems Engineering, Faculty of Engineering Sciences

Overview & Motivation

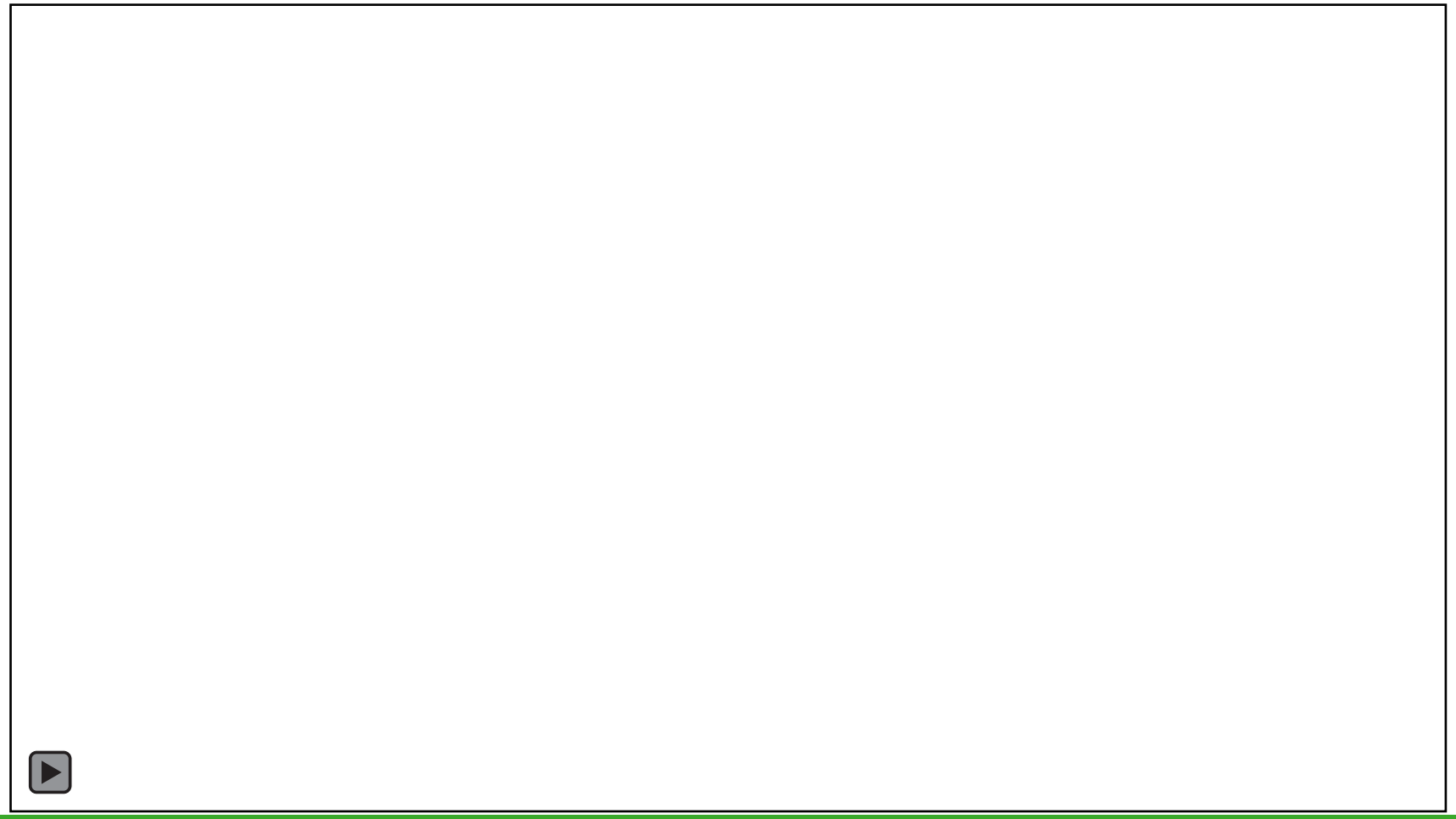
GeoScene is an android mobile device application that uses Augmented Reality technology in order to provide users, mainly travelers and tourists, information about the surrounding points of interest. GeoScene estimates the users Field of View and takes only the points of interests that are in his physical line of sight into consideration. Additionally, the app provides the users an interface to interact with an open-source map provider and allows him to tag new locations in several ways that can be later seen by the other app users.

Vision



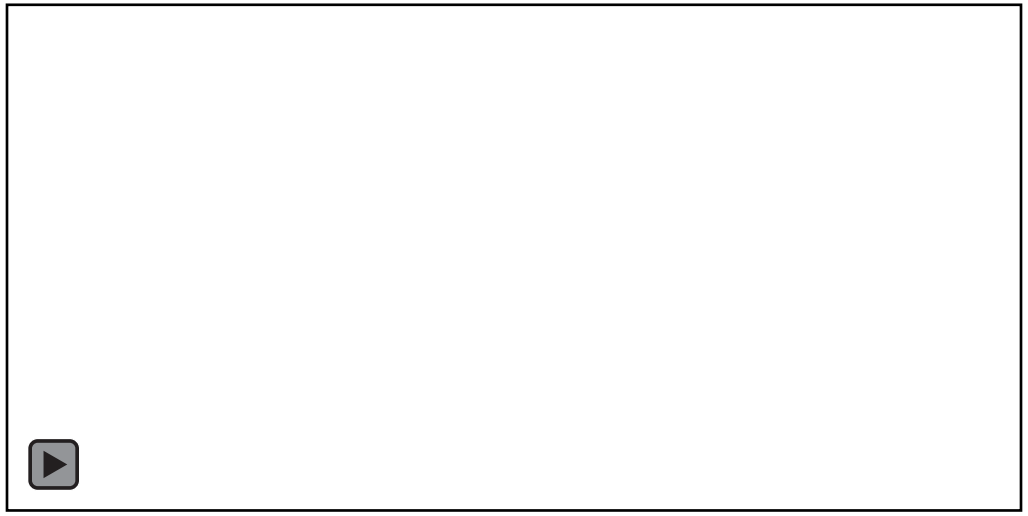
Product

Promotional Video



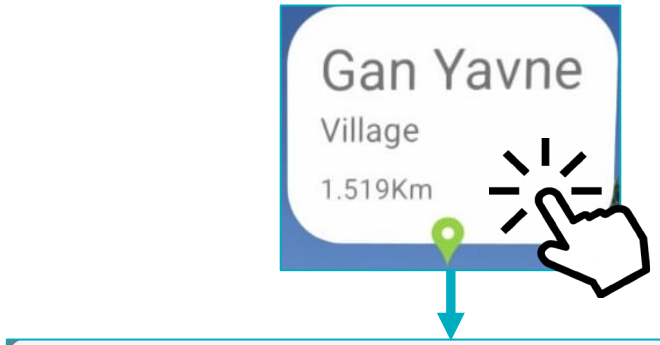
Main Features & Requirements

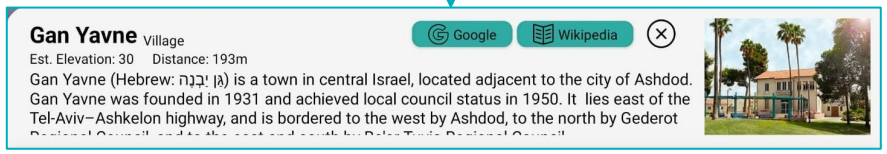
- Supports displaying nearby location markers over real-world view using the mobile device camera by using AR technology .
- Location markers size and orientation is determined relative to their height and distance from the user.
- Displays the name, distance, and type for POIs in the AR view.
- Pre-Downloads analyzes and uses topographic elevation raster data to estimate the user's relative height above ground.
- Estimates the users Field of View and filters surrounding Points-Of-Interest according to the calculated FoV.
- Uses faster loading caching in order to store recent data (POIs and elevations) and will re-load that data from cache when needed.

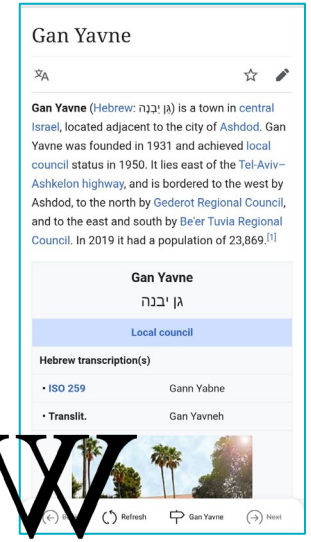


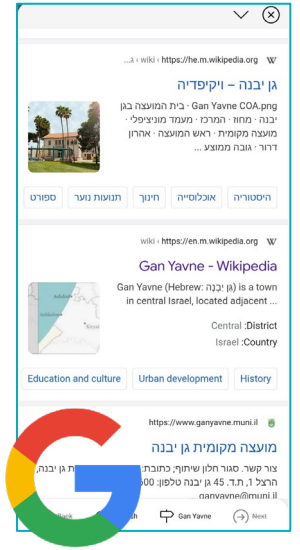
Main Features & Requirements

- Displays Wikipedia based information and allows Google search for POIs.



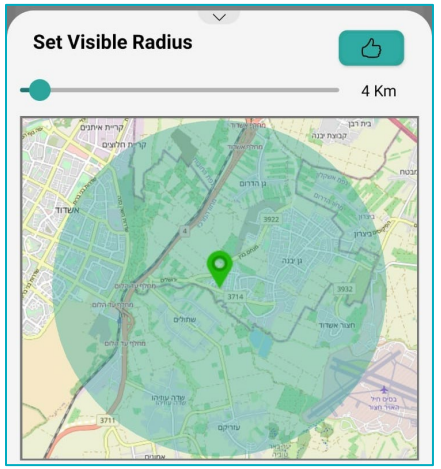


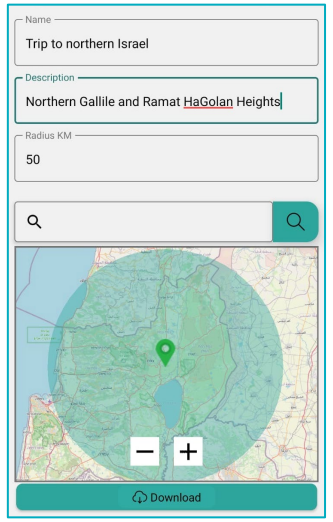




- Allows the user to specify his visible area circular radius to display POIs in.

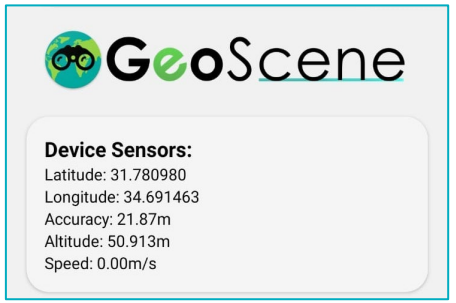
- Enables the user to download POIs and elevation data to be used offline.



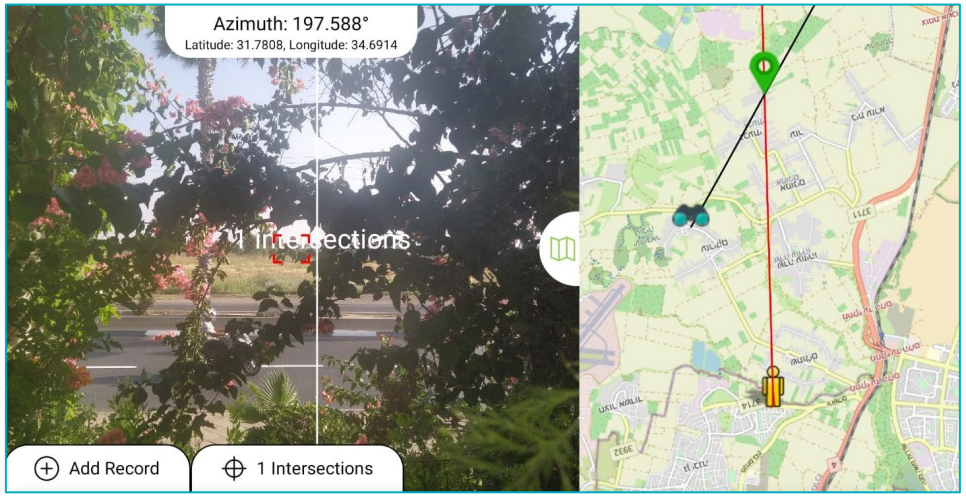


Main Features & Requirements

- Fully integrated with mobile device GPS and Orientation sensors.

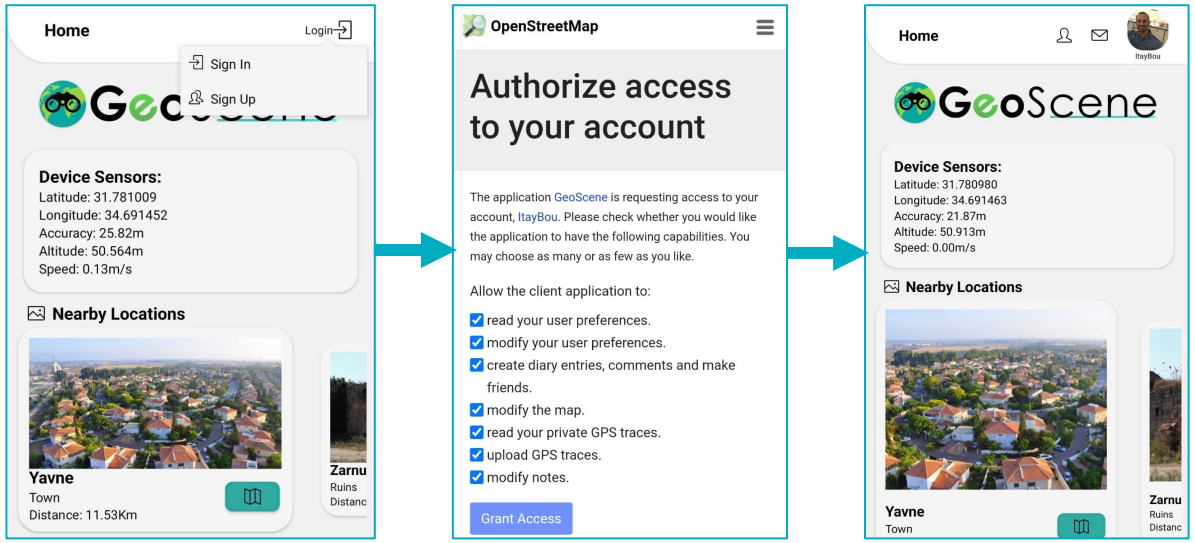


- Estimates two users' line of sight intersection using triangulation techniques.



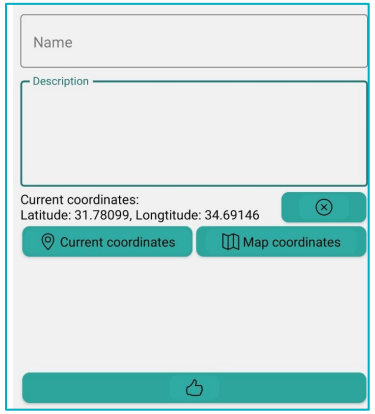
Main Features & Requirements

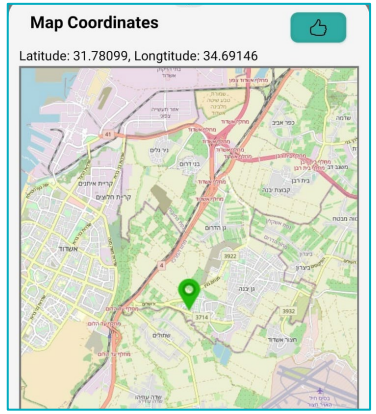
- Uses OpenStreetMap OAuth service in order to identify users.

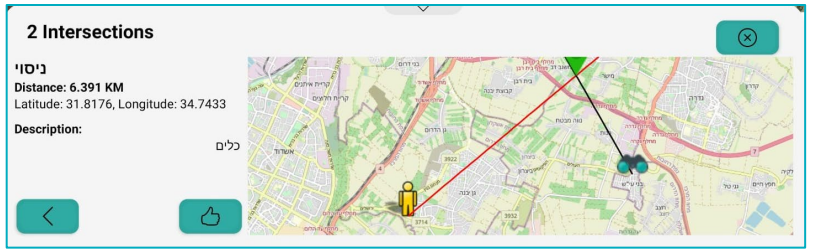


- Provides an interface for users that are logged in to add points of interest by their current location, map coordinates or using the triangulation feature.

- GeoScene supports maps provided by OpenStreetMap.







Challanges

- Since all the libraries, APIs and SDKs that were used are open-source the requests and functionality was sometimes limited, and a workaround was needed.
- When trying to determine the users FoV, precision is especially important because a too sensitive error tolerance can cause visible areas to not be marked properly.
- The algorithms that were used to compute the users viewshed triangulation and POI intersection had to be optimized to run on mobile device due to their computationally expensive nature and resource consumption.
- Offline usage and caching requires it to handle large amounts of data and store them locally in an efficient way such that the retrieval will be fast enough.
- Extensive geographic and topographic knowledge was necessary and familiarity with the common terms used in the field was needed.

Technologies

React Native

TypeScript

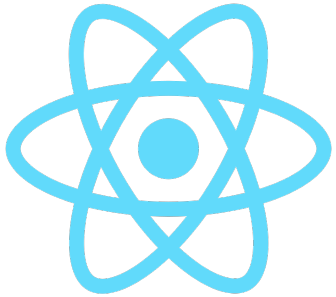
UI and API integration

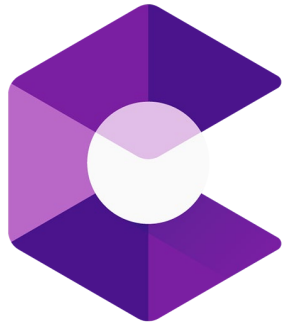
Android Java

Used for intensive calculations.
FoV Analyzation, Triangulation, Raster parsing, Caching, Mobile device sensors management

Google ARCore

Used to initialize and render AR Components





Data Management
External APIs

MediaWiki API
Wikipedia textual data retrieval and wiki page linking

OpenStreet Map
Used for OAuth user Authentication, Map Tiles and POIs CRUD operations

Overpass
Used for map features querying and fetching nearby POIs using OverpassQL

Open Topography
Topographic Raster tiles and elevation retrieval

MediaWiki

Overpass API

OpenTopography

Data Management
Internal Data and DBs

RealmDB

Local device database used to store and query application cache

MongoDB based NoSQL Android internal object store database

Firebase

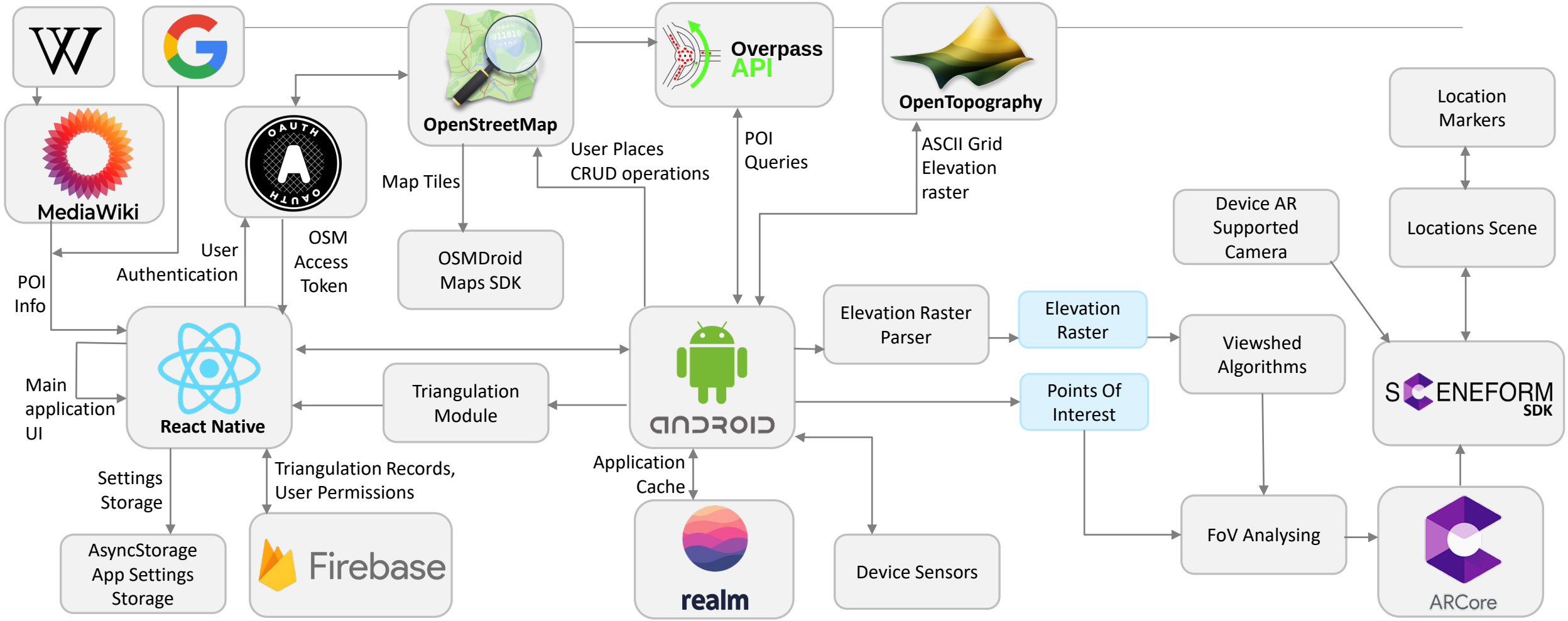
Dynamic real-time storage used to store raw triangulation data and as a user permissions proxy mediator

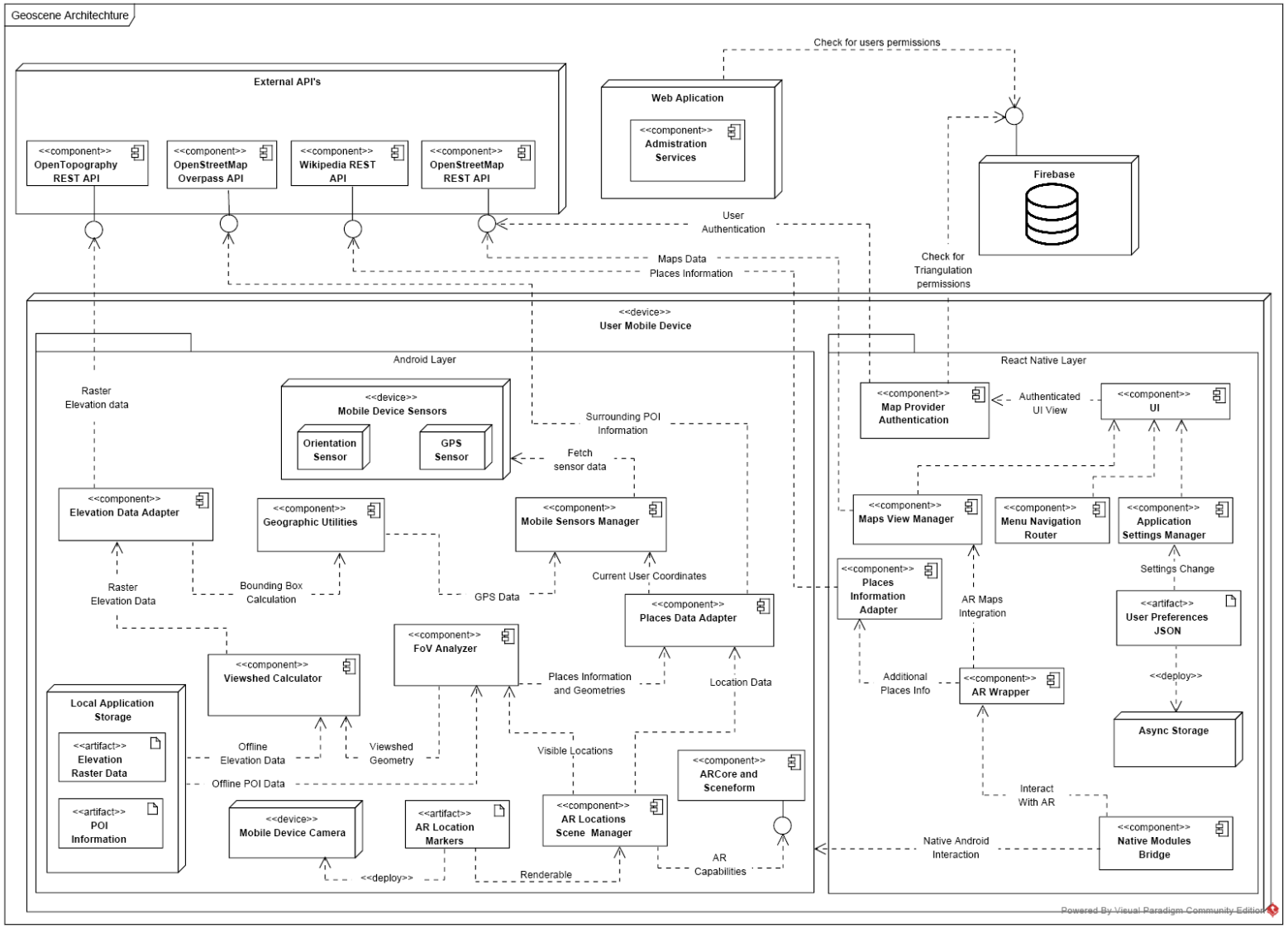
NoSQL database hosted by Google

Main application use case workflow

1. On initial application launch fetch the users GPS coordinates using the device sensors.
2. Fetch and analyze the following data concurrently:
    1. Fetch and analyze elevation raster data –
        1. Calculate the user's bounding box according to his current location and input visibility radius.
        2. Make a request to OpenTopography API with the calculated bounding box.
        3. Parse the response ASCIIGrid elevation data into a 2D matrix.
        4. Apply the viewshed algorithm on the input matrix and determine the visible raster tile coordinates.
    2. Fetch and analyze surrounding POIs information and geometry –
        1. Make a request containing the visibility radius to the OSM Overpass API to retrieve the POIs surrounding the user within the requested radius.
        2. Parse the response JSON to store the needed location information (i.e. name, geometry).
        3. Project the locations geometry to the xy plane using Mercator projection.
3. Intersect the viewshed result geometry with the POIs geometry to determine POIs that are currently in the user's FoV and determine the coordinates to place the location marker in the AR view.
4. For every place that is currently in the user's FoV, generate a location marker and place it in the coordinate location determined for the visible location.
5. Render location markers into the AR scene.

System Design





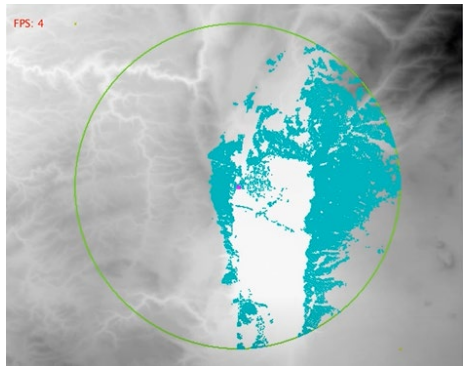
System Design

Algorithms

Viewshed (Determining FoV)

Viewshed analysis is a computational algorithm that delineates a viewshed, the area that is visible (on the base terrain surface) from a given location.

In Order to determine the users FoV we use a viewshed algorithm combined with Bershnham rasterization algorithms for straight lines and circle perimeter.

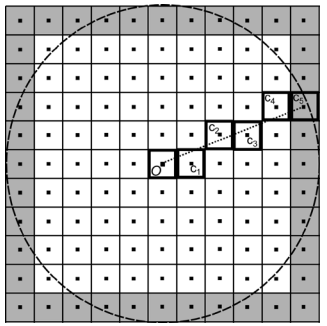


Algorithm Visualization written in Java Processing

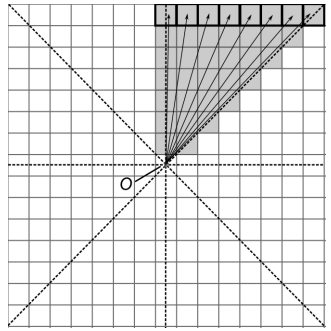
Viewshed algorithm general idea

Viewshed(Observer, radius):

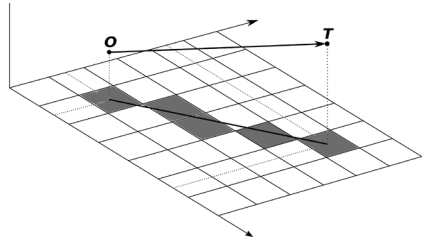
1. Bounding corners = bounding box rectangle(Observer.x, Observer.y, radius)
2. Init array[][] viewshed within corners
3. Calculate the rasterized circle from the viewshed array within the radius using bresenham circle algorithm.
4. For each cell in the calculated circle:
    1. Calculate the rasterized line from the Observer cell to the circle cell using bresenham line algorithm.
    2. Max slope = -inf
    3. For each cell in the calculated line starting from the observer cell:
        1. Calculate the slope of the line between the observer and the current line cell
        2. If slope is > from previous max slope:
            1. Max slope = slope
            2. Viewshed[cell.x][cell.y] = true
5. return viewshed



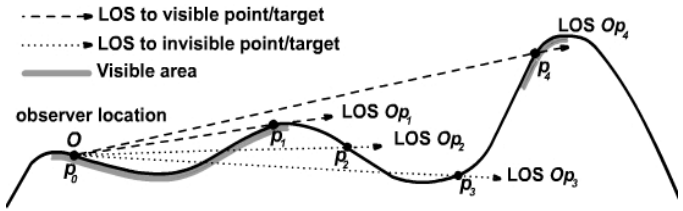
Circle rasterization



Viewshed computation along raster lines



Line rasterization



Viewshed calculation along a line

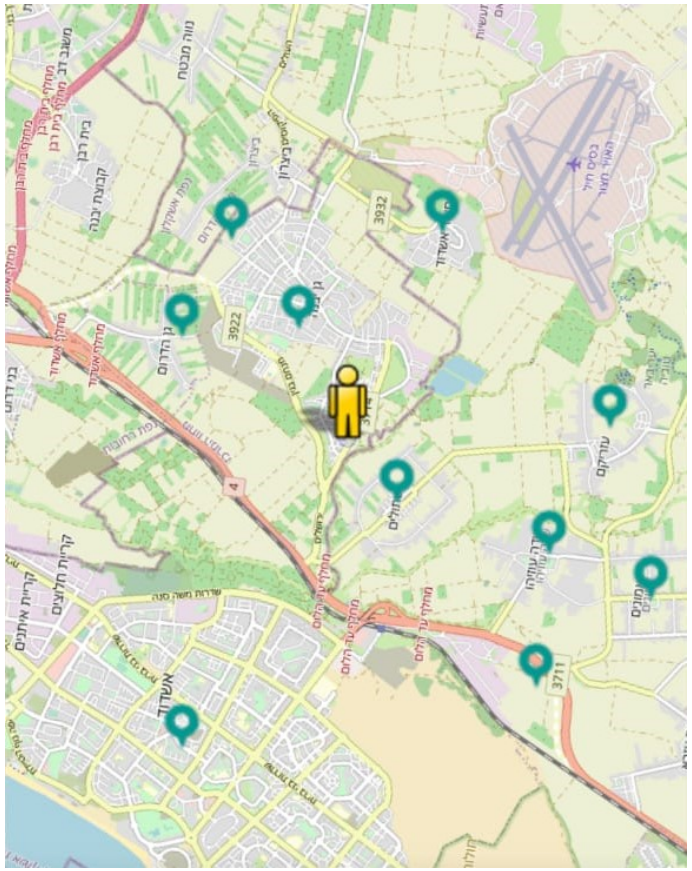
Algorithms

Analysing FoV

Most of the POI data we retrieve using the Overpass API is an areal data, meaning a place is identified by its geometry.

We intersect the ViewShed algorithm output data with the areal geometry data in order to determine what places are in the users FoV and should displayed.

The output of the analyzation process is a set of coordinates indicating the coordinate in which the location marker will be places in the AR view.



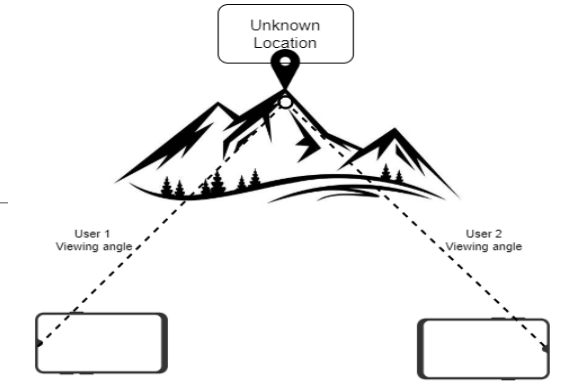
Algorithms

Triangulation

In trigonometry and geometry, triangulation is the process of determining the location of a point by forming triangles to the point from known points.

We use triangulation in order to estimate intersections of the lines of sight of two users in order to determine and tag an unknown visible location.

In order to estimate the seemingly straight lines used for triangulation on the WGS84 Geodetic earth system, we used great-circle distance arc calculation in order to produce accurate enough intersections of two arcs.



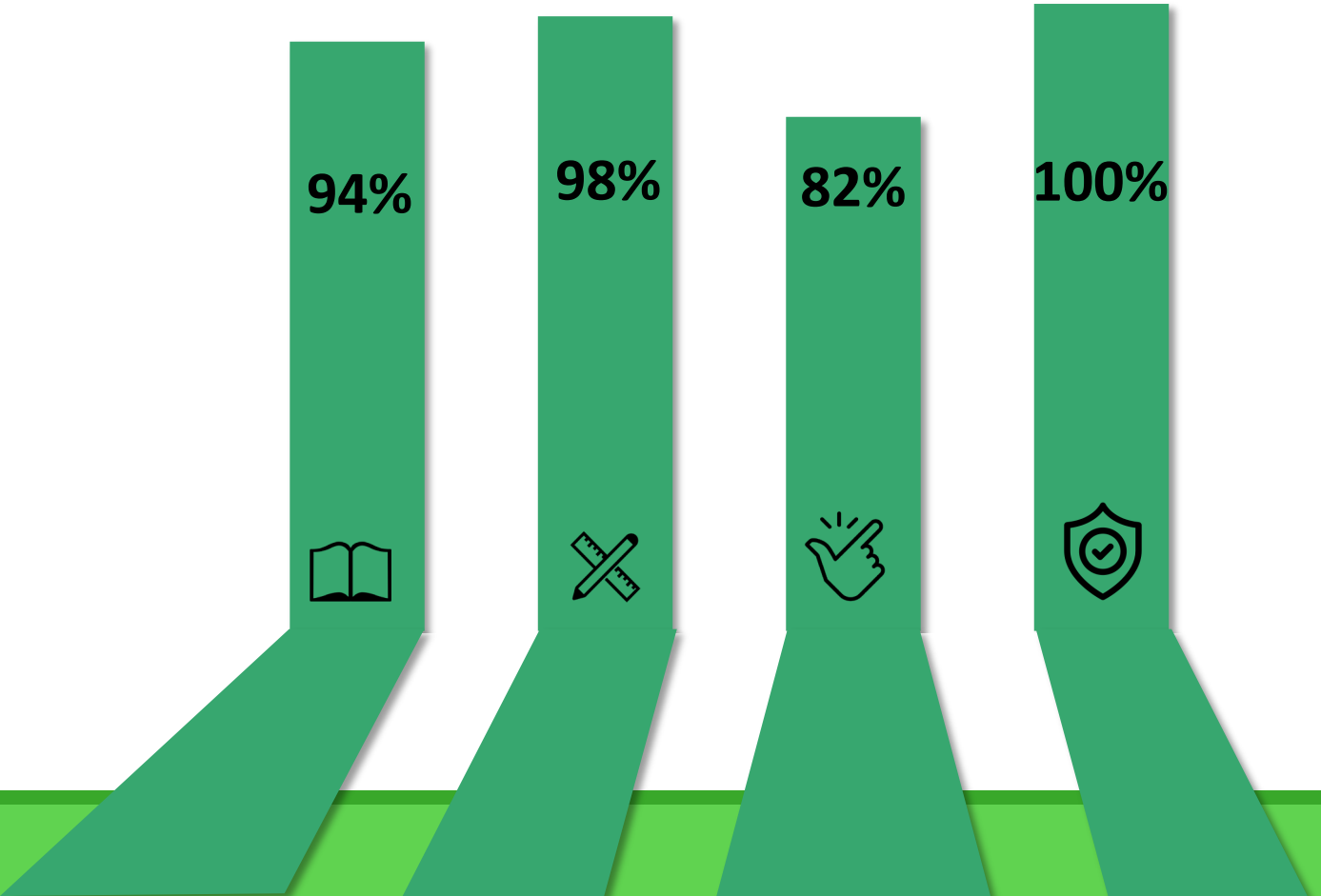
Triangulation process illustration

Testing

- Unit tests performed with Junit and Mockito in order to check that all the units of the application behaves as intended.
- the API tests validate the functionality, reliability and performance of the external resource's interfaces. Since our application performs API requests to third party providers, we tested that those requests are handled properly.
- In order to test the application UI interactions, we used Espresso automated test recorders. The UI tests were executed on Android emulator that simulated the application use on a real Android mobile device.

User Reviews

User reviews gathered, currently according to 16 users.



Understanding Application Workflow

Checked that the users understand how to use the main functionality of the application.

UI and Design Language

Check the application UX and gather the user opinions about the UI visual aspect of the application.

Ease of use and Usability

The users rated the ease of use of the application and the usability.

Reliability

The users were required to notify us if they experienced any issues or crushes during application usage.

Live Presentation

Questions?

GeoScene

Augmented Reality Explorer

Thank You