GO FISH!

Abhishek Paul

TABLE OF CONTENT

WHAT IS IT?

GoFish is an Android application with a Python Flask backend that has a lot of helpful and cool features for hobbyist fishermen. It uses services like Google Maps, Fishial.AI and Open-Meteo API to offer you everything you need at your fingertips.

You might be a novice trying out a new hobby or a seasoned pro who grew up on the lake, GoFish is just the right tool for you.

TECH STACK

MOBILE APP

Android

Kotlin

Gradle

RoomDB

Retrofit

MaterialUI

THIRD PARTY APPS

GoogleMaps

USGS Topographical Maps

Fishial AI

Open-Meteo API

Firebase Auth

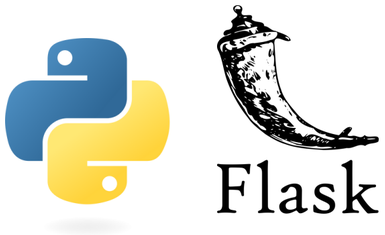
Glide

BACKEND

Python

Flask

PyJWT



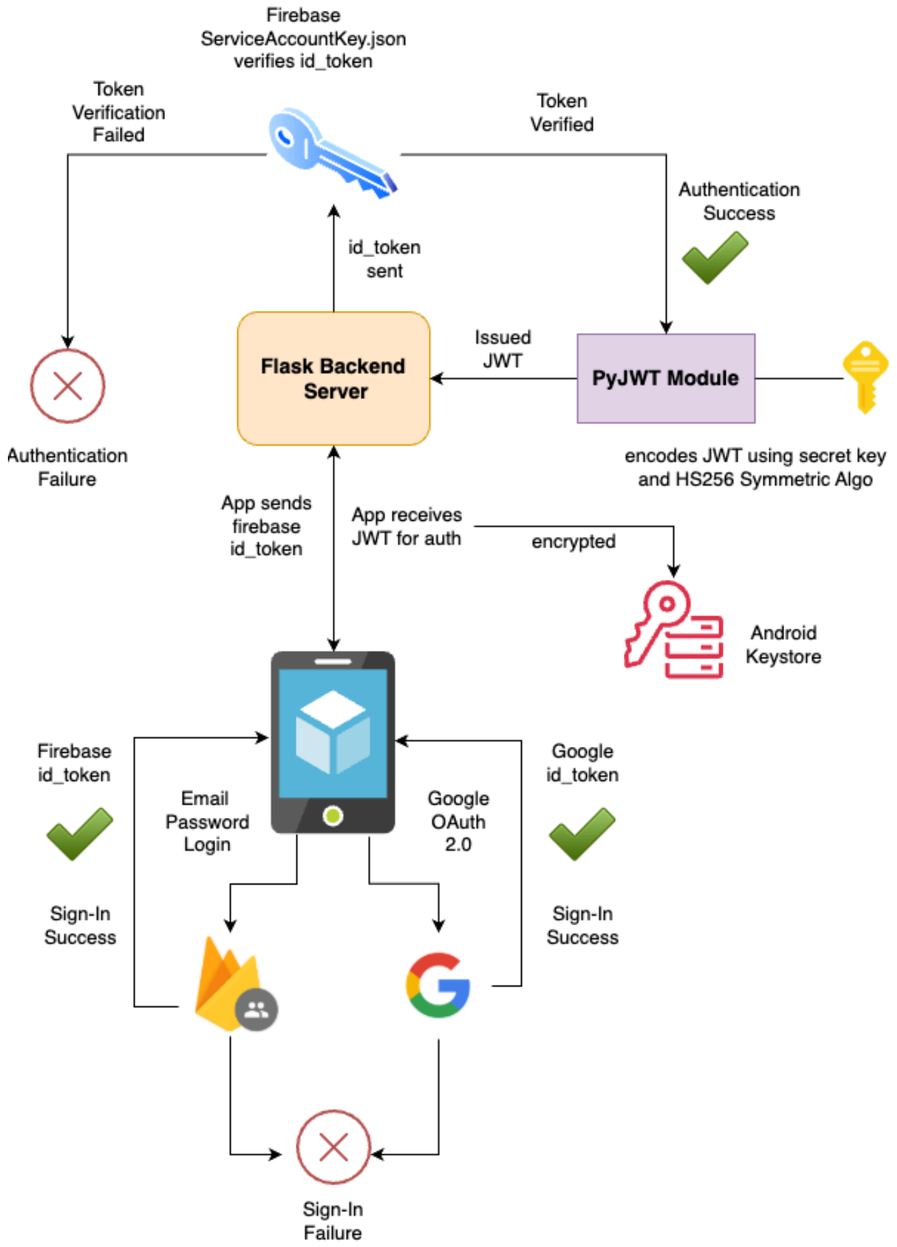
03.

LOGIN AND AUTHENTICATION

Go Fish

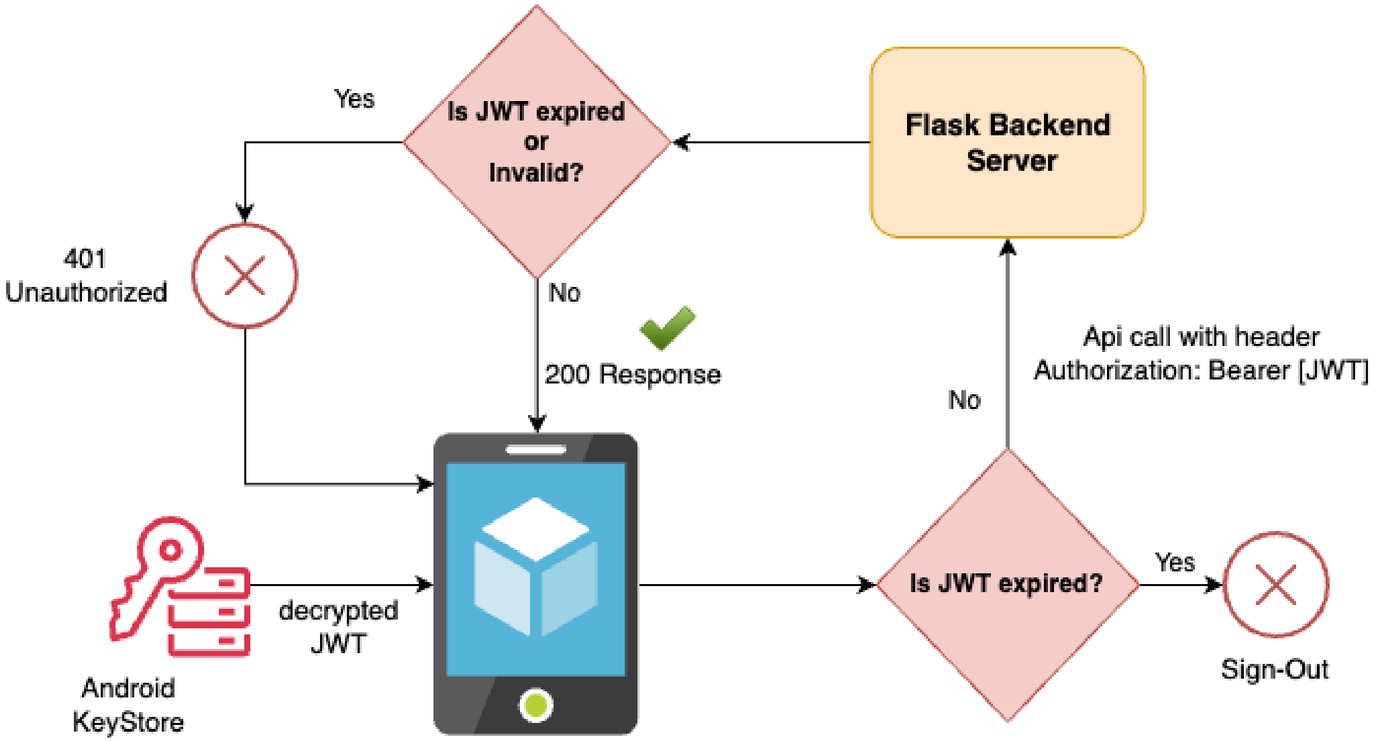
LOGIN

- Firebase Authentication for user management.
- Google Oauth 2.0 and Email/Password Sign-In Flows (requires **email verification** for identity protection).
- SHA-1 app signature used to register Android app to Firebase project.
- Upon successful sign-in, firebase id_token sent to backend with /auth/login call.
- Backend verifies it and generates a JWT token (valid for 90 days) with HS256 (symmetric) algorithm and signs it with a secret key.
- JWT is sent to app which encrypts it and stores it in the Android KeyStore.
- This JWT is used for session management and authenticating subsequent API calls.
- Session expires after 90 days, user gets signed out. Fresh sign-in needed.



AUTHENTICATION

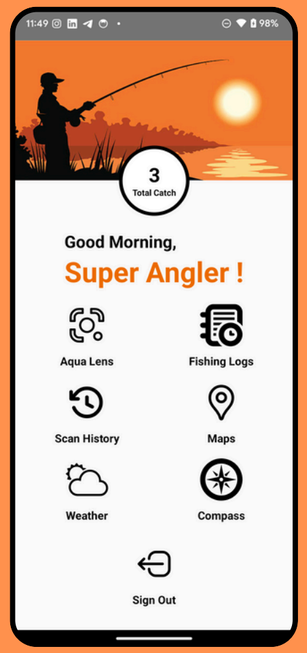
- On every fresh launch, the app checks the decrypted JWT from KeyStore. If its expired, the app signs the user out and clears all data.
- Post login, every subsequent call to the backend is made with an Authorization header - Bearer [JWT].
- All endpoints of the backend are protected.
- It verifies the Bearer token in request header.
- Returns 401 If
  - JWT is **invalid** - signing key of the key does not match with the key used by the backend for issuing tokens, or
  - JWT is **expired** - was issued more than 90 days ago.
- Otherwise the backend proceeds with the request.
- If the app receives a 401 response from the server, it immediately signs the user out and clears all related data.



BACKEND

- The app interacts only with the backend.
- The backend talks to all the third party services, validates their responses and formats them in a way the app wants.
- There are three protected endpoints in the backend.
    - **/auth/login** - It expects the firebase id_token in the body and returns the JWT to the app. Used for authentication.
    - **/fish/identify** - It expects a fish image as a ‘file’ in the formData and returns all relevant details about the fish.
    - **/weather** - Returns weather data. It expects either of the following query params:
        - latitude, longitude, and timezone - used for fetching data for current location.
        - city or place name - used for fetching data for any random region.

05.

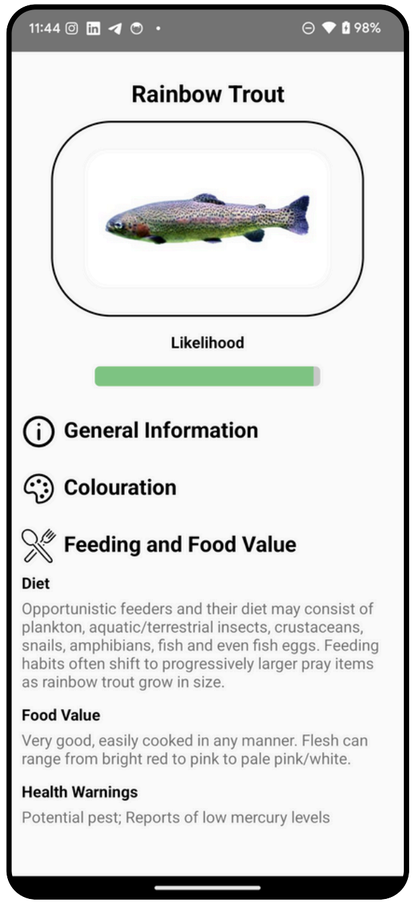


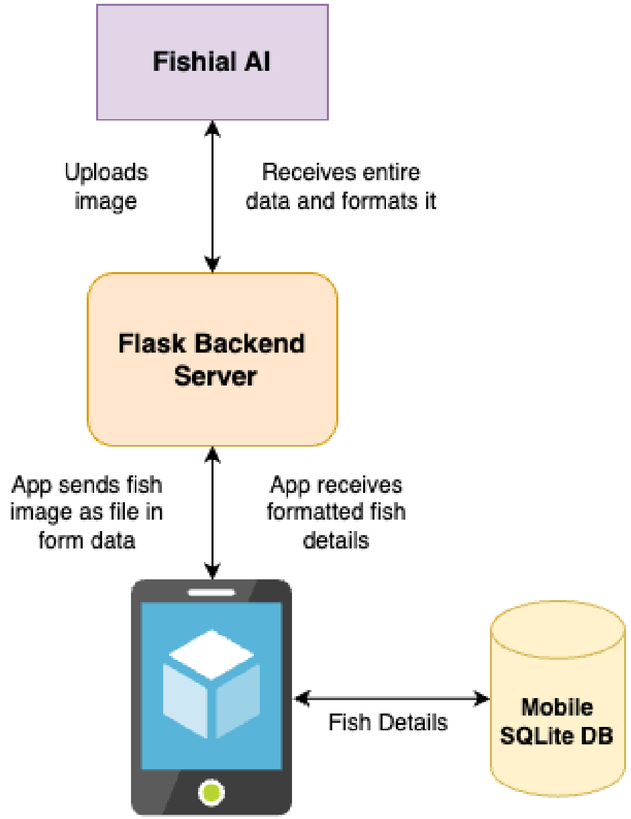
FEATURES

Go Fish

AQUA SCAN

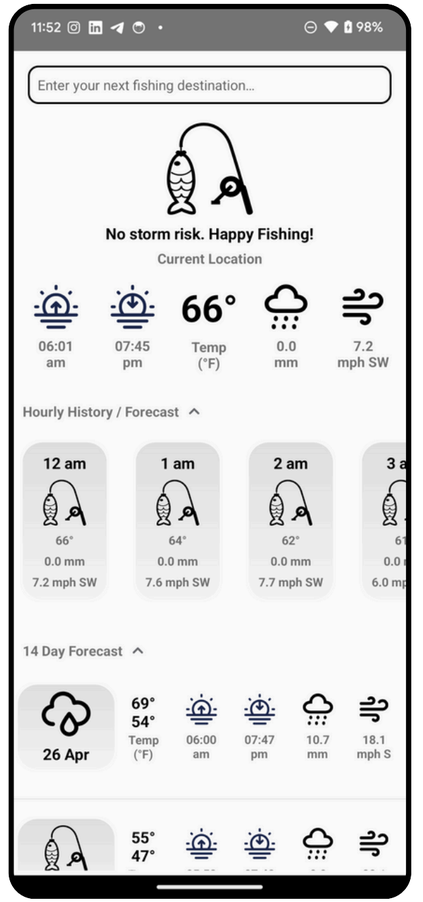
- The app allows the user to click a photo of any fish and it identifies the fish and also returns a bunch of insightful and meaningful facts and information about it.
- The clicked image is sent to the backend as a formData ‘file’ in a MultiPart request.
- The backend server uploads the image to Fishial AI, which is a third party service using ResNet and Computer Vision models in the backend to identify the fish and return its relevant data.
- The returned fish details is persisted by the app in the local SQLite DB.
- Thus, the app maintains a history of all past scans which can be retrieved by the user at any time without network dependence.

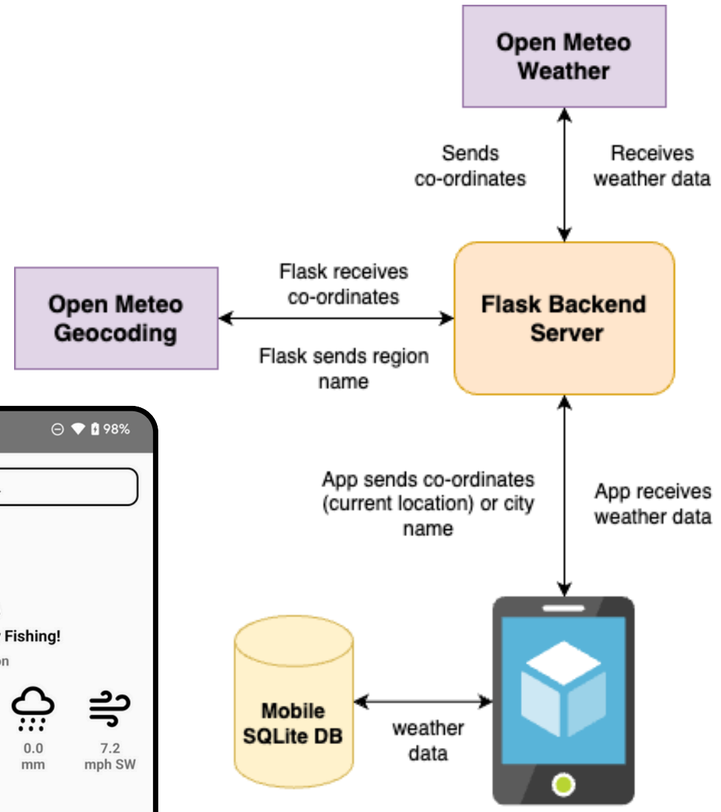




WEATHER

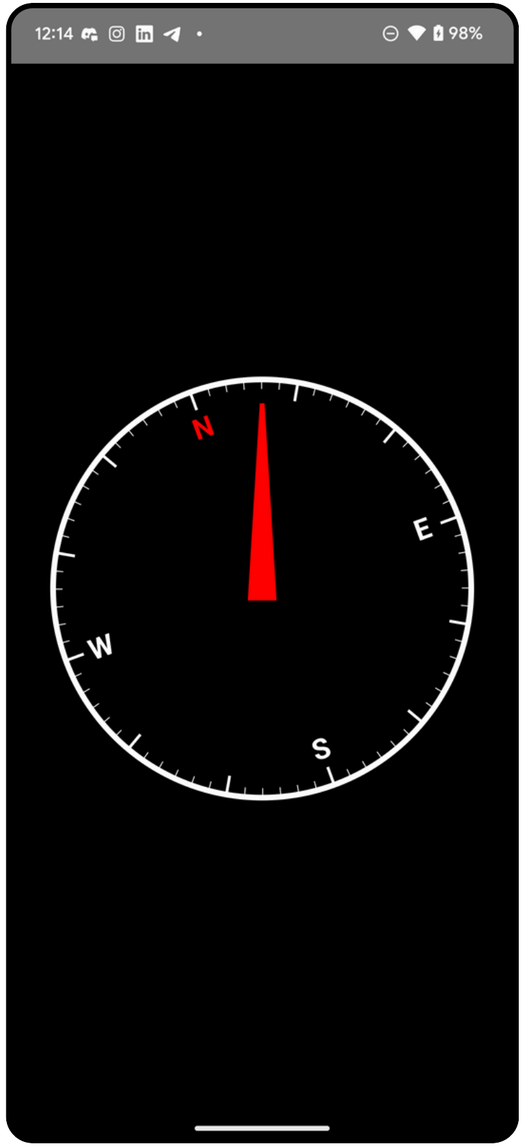
- The app shows the hourly forecast and the 14-day forecast of the weather in the current location, or of any particular region the user enters.
- The backend fetches the data from the Open Meteo weather API by feeding the location co-ordinates and timezone passed by the app to it.
- In case the app sends a region name, the backend uses the Open Meteo geocoding API to fetch the co-ordinates and timezone of the place and then proceeds with the request.
- The returned weather data is persisted by the app in the local SQLite DB.
- Thus, even without network connectivity, the user will be able to view the most recently cached weather data of a place (if available).





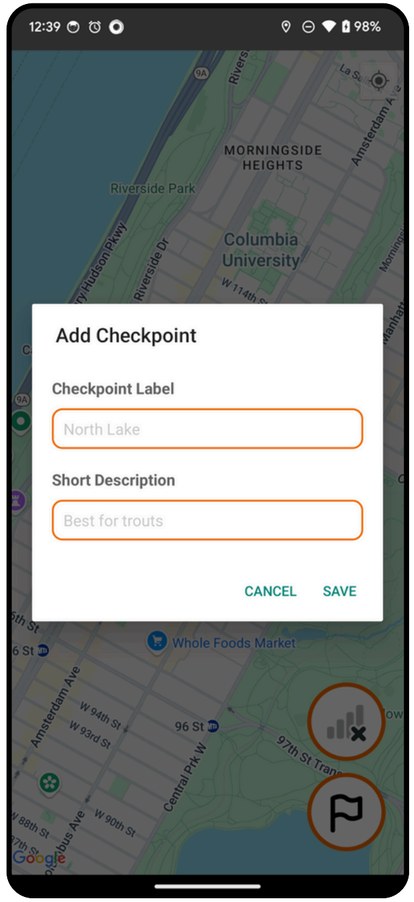
COMPASS

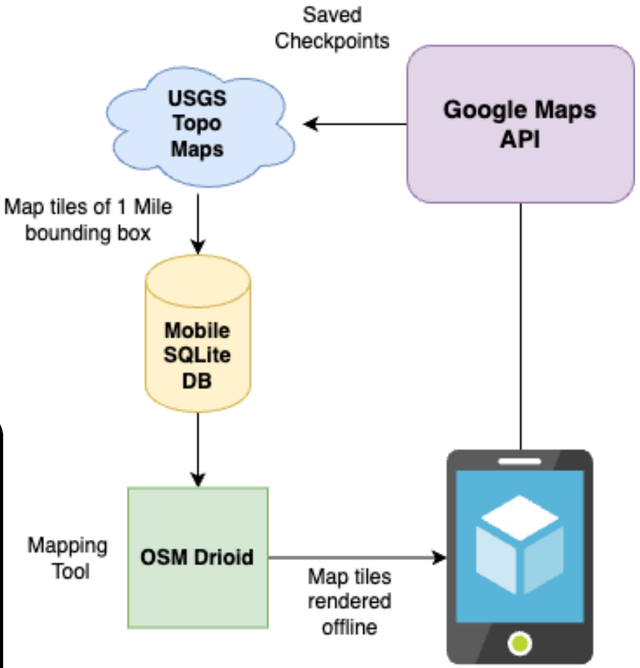
- Built in Andriod from scratch using readings from the device’s accelerometer and magnetic field sensor.
- Created using the Canvas class in Android.
- The readings from the sensors are mapped to 360 degrees and the drawn dial is rotated accordingly to create an effective compass.
- No network or GPS dependence.



MAPS

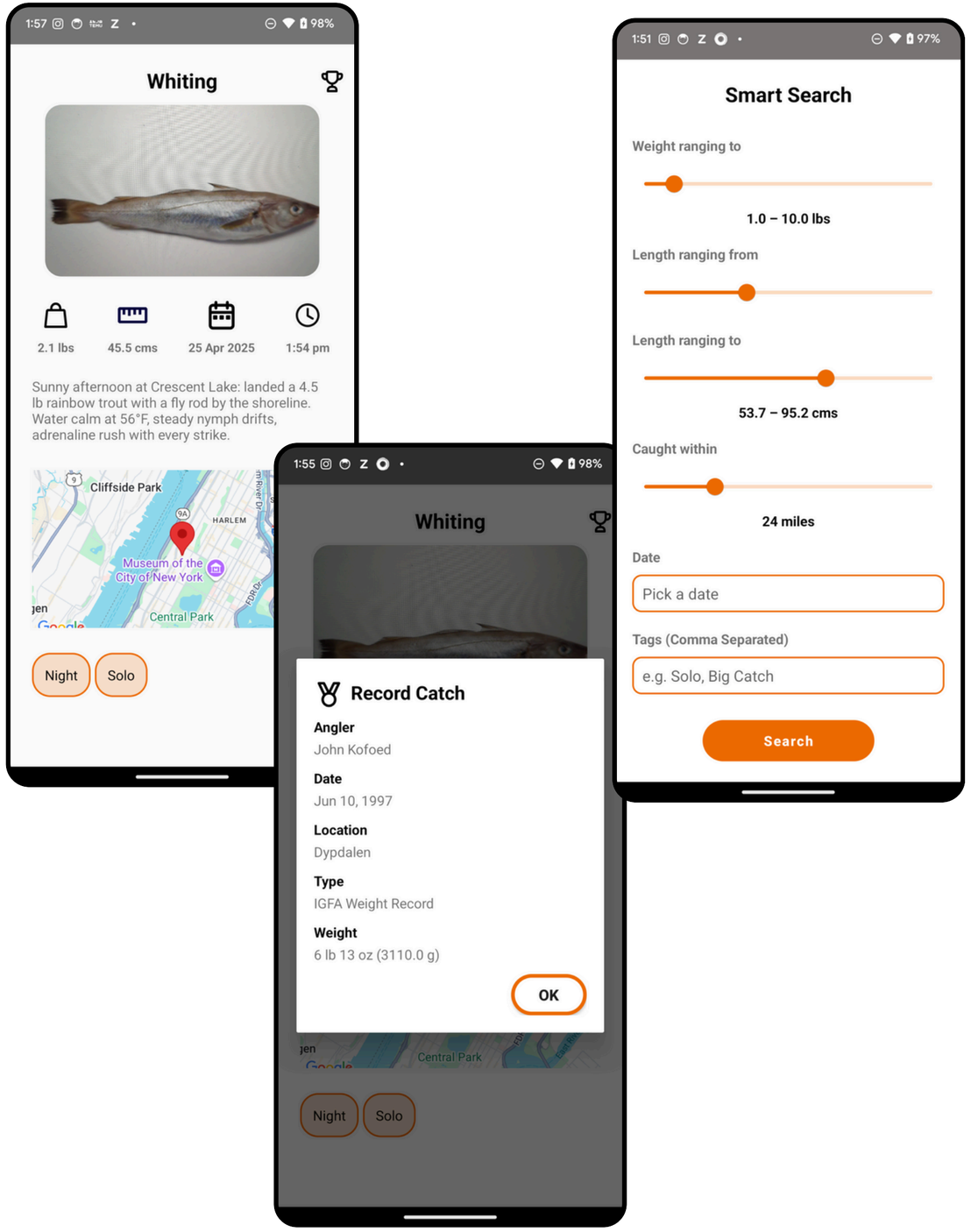
- Shows the current location of the user on a map. (Google Maps API)
- Users can save current location as a Checkpoint, which he can visit later.
- The app downloads map tile data from open source site USGS Topo Maps (which allows free bulk download) of a 1 mile bounding box around a saved checkpoint and stores it locally in the app.
- Third party library OSM Droid is used to render the stored map tiles.
- Thus, even without internet or GPS, the users can navigate in regions around checkpoints.

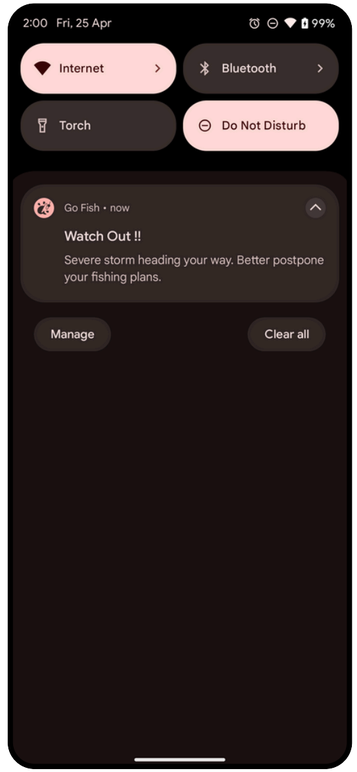




FISHING LOGS

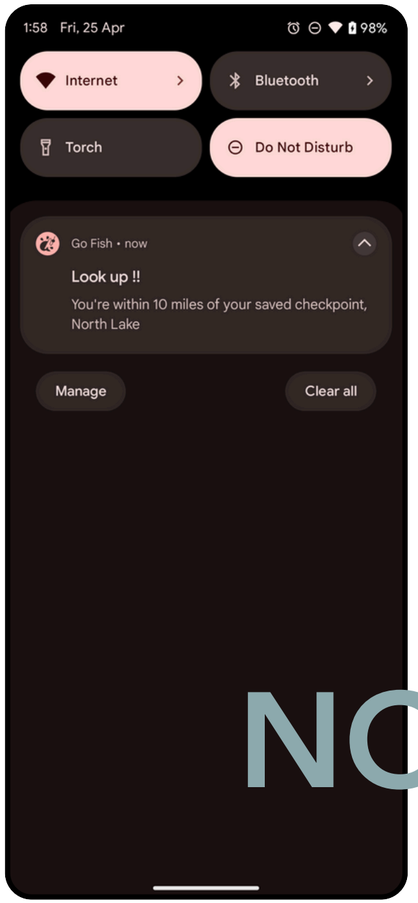
- Users can add logs of the fishes they catch.
- Fields of the logs include name, approx. weight, approx. length, a short description (related anecdote, some tips etc.), tags, an image of the fish, date, time and location of the catch.
- **Smart search** - An intuitive and robust mechanism to filter logs based on the combination of any or all of the above parameters.
- The users can also check the currently held record catch (heaviest fish of that type caught till date) for each of the logs.





Weather Alert

The app alerts the users in case the weather is extreme, prompting them to postpone their fishing plans if any.



Checkpoint Notification

The app notifies the the users of the presence of a saved checkpoint in the vicinity, if they are in a 10 mile radius.

ALERTS & NOTIFICATIONS

TO-DOS

Some best practices yet to be implemented

Code Quality

Static Code Analysis

Tools like PMD or Coverity can be used to improve code quality and remove bugs.

Testing

We can write unit tests for the helper classes and utils using JUnit. While, the android activities and fragments can be tested using integration tests with Espresso.

Deployment

Currently the backend is not deployed. It can be deployed in an EC2 instance and the public IP of the instance can used in the Retrofit module of the app to call the backend. This ensures the app is always up and running and ready for public adoption.

FUTURE PROSPECTS

SESSION RENEWAL

The backend should ideally use a combination of JWT (Access Token) and a Refresh Token for session management. When the AT is expired, we can use the RT to fetch a fresh pair of tokens to avoid the user from having to sign in every 90 days.

CLOUD SYNC

We can periodically sync the local database with cloud (based on network availability) to allow the users to switch between devices. The backend can be made to post the data to AWS RDS.

DATA PURGING

We need to occasionally purge older data from the local database to avoid memory wastage. Weather data older than 14 days is useless. Also, logs older than 3 months can be deleted from local storage, especially if cloud sync is functional.

THANK YOU

-Abhishek Paul

Feel free to ask any questions.