
GitHub Username: sushantchoudhary

WittyLife - relocate smartly

Description

Relocating to a new place should not be a hassle. WittyLife is the fastest, simplest way to search and share living condition data from countries across the world. Learn about the cost of living, health care and property cost from your favourite country or city and compare it with other places of interest.

Key features:

- Search reliable living condition data of a city or country
- Compare cost and other statistics to make informed decisions
- Share information with friends and family

Intended User

This app is intended for people planning to relocate or travel to a new city and interested in learning about quality of life, healthcare and cost of living.

Features

List the main features of your app

- Search city or country data
- Filter by criteria (e.g health care, pollution)
- Compare data between countries
- Share details with other apps

Backend : Numbeo API (https://www.numbeo.com/api/doc.jsp)

Stretch : Enable Auth and Chat functionality (Using firebase messaging and chatkit library)

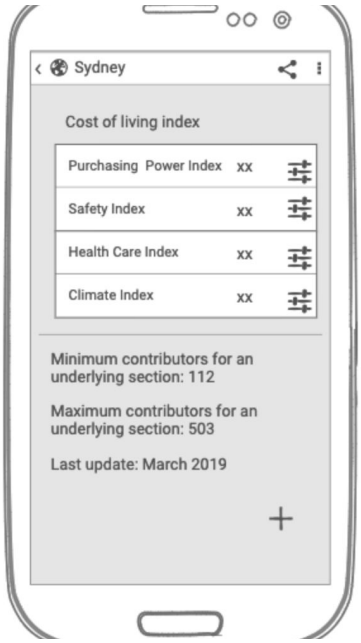
User Interface Mocks

Screen 1

MainActivity : Allows users to search for a destination, select a popular destination by selecting its image or view predefined filters e.g Quality of life ranking etc . Track user location and if permission granted show user's city and nearby cities data instead of popular destination.

Search spinner opens a single select list of cities.

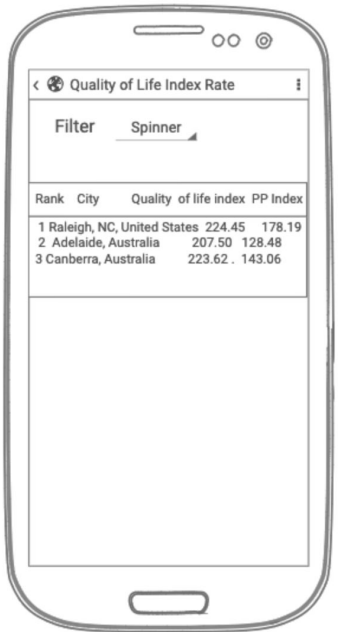
Screen 2



DetailsActivity : Shows details of selected city or country along with horizontal bar to reflect the ranking. Screen has menu option to share the data with other apps.

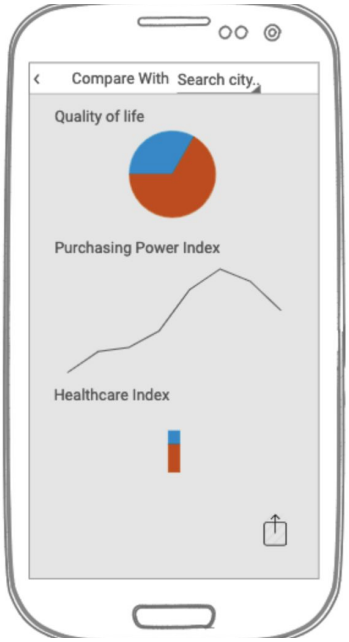
This screen also has a FAB to provide action like "Compare with other city"

Screen 3



RankingActivity: Shows ranks of cities or countries for the selected filter criteria e.g Quality of life. Filter spinner shows single select list to pick another filter option e.g “Healthcare Rank”

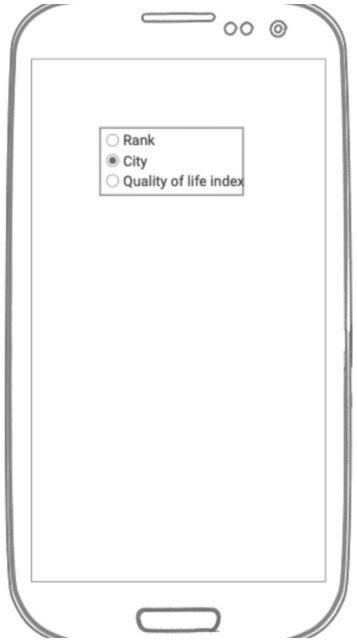
Screen 4



Comparison Activity : This screen comes up as user select “Compare” from FAB action in DetailsActivity. Shows graph to illustrate comparison of data between 2 cities or countries. Also provides a Share action button.

User can select the destination from app bar spinner to update the view with new comparison data.

Ranking Activity Spinner



Single select ranking criteria option

Search City Spinner

Single select city search spinner

Key Considerations

How will your app handle data persistence?

App will use Room with Repository pattern for managing data persistence.

Describe any edge or corner cases in the UX.

1. What happens in case of search when offline? Empty State or (Offline support:Stretch goal)
2. What happens when city/country record doesn't exist?
3. How does app behave in landscape and on tablet?

Describe any libraries you'll be using and share your reasoning for including them.

Retrofit : Handle network calls
Picasso : Image loading/caching
https://github.com/PhilJay/MPAndroidChart : For graph plotting to show comparative data

https://github.com/airbnb/lottie-android : UI Animation

Describe how you will implement Google Play Services or other external services.

Location service : Track user location and if permission granted show user's city and nearby cities data.
Analytics services : Track download and user activity in the app
OR
Crashlytics : For tracking crashes/non-fatal exceptions

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

List the subtasks.

- Create new project and publish to github
- Configure gradle build and libraries
- Identify the min and target SDK

Task 2: Implement UI for MainActivity

List of subtasks.

- Define DB Schema, ViewModel and LiveData based on the API
- Build UI for MainActivity
    - Build list of predefined criteria using RecyclerView with LinearLayout
    - Build grid view of destination using RecyclerView
    - Implement Search feature
- Add Google location services to localize behavior in the Main Activity
- Implement caching for city search list

Task 3: Implement UI for the DetailsActivity

Activity shows list of data for the selected city/country. User can navigate back to MainActivity.

List of subtasks.

- Setup API to fetch the relevant data.
- Table view with list of data for the city/country
- Horizontal bar to show the relative rank
- Implement Quick Share feature
- Implement FAB with “Compare” Action

Task 4: Implement UI for the ComaprisonActivity

Activity is launched as FAB action from DetailsActivity. Illustrates comparison of data between two cities/countries. Use can select a new destination to view the updated comparison.

List of subtasks.

- Setup API to fetch the relevant data.
- Setup ViewModel and LiveData for the activity.
- Create UI for the destination search.
- Implement graph plotting based on data.
- Animate the graph plotting.

Task 5: Implement UI for the RankingActivity

Shows details of predefined filters e.g Quality of Life Index Rate. User can change the filter from the picker.

List of subtasks.

- Setup API to fetch the relevant data.
- Setup ViewModel and LiveData for the activity.
- Create UI for the Filter search
- Create table view to display the data.

Task 6: Implement Google Services

Implement firebase crashlytics to track crash and non-fatal issues in the app.

List of subtasks.

- Setup firebase crashlytics service
- Integrate with the app.
- Test the integration

Task 7: Polish and test the app

Update the theme and animation according to material guidelines. Test the app.

List of subtasks.

- Add animation and transitions to views
- Handle any error cases
- Verify token/API key security
- Verify the behavior on phone and tablet.

Task 8: Prepare for publish to playstore

Follow the checklist to prepare the app for the wild .
List of subtasks.

- Follow https://developer.android.com/studio/publish
- Versioning and Signing app for release.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"