CAPSTONE STAGE 1: PROPOSAL

GitHub Username: bruno78

World Landmarks

Description

World Landmarks allows users to upload or take picture and it will identify which landmark is and by using Google Vision Api, and Wikipedia to bring some information about it.

Intended User

This application is intended for tourists and students interested in learning about famous landmarks around the world.

Features

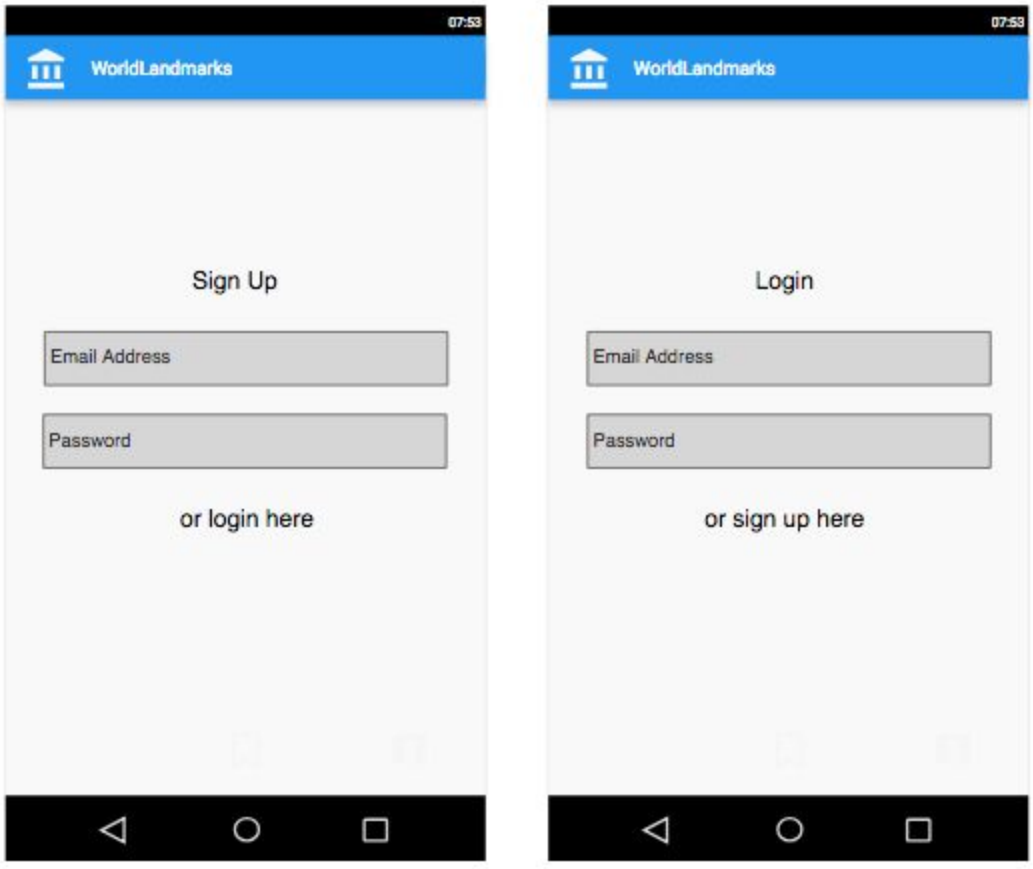
This application will have the following features:

- Take pictures
- Upload pictures
- Use MLKit and Google Vision API to identify the landmark
- Use Google Maps to show the landmark location
- Save picture with landmark information
- Display a list of landmark pictures with info

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

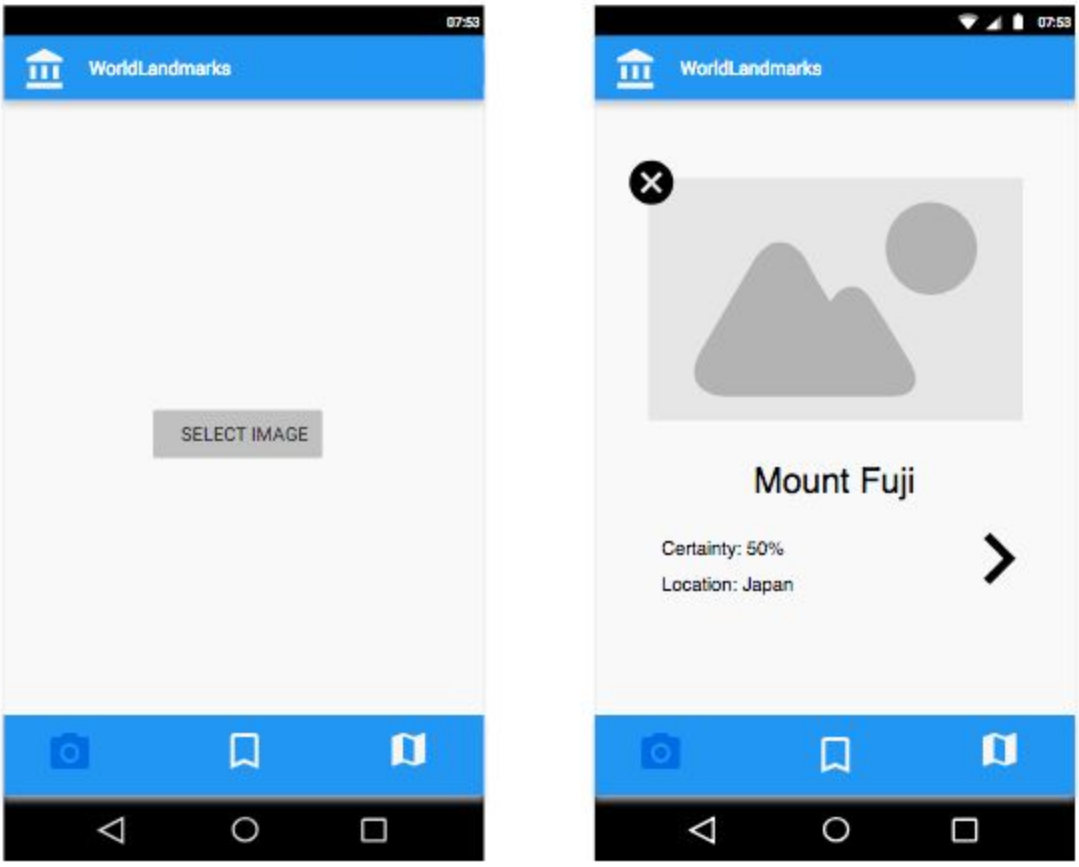
Screens 1 and 2



Screen 1 - Sign up Activity: When the user opens the up, s/he is prompted to sign up. If s/he is a returning user he can click on “login here”. [login here] -> Login Activity

Screen 2 - Login Activity: When the user already has an account s/he can login here. [Sign up here] -> Sign up Activity

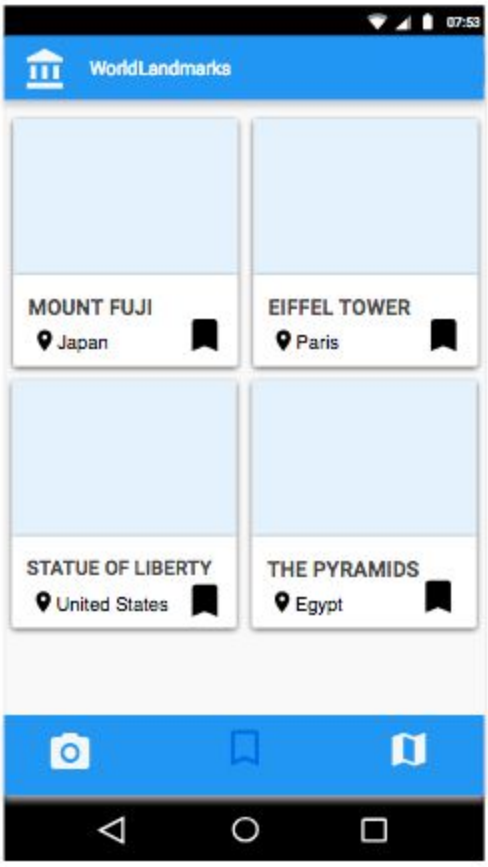
Screens 3 and 4



Screen 3 - Processing Image Activity state 1: When user clicks on the “Select Image” button, s/he prompted to either upload an image or use their camera to capture an image. [select image] -> Processing Image Activity state 2. [Camera icon on bottom bar] -> Processing Image Activity

Screen 4 - Processing Image Activity state 2: Once the image is generated then the application will try to identify the landmark. If the user clicks on the arrow right it will take the user to the landmark information activity. otherwise by clicking on the “X” on the top left of the screen, it will destroy the image, back to the initial state. [x] -> Processing Image Activity state 1. [>] -> Image Info Activity

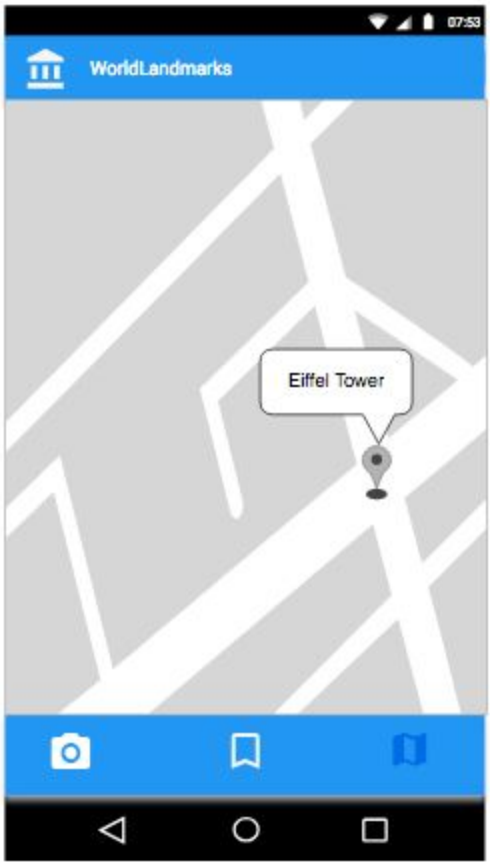
Screens 5 and 6



Screen 5 - Image Info Activity: The user will have the information about the landmark brought by Wikipedia. If s/he decides to save, the bookmark button on the top will save the picture and location info in the database. Otherwise hitting back button will bring him back to the main screen where s/he can take another picture.

Screen 6 - Saved Landmark list Activity: When user hits the bookmarks button on the bottom of the screen, s/he is prompted to the bookmarks activity where they can see all their bookmarks. [bookmark button bottom bar] -> Saved Landmark card list. [card] -> Image Info Activity

Screen 7



Screen 7 Landmark Maps Activity: When user hits the map button on the button, the user will see all the bookmarked spots on the map. [Map Button on bottom bar] -> Maps. [Places] -> tooltip image with info.

Key Considerations

How will your app handle data persistence?

This application will save the photos using Firebase which will be Storage for photos and real time database for data.

Describe any edge or corner cases in the UX.

Since the application uses Firebase, it needs internet connection to run its features. From user's authentication, to process image, saving and deleting data, the internet connection will be

checked for every of these situations. If there's no internet connection, a toast message will be prompted to the user.

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

Firebase will be used for user authentication, storing data and image files. It will keep these features centralized, on the cloud, and easily available across other mobile devices that the user has.
Glide to handle the loading and caching of images.
MLKit for processing landmark images .
ParaCamera for capture and get bitmaps images using the camera. It takes out a lot of the boiler plate on handling images.
Timber for log debug messages.When set for production, these debug logs are removed.
ButterKnife for field and method binding views. This library helps to create a more readable and cleaner code.
Espresso for integration tests

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

The application will be implementing MLKit that depends on Google Vision API, as well as Firebase which all depends on Google Play Services.

Required Tasks

Task 1: Project Setup

Create a new project on Firebase console:

- Configure MLKit fof Landmark detection
- Allow Google Vision api on Google Cloud Console
- Configure authorization for Sign Up and Login
- Configure Realtime database
- Configure Firebase Storage
- Switch Firebase plan to Blase.

Add libraries dependencies to Gradle:

- Firebase core
- ParaCamera

- Material Design
- ButterKnife
- Timber
- Glide

Task 2: Implement UI for Each Activity and Fragment

Build UI for the following Activities/Fragment

- Authentication Activity
    - Fragment for Sign Up
    - Fragment for Login
- Image Processing Activity
- Image Info Activity
- Landmark List Activity
- Landmark Location Map Activity

Task 3: Implement MLKit

Implement MLKit to recognize landmarks:

- Create a new Firebase Project
- Implement MLKit to recognize landmarks
- Enable Google Vision API
- Switch Firebase plan to Blaze to enable MLKit
- Apply landmark recognition feature on Processing Image Activity

Task 4: Create Build Variant

Build two variants

- A paid variant with ad free.
- A free variant with ads on the bottom of the screen + interstitial ad during image processing.

Task 5: Create Tests and Handler Error Cases

Create integration tests and Unit tests to make sure the application handles all corner case. Implement handler error cases.

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**"