Executive Summary

Traveling should be memorable. With hectic lives bombarded by the responsibilities of work and school, the average person tends to travel only one to two times per year. Given that fact, it is vital that the limited amount of time people have for these experiences is fulfilling in every way. How can we ensure this? How can we make travel more exciting, engaging, and interactive for people all over the globe? The answer to this is Travel Trails, a mobile app for travelers to collect virtual models of the places they've visited that will allow them to re-experience their adventures from home and find their next one in the meantime.

Our app's functionality is simple. First, open the app and use the map to find a place to visit or see where your friends have gone to find a nice hotspot. Next, visit and take a photo. Our algorithm will localize the image to a 3D model which is constructed from our crowdsourced dataset of images. This will verify that you were there and then add the 3D model to your gallery. Revisit your adventures later through the gallery and refollow your trails!

Our target market reaches a wide audience. In today's world, there is a community of 450,000 people who geocache per year and 160,000 daily active Pokemon Go users. We appeal to these users and more. From the average vacationer to the avid collector, there is something for everyone to get out of Travel Trails. First, the average vacationer. When I go on vacation, I frequently take pictures of the scenery to capture and revisit later. But admittedly, I rarely revisit them; it is impossible to recreate the feeling of what it was like to actually be there. 2D photo albums offer limited perspective and are highly non-interactive. By comparison, our 3D photo gallery will allow travelers to retrace their steps by taking pictures that contribute to our crowdsourced dataset of images and that will allow them to collect a 3D model of the place they captured. These 3D models will be saved to their "trail", which organizes their adventures for them in an album for them to revisit these places virtually. Next, the avid collector. Our app will have a point system and a leaderboard for people to compete against each other to find some of the most rare and unique places on Earth. Collectors can scavenge the earth to add new models to their repertoire and climb the leaderboard to make themselves known as an adventurer to be reckoned with.

With our target market described, it is important to highlight how our application differs from what is currently available on the market. Our app may appeal to the same type of people who engage in geocaching or Pokemon Go. But geocaching is more for the committed adventurer while Pokemon Go is more for the avid collector. What about the average person who just wants to go on vacation and make memories they can vividly keep for a lifetime? That is what we offer with our five primary competitive advantages. The first advantage is that we are immersive in that customers can retrace their steps through their interactive gallery of 3D models, reigniting their sense of adventure as soon as after work. Second, with our point and leaderboard system, we are incentivizing. Adventurers can compete against friends and a like-minded community of travelers to discover and collect models from some of the most exciting places on Earth. This brings us to the third competitive advantage, which is that we are social. Customers can see their friends' trails and galleries and let that inspire them to visit some of their friends' favorite places on their next excursion. Fourth, we are validating, providing a 3D matching algorithm to our users, verifying that they were actually there. Let it serve as a flashy digital passport that they can show off to their friends. Finally, we are nostalgic, as users can relive their adventures through their trails of 3D models.

While it is beyond the scope of our launch, our application also has potential for research interests that will benefit society. By having people contribute to our dataset of images, overtime we will acquire historical data of landmarks and how they change over time. That opens the door to selling data to corporations as well. However, for now we focus on the benefits for the average consumer. And with our advantages and more, Travel Trails is a no brainer for travelers around the globe.

Technical Summary

Travel Trails presents an immersive travel companion app that encourages users to explore and relive travel experiences by contributing to crowdsourced 3D models. While high resolution models of famous locations exist online, they are often blocked by paywalls. A single model can cost over $40 to purchase, and not all landmarks have 3D models. It is expensive to create 3D models through traditional photogrammetry methods and it typically requires expensive camera/LiDAR equipment that few can afford. However, 3D reconstruction pipelines now exist that can create high-resolution 3D models from a set of photos captured by many different cameras. Travel Trails leverages this method by utilizing our users' images to create models of landmarks with historical and cultural significance around the world.

The infrastructure of Travel Trails is composed of three main elements: (1) the Android app, (2) the backend server, and (3) the imaging processing pipeline. These technologies might not be very innovative when viewed alone, but together they create a powerful platform that makes creating models of the world accessible to the average person. The app will utilize the Google Maps API, Android Scene Viewer, the Camera API, and the OkHttp client to make API calls with our backend server. The API will be served using Python Flask with a Postgres database to store models and user data. A Rocky Linux workstation with four NVIDIA GPUs will provide the computation power to generate our 3D models and process user requests. Images sent to the server will be processed using 3D reconstruction libraries including COLMAP, OpenSfM, and Bundler. These libraries take a set of 2D images and create a dense 3D reconstruction.

Our technical component will be adding fast image localization to the 3D reconstruction pipeline. We also want to be able to highlight areas that users can focus on when contributing to our model dataset. We will use COLMAP as a baseline for this goal, a library that uses matching pairs of features from different perspectives, creating a 3D point cloud and then creating a 3D mesh from that point cloud. However, COLMAP is slow because it runs through the entire image processing pipeline from feature extraction to dense reconstruction. One of our novelties and algorithmic complexities will be developing an algorithm to quickly match new user photos to an existing model and verify user location. We want to reduce this processing latency so that users can get real-time feedback as to whether their image was a match. This will require implementing a fast image registration technique in COLMAP. This will be a challenge, but many papers have been published on this subject, so it is mostly a matter of implementation.

Ryan Hudson, Jonathan Lee, Austin Theriault

Product Specifications

User Stories

As a user, I would like to be able to see a map of landmarks that I can collect models of.

As a user, I would like to be able to take a picture of a landmark and collect it in a gallery.

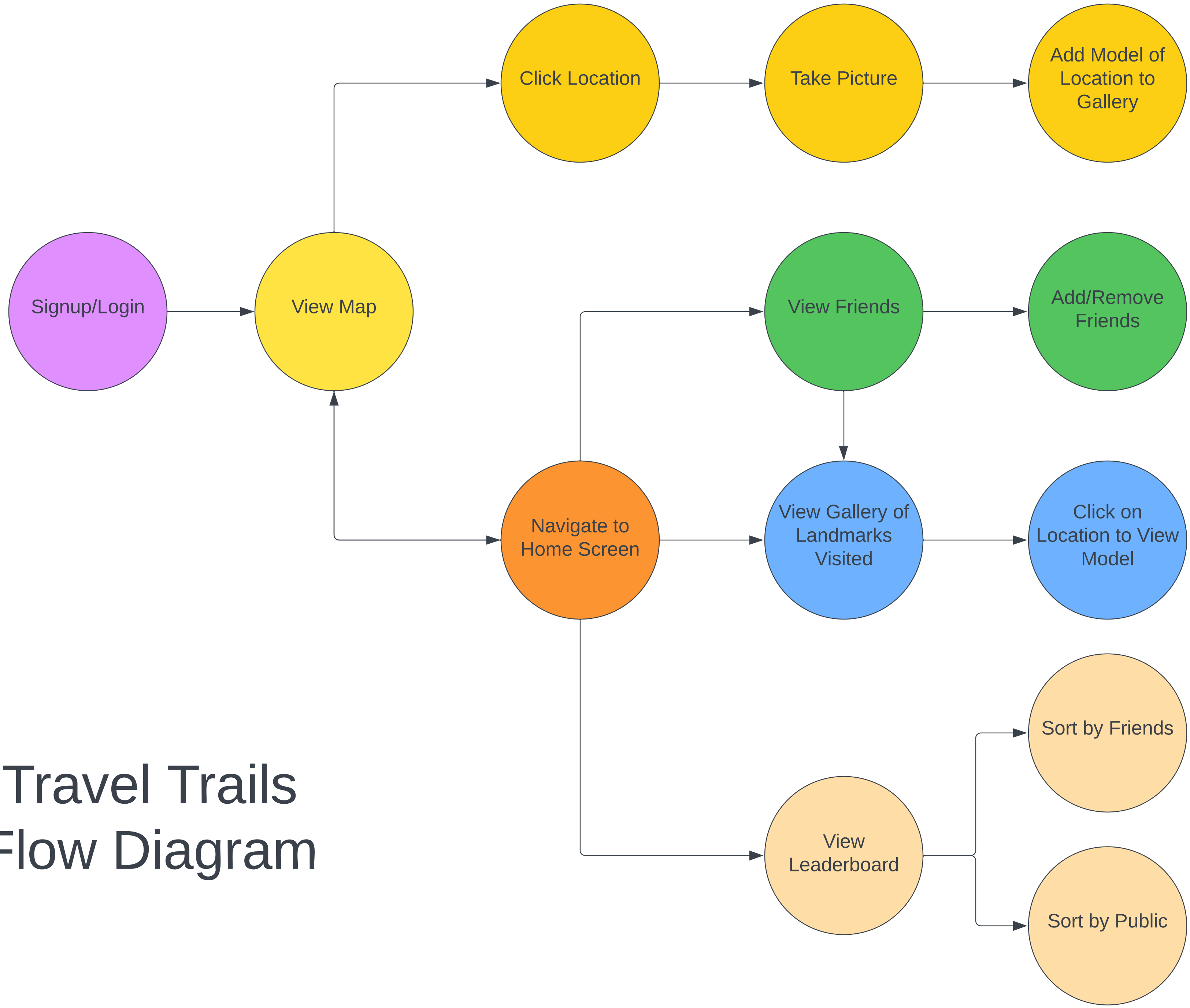
As a user, I would like to be able to view 3D models of my landmarks in my gallery.

As a user, I would like to be able to add my friends.

As a user, I would like to be able to see a list of my friends and be able to view their galleries.

As a user, I would like to be able to collect points for taking my pictures and see how I rank against others.

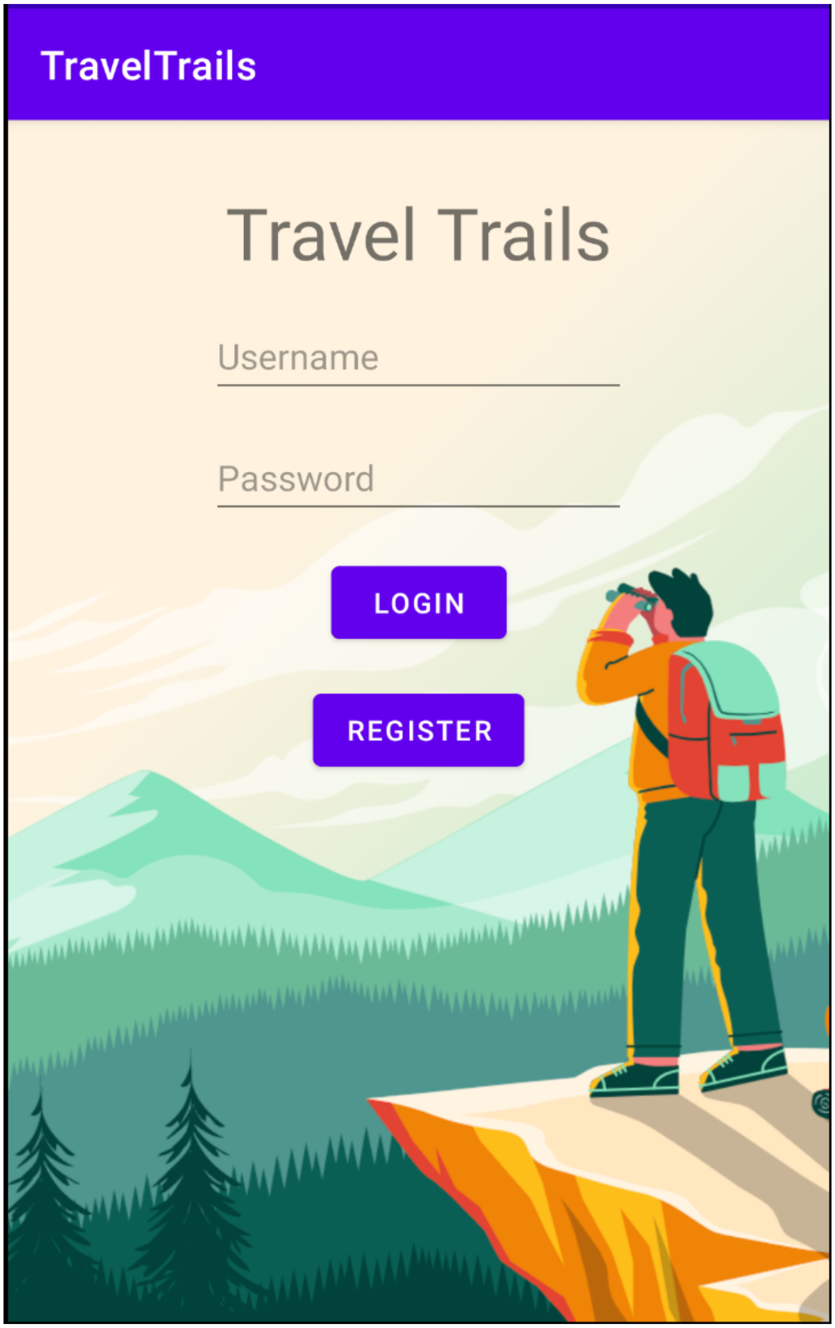
As a user, I would like to see how I rank both against my friends and against the public.



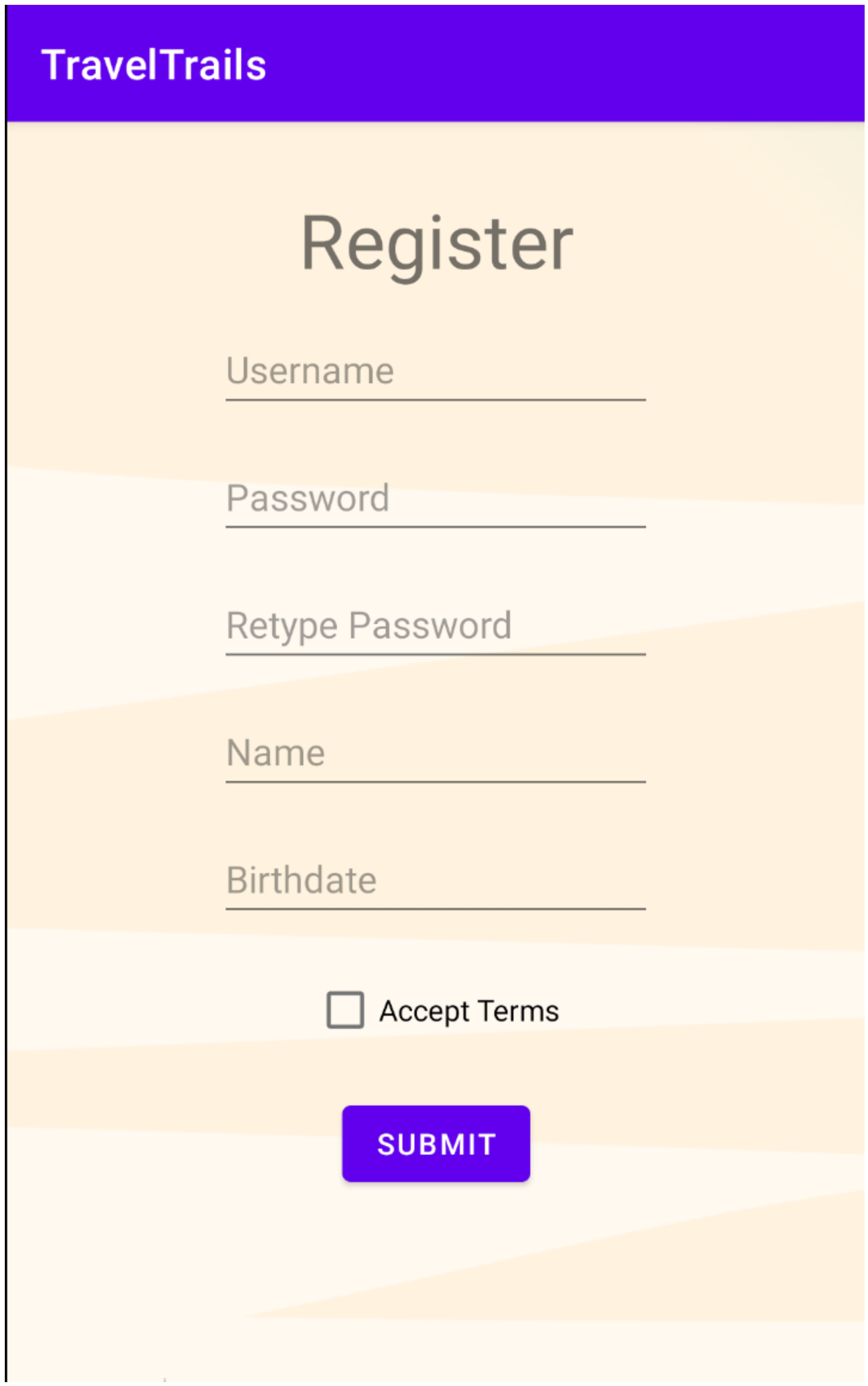
Travel Trails Flow Diagram

Mockups/
Wireframes

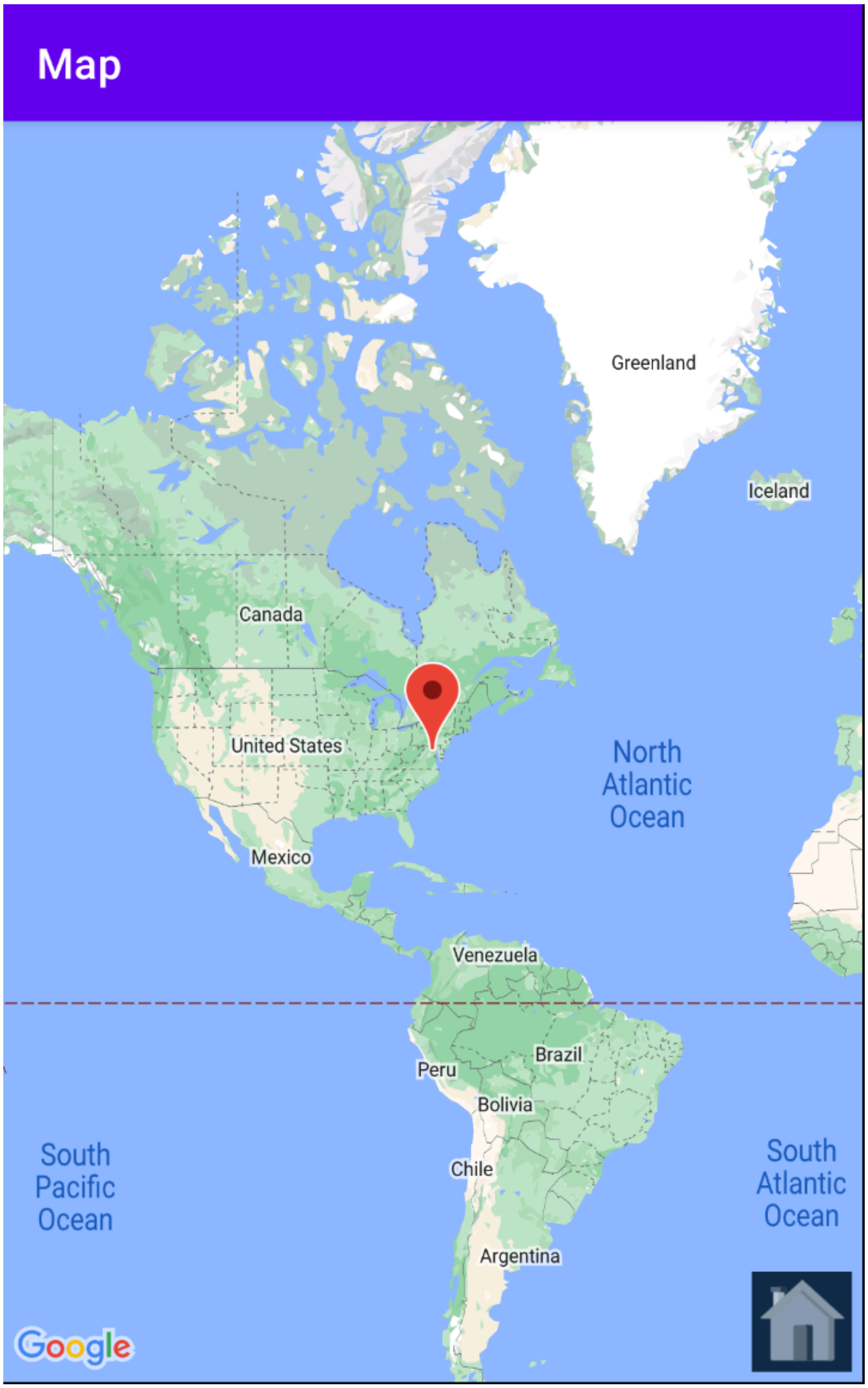
Login


TravelTrails
Travel Trails
Username
Password
LOGIN
REGISTER

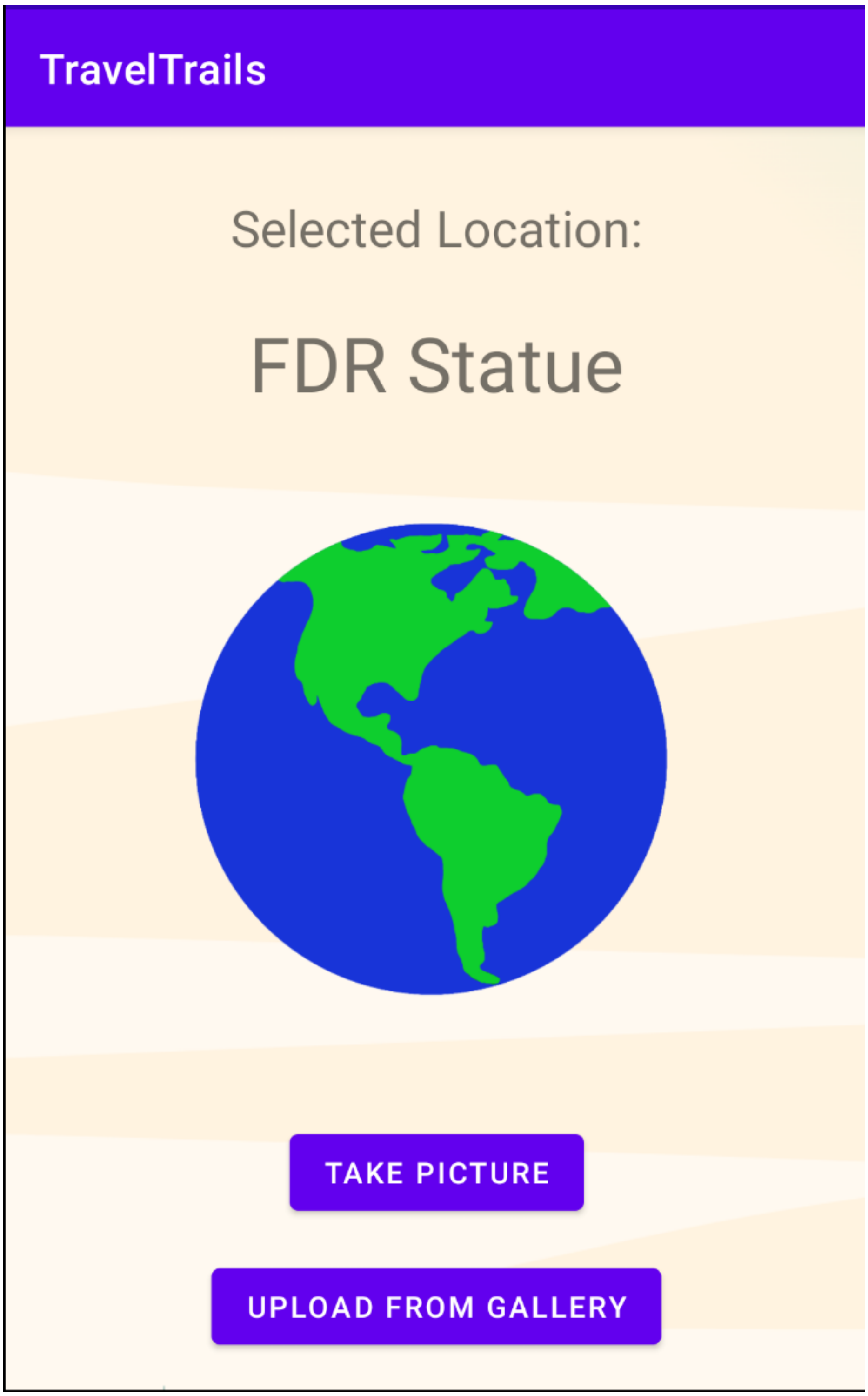
Register


TravelTrails
Register
Username
Password
Retype Password
Name
Birthdate
Accept Terms
SUBMIT

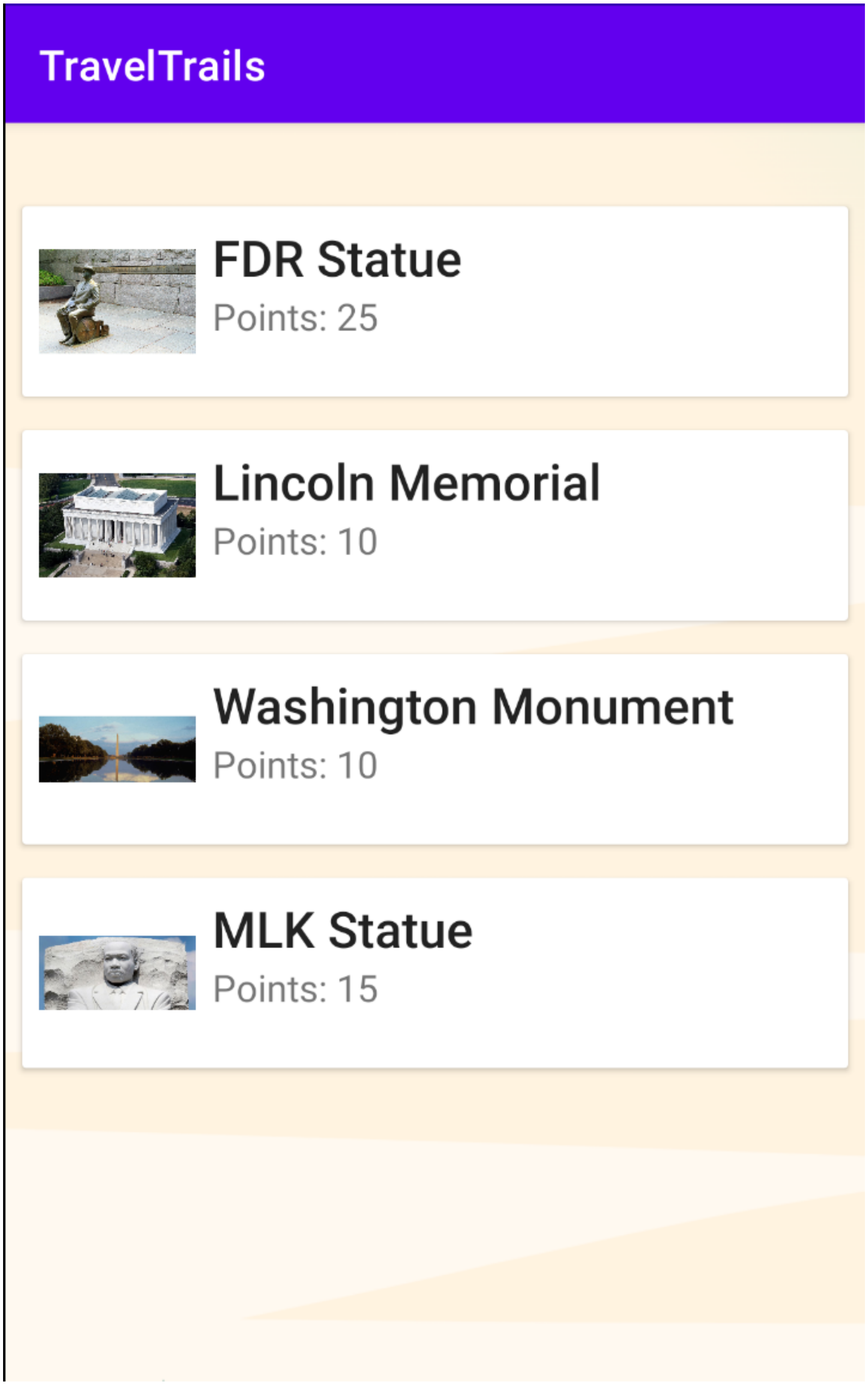
View Pins of Landmarks on Map



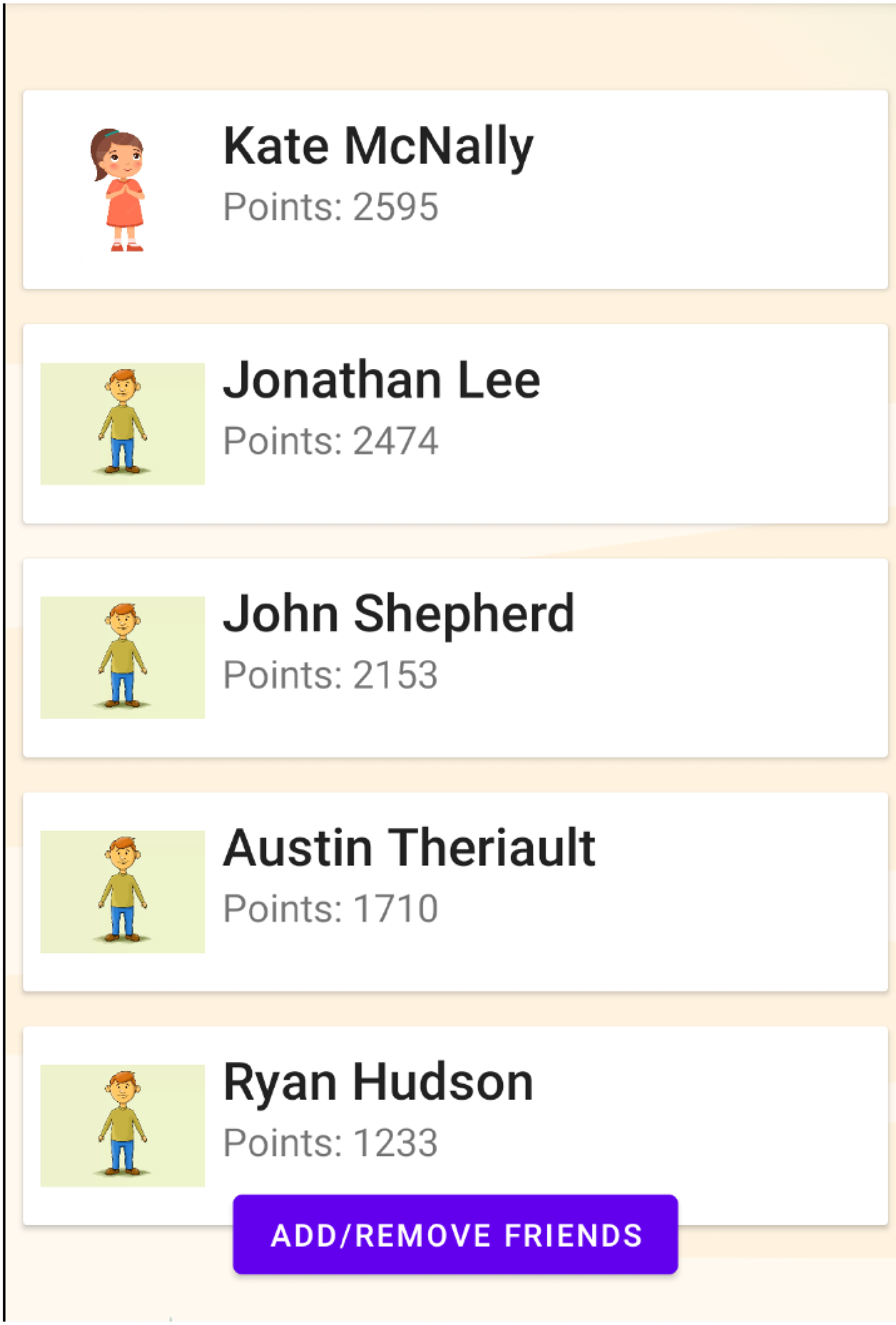
Select Location and Take Picture



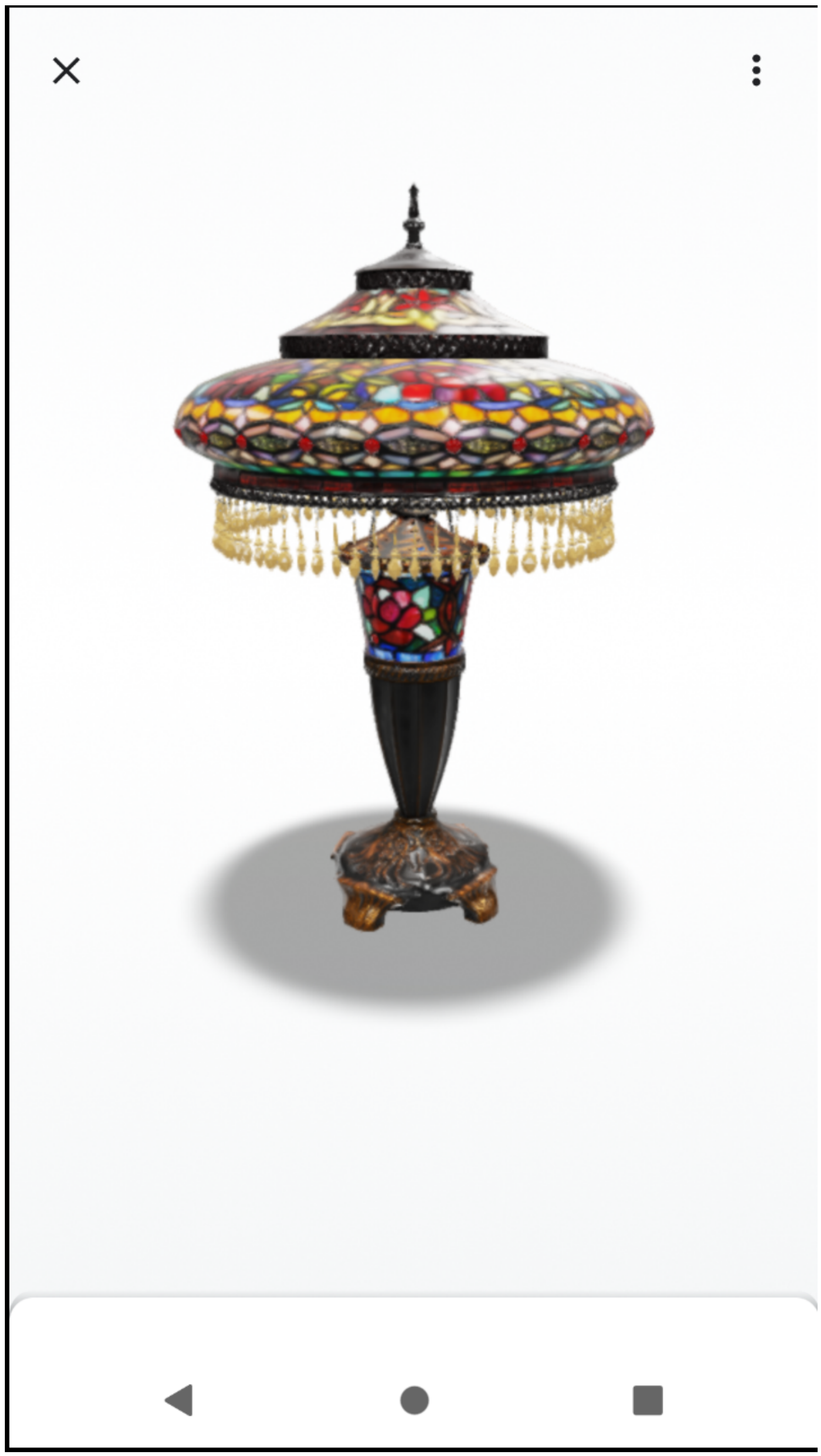
View List of Landmarks Visited in Gallery



View Friends/Leaderboard

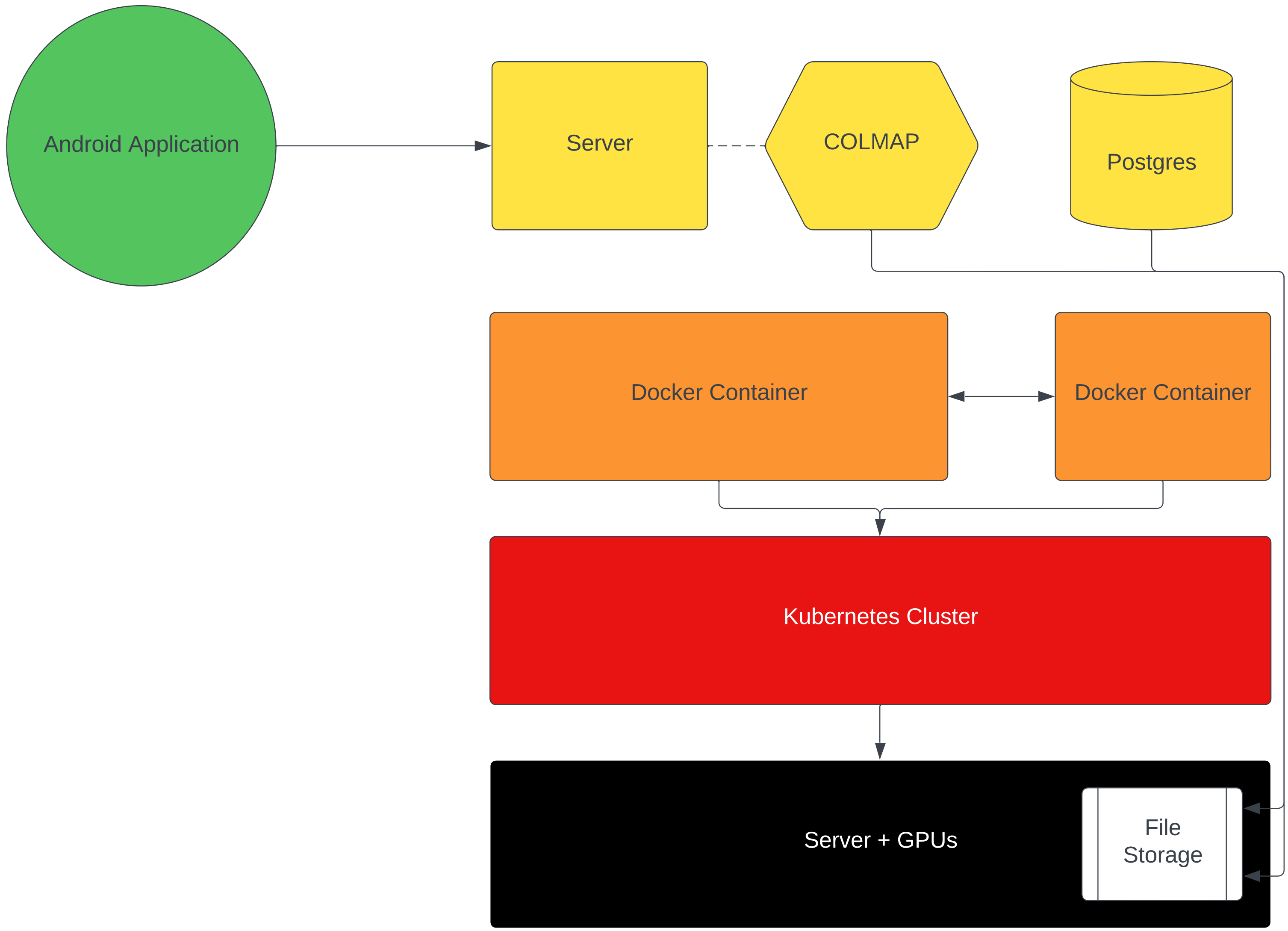


View 3D Models



Technical Specifications

Archiecture/System Diagrams



External APIs and Frameworks

Android Studio

- Goal
  - Develop the front-end application
- Description
  - Android Studio is the development environment for our app. The app is coded in Kotlin and Android Studio comes equipped with an emulator for testing purposes.

Camera

- Goal
  - Take pictures of locations
- Description
  - The camera is used in the CameraActivity file of the code. When a user clicks on a pin on the map, it will launch the camera for them to take a picture of the landmark. This picture will then be sent to our backend via a POST request in JPEG format for 3D reconstruction.
- Endpoints Used
  - POST /Upload

Google Maps

- Goal
  - Find locations for collecting 3D models
- Description
  - Google Maps is used in the MapsActivity file of the code. When a user logs in, they view the map which displays the available landmarks for 3D models. These pinned locations are obtained from our backend via a GET request.
- Endpoints Used
  - GET /Nearby

Google Scene Viewer

- Goal
  - Display 3D models
- Description
  - Google Scene Viewer is used in the GalleryAdapter file of the code. When a user clicks on a landmark in their gallery, it launches Google Scene Viewer from a 3D

model file in .gltf format, which will be obtained from our backend via a GET request using the name of the landmark.

- Endpoints Used
    - GET /Model

COLMAP

- Goal
    - Create 3D models from images
- Description
    - It is a structure-from-motion pipeline that creates dense reconstructions. COLMAP is a package and we have it running in our production dockerfile and import it in server.py. Since generating a model is an expensive, and time consuming task, new model updates will happen more periodically as new user images are processed in batches.

Postgres

- Goal
    - Postgres is used to persistently store data for the application
- Description
    - We have a database schema that includes locations, their associated model files, photos, descriptions, names, and location, and finally basic user information. The schema is in the tt-server/server/models folder, and is declared in the SQLAlchemy ORM format.

Docker

- Goal
    - Provide a consistent build and running environment for our application backend that can be easily deployed without reinstalling dependencies like COLMAP or Postgres everytime
- Description
    - We utilize two docker images, one with COLMAP preinstalled, located in tt-server/Dockerfile, that then runs the flask server, and another that runs the Postgres database instance, located in tt-pg/Dockerfile. This is all run in a kubernetes cluster defined in the top level tt.yaml.

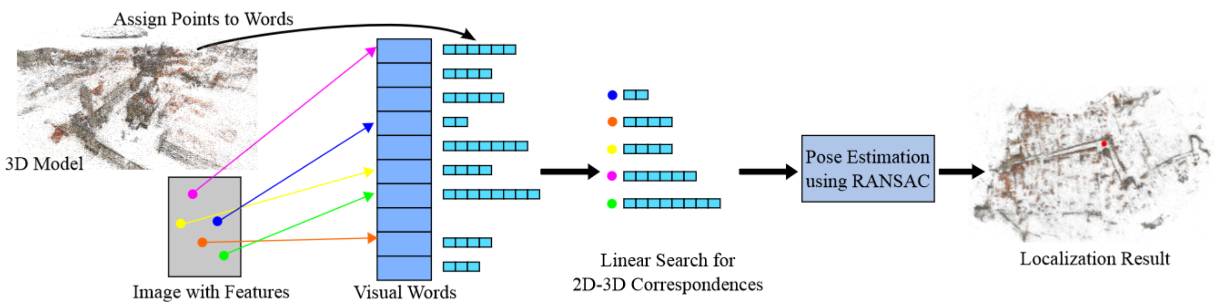
Flask

- Goal
  - Serve a REST API so the app can upload photos, they will be added to models, and the models are returned. Additionally basic user info, and info about the locations can be served easily
- Description
  - We use Flask to have endpoints for uploading a photo, getting nearby locations, info about locations, and downloading a model for a location. This server interacts with the Postgres using SQLAlchemy. The endpoints are located in tt-server/server/server.py.
- Endpoints Provided
  - GET /locations
  - GET, POST, DELETE /location
  - GET /location/images
  - GET /locations/nearby
  - GET, DELETE, POST /image
  - GET /image/data

Algorithms

Fast Image Registration and Localization

- Goal
  - Quickly verify that a user's image matches a model in order to provide them with some immediate feedback as to whether or not they have unlocked the model, the model update will be done in batches more periodically.
- Description



  - Adding a new image to an existing model through the 3D reconstruction pipeline is a time consuming process because of steps like bundle adjustment, an optimization task, which is computationally expensive and alters the existing 3D points of the model. Therefore, a separate localization algorithm is used to match the image matches to the model. We will be implementing a direct 2D-to-3D matching algorithm (from Sattler et. al) which quickly localizes an image to a 3D model. The algorithm works by finding a 2D-to-3D image correspondence to 3D points in the model. Image features are detected using standard feature detectors (SIFT or Harris Detector) that are vectorized into a visual words histogram model. This is essentially a short descriptor of the image that is compared against other descriptors in the model dataset, and the closest match is found using a linear search.