May 16, 2021

Milestone 4 - Team Ornithomimus

State of the Project

The Flutter application now has all features fully implemented. In addition to the features implemented in Milestone 3, the application now makes three API calls to the backend: posting preferences, posting reviews, and getting hikes. When the user switches from the profile screen to any other screen, an API call is made to post their preferences to the backend. When a user swipes left or right on a hike, an API call is made to post the review to the backend. When the user reaches the end of the stack of hike cards, an API call is made to get more new hikes for the user to see. On top of the API calls, the user can now see their matched hikes on the AllTrails website. When the user taps on a hike in the matches screen, there is a button on the hike details page which takes them to AllTrails. Tapping this button opens the hike's AllTrails page in a browser, where the user can see more photos, reviews, and trail guides for the hike.

In the backend, the two machine learning models have been connected; there is now a clear flow of information in between API calls for posting hikes and getting hikes. When the user posts a review for a hike, the API now updates the classifier dataset and trains both models. Then, the recommender system is able to produce a list of the best hikes for the user and sends it to the image classifier. The image classifier receives the shortlist, passes each hike through its model, and removes any hikes which are predicted to be a dislike. The resulting list of approved hikes is saved in preparation to send back to the application. At the next API request to get hikes, this list is sent to the mobile application, where hikes can be shown to the user.

The Flask API is now complete, and can send JSON objects back and forth between the frontend and backend. The RESTful API was created using Flask Python, and accepts POST and GET requests for three resources: hikes, reviews, and users. The API responds to each request with a response code and, in some cases, a JSON formatted string. The response code is used by the application to determine if the request was successful, and contains more details about why the response was denied if it was unsuccessful. The JSON object, in the case of GET requests, contains information about new hikes which the application can parse and display to the user for viewing and rating.

Feature Changes

In the final submission, we did not make any significant feature changes from the target product in the original proposal. However, we did omit two of the three stretch goals: displaying hike reviews in-app and filtering hikes by location. Neither of these features were considered essential to the success of the project, and we decided to focus our time and effort on more critical features. That said, we were able to display the location of each matched hike in-app, and we allow the user to reject hikes if they decide they are too far away. We were also able to implement a button which opens the AllTrails page for a specific hike in their browser, so the user has easy access to reviews and maps if desired. The final stretch goal, which we did implement, is persisting the user's preferences across sessions. Using Firebase, we saved the user's preferred hike length, elevation gain, and difficulty locally, which eliminated the need for tokens.

Current Challenges

The toughest challenge we faced on the way to this milestone was connecting the Flutter application and backend. During testing, we decided to host the python server locally so that we could quickly make changes. As a result, we had to run both the server and an Android emulator on the same machine; this was a tough ask for a single laptop to run, meaning testing was frustratingly slow. We also faced challenges trying to make the requests compatible from the Flutter end to the Flask API server. The seemingly same requests failed when sent from the Flutter application, but would succeed when sent from a Python program. In the end, we were not able to send API calls between the server and application; omitting this, all other components of the project are fully connected and functional.

Team Contributions

Andrew Forde

Contributions: Organizing and contributing to the final report. Recording video for Flutter application demo and image classifier, including voiceover. Unified testing with API calls between Flutter app and server. Code cleanup and bug hunting in Flutter app.

Taylor Mcouat

Contributions: Contributing to the final report and video. Reformatting the API server for compatibility with the Flutter application. Linking the image classifier model to the API server.

Leo Sun

Contributions: Final report and some initial testing preparation.

Jayden Wong

Contributions: Flutter application section in final report, video editing, and voiceover.