final-project-umbreon

Developers:
- Brandon Whitfield (bjw4ph)
- Ryan Coughlin (rlc4sV)

Device name: Umbreon
Platform: Android
Project Title: Gavel Guide

Project Pitch: Gavel Guide is a comprehensive, one-stop, debate-tournament management application. As a contestant, one can use Gavel Guide to find information about their upcoming pairings (debates). For each pairing, there is displayed the two teams and their side (Pro or Con), the presiding judge, and the location of the debate (complete with a picture of the building and location via Google Maps). Additionally, debates can be recorded and linked to a pairing within the app for the listening pleasure of all users. Judges can score speakers and submit ballots with a designated passcode. After ballot submission, scores can be viewed by all users in a separate "Results" tab. A third "Standings" tab displays the current record of each team. The data is stored in a Mongo database served by our own Node.js backend web service.

Platform Justification: Neither of us developers has a Mac, so developing for iOS would require commuting to the lab to use Xcode very frequently. This was not particularly appealing to either of us, thus, our decision to use Android was almost purely logistical.

Key Features:
- **View pairings -** Users can view all pairings in a list that displays the round, team names, and their side
    - **View pairing information -** Users can view the teams, sides, judge, and location of each pairing
    - **Submit ballot -** Judges can score debaters and submit ballots with their designated passcode
    - **Set recordings -** Users can record and link recordings to a pairing for later replay
    - **View building/View map -** Users can toggle between the location of the debate on Google Maps and a picture of the physical building
        - We utilize GPS (location manager) to get the latitude and longitude of the device current location to plot onto the Google Map
    - **Get Walking Directions** - launch Google Maps from within the app to get step by step directions to the location of the round
- **View results -** Users can view all results in a list that displays the round, team names, and their side
    - **View result information -** Users can view the winner and the speaker scores

    - **Play recordings -** Users can create recordings for a round and upload it to the cloud or download and play recordings associated
- **View Standings -** Users can view the standing of each time in a list that displays the team and their record
- **Recordings** - Uses device microphone sensor to create recordings for each round. Recordings can be uploaded to cloud-based AWS S3 Bucket using the Amazon SDK, as well as downloaded from this bucket using the same SDK
- **Web Service Implementation** - utilized a Node.js server with a MongoDB database to power our web service. The server was hosted on a t2.micro instance on Amazon Web Services. Code for the web service can be found at: https://github.com/bjw4ph/GavelGuide. Created the following endpoints:
    - getRankedTeamsJoin: used to get the standings of teams in the tournament to populate the standings tabs
    - getAllPairings: used to get all of the pairings to populate our current round and previous results tabs
    - addS3Key: used to save information regarding recording information to download recordings from S3
    - submitDecision: updates pairings entry with the results for a round,
- **Rotation** - Different layouts loaded based on portrait or landscape orientation of the device for main screen pages and view pairings. Additional Information displayed on the main page when rotated to landscape mode
- **Refresh data -** Dragging down gesture refreshes data
- **Store data in Memory -** Each API call stores data in local data
- **Load data from Memory -** If no network available, app loads data from local data if it exists

Testing Methodologies:
Tested all of the different functionalities of the device in all of the different conditions which we could think the device would be subjected to. These conditions include on device rotation, when the internet is not available, when operation is interrupted in the app, when the permissions for the app is not granted.

Usage:
Navigation of the app is hopefully pretty intuitive and doesn't require any special actions

Submitting a Ballot:
Brief Explanation of Debate to understand what you are actually putting on this ballot: Pairings in the form of debate we were trying to model consisted of two teams, Pro and Con, which either defend or attack a resolution for the round. A team consists of two debaters, who are assigned scores at the end of the round in order to judge the quality of their respective speeches. A score of 24 is for a complete beginner and a 28 is for the best speech of the year. The judge also determines who won the round based on who convinced them better

The passcodes to submit a ballot is ‘star’ + the number of the judge (e.g. Judge5’s passcode is ‘star5’)

Other useful tips
To reset the database (if you’ve no more ballots to submit), visit this URL:
http://ec2-54-145-200-199.compute-1.amazonaws.com:1234/setUpMongo

Lessons learned:
The lessons that we have learned throughout this Android project are several.
Web Services and AWS are super cool and pretty easy to set up. We were able to set up our EC2 instance with relative ease once we constructed the Node.js server. The documentation is great and Amazon is nice enough to give a free tier for us to use as we learn more about cloud based computing. Also, S3 is super powerful. Amazon provided a ton of example apps which made implementing the technology pretty easy. This project was a great introduction to setting up services in the cloud in the future through AWS

Formatting and good UX design can be a real pain. Gaining all this knowledge from this semester of good UX design and more specifically implementing Google’s material design. We were barely even able to scratch the surface on those requirements, and formatting still took a considerable amount of time. There are so many small things which you never even noticed before, which take a lot of time to implement in a solid UX design. All the little things which make an app look really sleek are usually features of a device which take the longest time to construct.

Google does a good job ensuring proper sensor usage. We needed several permissions for our app to utilize current location for the map and the microphone for the audio recordings. It was cool to see how you check user permissions and then create popups to ask the user for permission. We now know an app can’t use these critical sensors without a user’s permission which can give us a little peace of mind.

Finally, we simply can’t say it enough - Git is good! However, it is only as good as the users. We learned that using good branching strategy is essential for efficient collaboration and code integration. By working initially in separate feature branches, it was easy for us to organize code and allowed us to work in multiple different directions simultaneously. Opting for intermittent pull requests to the master branch instead of continual commits saved us a lot of headaches and needless conflicts.

Citations:
https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en

https://developers.google.com/maps/documentation/android-api/start#get_an_android_certificate_and_the_google_maps_api_key

http://docs.aws.amazon.com/mobile/sdkforandroid/developerguide/s3transferutility.html

https://developer.android.com/guide/topics/media/audio-capture.html