REDKK Project Ideas

Contributors: Reid, Hengyuan, Jianchao He, Kyran, Zhiyuan

* **LOL Esports Dashboard**
  + **Description:**
    - A web application serving for League of Legends (A popular PC video game) esport-lovers. Providing latest information for esport teams, matches and schedules. It supports highly-customoized data dashboard and standing visiualization for users who would like to dive deep into their favorite teams and leagues.
  + **Features:**
    - **Schedule, Vods and Standings**: Our overall dashboard provide insights with latest LOL e-sport statistics and news. Various filtering conditions are supported from users’ end for customized information inquiries.
    - **Match History Tracking:** Track the historial match record between two teams for any incoming match. Giving out winning rate prediction based on the match history and contestants‘ recent form.
    - **All-in-One interface:** All related information and links are intergrated together for users’ easy navigation. Beginning from a single match history, users are able to be redirected to the detailed player stats, match vods, team info and so on. Thorough support on subpage navigations gives users best exploration experience and lower use cost.
    - **Customized Watching List:** Pick your favorite team and players then store them in our cloud servers. Login with your personal account to retrieve the search history on any portable devices. You can also mark your favorite team member, we will prepare his personal highlights for you.
    - **Forecast the result of the competition:** Pick your two favorite teams, and our overall dashboard can provide predicted results of the game through our database and calculation methods, such as the results of the previous games of the team you selected, and the matchup of the participating players will be considered.
  + **APIs:**
    - [API for LOL Esport Standings](https://rapidapi.com/snldnc-kpCtDKbxo_F/api/league-of-legends-esports)
    - [API for LOL Game Info](https://rapidapi.com/snldnc-kpCtDKbxo_F/api/league-of-legends-champions/)
* **RelativityRadio/Trailblazer**
  + **Description:**  A web app displaying a location map of songs listened throughout city, able to be viewed, filtered, and queried according to location, time, and song information.
  + **Main Features:**
    - *Music Map:* visualize which songs were listened to when & where! (as well as other stuff based on [spotify data](https://research.atspotify.com/datasets/).)
      * *Track Tracker:* see locations where a specific song has popped up
    - *Local Listening*: compile songs from within a space-time radius of a into a static playlist.
    - *Dynamic Queue*: filter, weight, rank, and auto-play nearby songs based on specific criteria! Hear the songs played by others around you.
    - *Anonymous Chat:* say hi to an unknown nearby user!
      * *User Link*: create an auto-updating shared playlist!
      * *Song Pong*: interlace queues! (song choice alternates)
  + **Permissions:**  Users of this web app can link their Spotify account (using OAuth) and grant location permissions (possibly using a mobile app, or via HTML5).
    - Private Zones: Users can specifiy certain areas or timeframes within which data will automatically stop being shared. Definition of private zones will be part of the default registration process.
    - Obfuscation: user IDs will not be stored on the frontend, and only accessible on the backend. A rate limit (~60 tracks/minute) will apply to queries on "trails" of next- and previously-played tracks, preventing large-scale data harvesting of location patterns.
    - Location inaccuracy: geotagging will have some (transparent) amount of noise introduced purposefully—users can specify accuracy range betwen "within 5 feet" and "within 100 feet" (relative to user observation). approximate distance will be given for tracks,l or coordinates will be approximate (rounded to the nearest 50 feet, or something similar.)
  + **Data Management:**  the app will automatically approximately geotag and timestamp each track as it is listened to, and upload the record to a non-relational serverside database along with (anonymous) user ID. Each data object will also contain links to the next and last song played, as well as other optional flags (such as whether the user was listening to an album all the way through or what playlist the user was listening to).
    - *Validation*: if the interval between start and end timestamps are not at least half the length of the track, the entry will not be logged (presumably the song was skipped).
    - *User Control*: any and all records created by a specific user can be deleted at any time, or permanently dissociated from any user account and kept on the server for records purposes.
    - *Comments*: users can enable or disable comments on their tracks; user-dissociated tracks always have comments enabled.
  + **External Data Streams:**
    - map data (Google)
    - general listenership statistics (Spotify)
  + **APIs & frameworks:**
    - [SpotifyAPI](https://developer.spotify.com/documentation/web-api/)
    - [Google Maps API](https://developers.google.com/maps/apis-by-platform)
    - [HTML5 Geolocation](https://crate.io/blog/geolocation-101-get-users-location)?
    - [Anonymous Chat](https://github.com/MRHRTZ/Anon-Chat/blob/main/README.md)?