Alex Trahan, John Popich, Osayd Esmail

03 March 2022

EndGameMaps Specifications

User Stories:

- *Users Flow:*

  Overall Flow:

  Upon first opening the webpage, all users are faced with a homepage screen. The intro/homepage screen will display information regarding the app such as its uses, goal, etc. Perhaps a section for displaying some sort of user data. This page will allow for creation of an account, which verifies the users location for determining user's locality. Using the Geolocation API If a user wishes to just view the app without an account, then they will default to be 'guest user' when in the mapview. (Guest users have the same permissions as visitors/tourists in any given location until account creation, MINUS ability to create pins).

  Once UserA creates an account and location is verified, UserA will then unlock the visibility of pins that other locals have dropped (within their local region of course).

  After account creation, as well as guest users joining the mapView: The page will then render the map centered around that user's

location, with other local users' pins being perhaps green, the visitors/tourists pins being purple, and friends' pins being gold. There will be a main web navbar that will return to the homepage, view the 'Friends' tab, or Settings.

There will also be a menu for the map itself, displayed on the left side for familiarity between apple & google maps. Options here include a "Find me Something" button(returns a random pin within x distance to do), "My Pins" section (shows user's placed pins with modify/delete permissions), "Create Pin" button (which brings up pin creation menu), and Pin Visibility Settings (checkboxes for locals, tourists, friends, and their activity category).

Users now will be able to view pins on the map in their respective locations (for reference I'm thinking size of New Orleans). The pins will hold information regarding:

1. Pin Name
2. Location
3. Activity Category (Niche* Oddities* CoolArt* or HaventSeenThis*) So perhaps if we limit our Category list to a certain few checkboxes, we can filter out duplicates almost naturally. These

categories though will take some verbose language to cast a net on all the things we want vs. don't want.

4. Timestamp (future)
5. (Optional)Possible items to bring?
6. (Optional)The experience of the user who placed the pin
7. (Visibility…I'm leaving it for now, but on the fence about the implementation.)

Once a user selects a pin and reads all info regarding it, we should implement a way to feed the location data into apple/google maps for directions to the location.

//under review

Or alternatively if a user has an experience/place they'd want to share they will be able to add a pin. Pins also have a hidden property which maintains visibility to those in the selected group the user wishes to allow visibility to (tourists? Friends? Locals?). Think like whitelists & blacklists. //

Regular User:

Regular users will start at homepage, log in, and be redirected to the mapView.

Admin Users:

Admin users will only be those users whose usernames are on a whitelist for admin privileges. These would be in their own div rendered server side, and include; modify/remove pins, …, etc.

Guest Users:

Detailed above, but limited visibility as a tourist to everywhere. Also no publishing permissions.

Frontend Features:

- Homepage (almost exclusively frontend)
    - Welcome user to webpage
    - Display app name, explain where name comes from
        - Natural segway: now explain uses of the app and what to expect
    - *(maybe display some fun data about user experiences so far?) - backend grab of json of random pin*
    - Login/Create Account

- Login/Create/Recover Account (which interfaces with backend)
    - Login (default view):
        - “Username or Email”
        - “Password:”

        - “Submit” → send request to backend & await response
        - “No account?” -> Create Account
        - “Forgot user info?” -> Recover Account
    - Create Account: https://rapidapi.com/collection/profanity-filter https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API/Using_the_Geolocation_API
        - “Enter a username (this will be public)”
        - “Email:”
        - “Password:”
        - “Submit” → send info to userinfo on backend
    - Recover account:
        - “Enter email associated with account:”
        - *captcha*
        - “Submit” -> endpoint here for server to catch
- Main(map) view (displays all current pins from db)
    - Upon entering:
        - Focus on user’s geolocation gotten from backend geolocation api
        - /*stubbed Show all local/tourist pins for all pins within 20 min drive/ 15 miles https://docs.traveltime.com/api/overview/introduction?ref=apilist.fun */

        - Select three closest pins and display their information as if they were clicked by user (this will give them an idea of the overall UI and examples of pins)
    - Map's Menu:
        - Display on left side for familiarity between apple & google maps
        - "Find Pin" → Returns a random pin within x distance to do or get data on clicked pin
        - "Sort" → Displays a little list of all pins placed by the user. Upon click: zoom to location of pin, show pin details, and option to edit/delete pin.
        - Create Pin button: https://rapidapi.com/collection/profanity-filter
            - Brings up PinCreationMenu
                - Pin name: (name user provides for pin)
                - Pin location: (coordinates or address)
                - Pin Activity Category: (checkbox) (Good with groups? Adventurous? Relaxing? Cost?)
                - Pin Visibility: (checkbox) (Friends, Locals, Tourists)
                - (Optional extra info):
                    - Need any items for the best experience?
                    - What is one thing you remember from this excursion that was memorable?
                - Submit → Sends all info to profanity filter, then to backend, which sends to db

        - Can sort visible pins (checkbox) based on local pins, friends pins, or tourist pins.
        - Can sort visible pins (checkbox) based on activity categories
        - (Perhaps to combat pin overlap, stack them ontop eachother)
        - (Maybe limit amount of pins to place?)
            - Pin currency
        -
- Main Navbar:
    - Home, view the ‘Friends’ tab, or Settings
    - Home:
        - Go back to welcome page
    - Friends tab:
        - Shows current added friends if any
        - Add friend option (send request to DB, DB sends request to other user)
        - Incoming Friend Requests (show all friend requests)
    - Settings tab:
        - Account settings:
            - Change email?
            - Change pw?
            - Change username?
                - **Extra verification needed here**
                - Maybe a phone number or security questions?

        - Logout:
            - Logs user out
            - Returns to homepage

Backend Features:

https://docs.traveltime.com/api/overview/introduction?ref=apilist.fun

This would also be a great addition for finding things to do near you!

/*stubbed

- Cookies:
    - If we implement cookies, I could see it being the most useful in verifying user locations, checking if they're still in their local region per website access. As much redundancy to maintain correct visibility of pins to clients.

*/

- AccountManagement
    - Listen for endpoints:
        - login/submit/
            - Do something
        - createAccount/submit/
            - Do something
        - recoverAccount/submit/
            - Do something
- Pin Management
    - Listen for endpoints:
        - mapView/findRandomPin/Button/

            - Retrieve random pin from db within 10 miles of user
        - mapView/myPins/button/
            - Retrieve user's placed pins
        - mapView/createPin/button/Submit/
            - Get Pin data from client (pinName, pinLocation, pinCategory, pinVisibility, itemsNecessary, userMessage)
            - Send to DB, update all users' map with new added pin
- APIs that will be helpful that have not been linked:
    - https://rapidapi.com/dev.nico/api/ai-powered-content-moderator/details
    - https://rapidapi.com/community/api/purgomalum-1/

Database Features:

- Determine all the app features that the database is responsible for. Define those features.
    - As a start, we can have the user data (username, password,email, user's IP address), the pins that each user created, each user's list of friends
    - The pins' data: pin name, pin location, activity category… the creator of the pin
    - Admins' data: email, username, password…

- DB & Server (Backend) communications:
    - /mapView
        - Retrieve all pins from db, send to client from server

- mapView/createPin/button/Submit/
    - Post new pin w its info to DB from server
- mapView/myPins/button/
    - Retrieve that users list of placed pins (Maybe this could just be exclusively frontend) from server
- mapView/findRandomPin/Button/
    - Get random pin request from server, give random pin within R radius of user
- createAccount/submit/
    - Add new account to db from server
- recoverAccount/submit/
    - Get user info from email requested from server
- mapView/friendsMenu/
    - Get user’s addedFriends & user’s pending friend requests from server request