

README

CodeOfDuty

Summary

GeoXplorers is a web Application designed for users to discover points of interest by using Google Maps combined with our search filters. The main technologies that were used are ReactJS, Django, SQLite. Users can search for specific keywords and apply filters to refine their result!

To enhance the user experience, GeoXplorers offers user registration and login functionality to make an account, allowing logged in users to save their searches and access them at any time. They can also receive real-time notifications about newly added points of interest that align with their saved searches, ensuring they stay informed and up-to-date.

GeoXplorers gives the capability to the website's staff to contribute to the platform's content. Staff members can add new categories and new points of interest by just uploading TCV files with all the required information.

With its minimalistic interface, search features, personalized saved searches, and seamless integration with Google Maps, GeoXplorers offers a unique and user-friendly platform for users to explore and discover new exciting locations.

Note: Both the Backend and the Frontend require an .env file, one for the Mail Sender credentials and the other one for the Google Maps API Key, respectively.



Team Roles

- Project Manager
 - Nikolaos Pnevmatikos (ID: 1115201900157)
- Backend
 - Nikolaos Pnevmatikos
 - Ioannis Kalesis (ID: 1115201900069)
- Frontend
 - Nikolaos Pnevmatikos
 - Maria Diamanti (ID: 1115201900052)
 - Eleftherios-Efkleidis Stefanou (ID: 1115201900176)
 - Aikaterini Milioti (ID: 1115201900112)

Back End

- The backend of our application has been built following the MVC design pattern
- For the Controller aspect we use the Django Views
- For the View aspect we use Django Serializers
- For the Model aspect we use Django Models
- The business logic of the application exists in the models, with an example of the Search object having the methods findNewData() and findMatchingLocations(), separating the search logic from the Controller (The search being the main business aspect of our backend application)
- The search is made by implementing a Query builder, and has a range of different resulting queries depending on the combination of search parameters.
- Exact details on how the search logic functions are explained below:
 - If all of "latitude", "longitude", "kilometers" parameters exist then the results will be limited by whether or not the point of interest is inside the circle defined by the parameters.
 - If the "categories" parameter is present, then a location will be included in the final result only if at least one of it's categories matches the input category list
 - Likewise for the "keywords" parameter, if it is present, then a location will be included in the final result only if at least one of it's keywords matches the input keyword list
 - If the "text" parameter is present, it is always applied to the "title" and "description" attributes of the locations, and a location must also satisfy either one of the above to be of match.
 - In the case that no input categories have been given, then the "text" parameter is also applied to the categories.
 - Similarly for when the input keywords is missing, the "text" is applied to the keywords attribute
 - The previous two mean that in the event that no categories and no keywords are present, the "text" parameter can satisfy either one of the "title", "description", "categories", "keywords" attributes of the location.

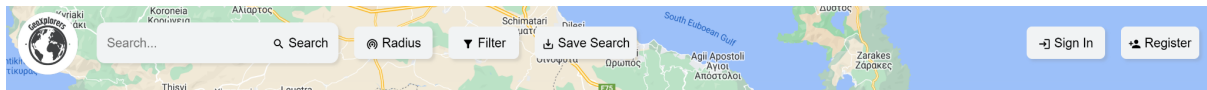
- The matching for the "text" parameter works with by whether or not the attribute we match against contains a substring matching the "text"
- i.e If one location has "title": "Lake Dunmore" and another has "categories": ["big lakes", ...] then both will match the "text": "lake" input parameter.
- For the REST API, we have tried to minimize the amount of endpoints present in our application, by making full use of the HTTP methods GET, POST, PATCH and using only one endpoint per resource where it is feasible.
- We have placed a very large emphasis on the request validation for the backend, making sure that every input parameter is of the correct type and that no unauthorized access can occur on an endpoint.
- Endpoints have been built under the assumption that anyone can send anything, including bad actors that will directly hit the endpoints with ill-formatted requests to crash the server or try to get access to resources that belong to other users.
- For testing the REST API, we have performed numerous integration tests by the use of the POSTMAN tool.
- After making sure that the backend works correctly on its own, we have performed system tests using the finished frontend.
- On the subject of logic and validations errors, we raise a ValueError (the corresponding IllegalArgumentException in Java) accompanied with a detailed description of why the request was rejected, which is then displayed to the user.
- This allows the user to understand what went wrong, and either assist him in fixing his mistake, or allow him to submit a bug report in case the error wasn't his fault.

Front End

A complete description of the main Components and Screens can be seen below :

Header:

Navigation bar for users that aren't logged in:



Navigation bar for users that are logged in:



Navigation bar for users that are logged in and have admin privileges:



Here's a breakdown of the component's features:

- Clickable Logo: The logo navigates the user to the Home Page.
- Search Bar: The component includes a search bar where users can enter search terms.
- Search Button: Submits all the search options (text, categories, keywords, radius).
- Radius Button: Allows the user to define a search radius.
- Filters Button: Opens a Modal with the Categories available and the an Input that the user can use to add keywords to their search
- Save Search Button: Only available for logged in users. Used to save a successful search that can then be viewed in the Saved Searches section on the Profile tab.

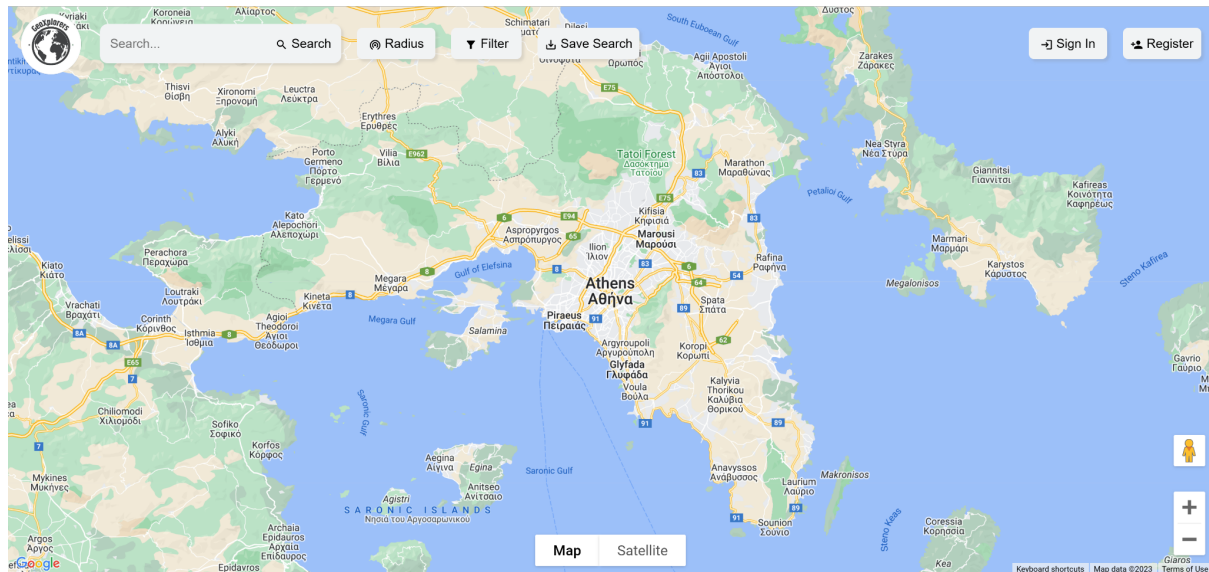
And lastly if a user is logged in, their account-related options are displayed in the navigation bar. These options include:

- a Notification Button to display the notifications (more details can be found below).
- an Admin Page Button (appears only if the user is staff).
- an Account's Dropdown Menu with options such as
 - "Saved Searches" Button, for displaying the user's saved searches and
 - "Profile" Button, (wasn't implemented) and

- a Logout Button, for logging out of an account.
- If the user is not logged in, the navigation bar displays
 - the "Sign In" Button and
 - the "Register" Button.

Home Page:

The Home Page display:

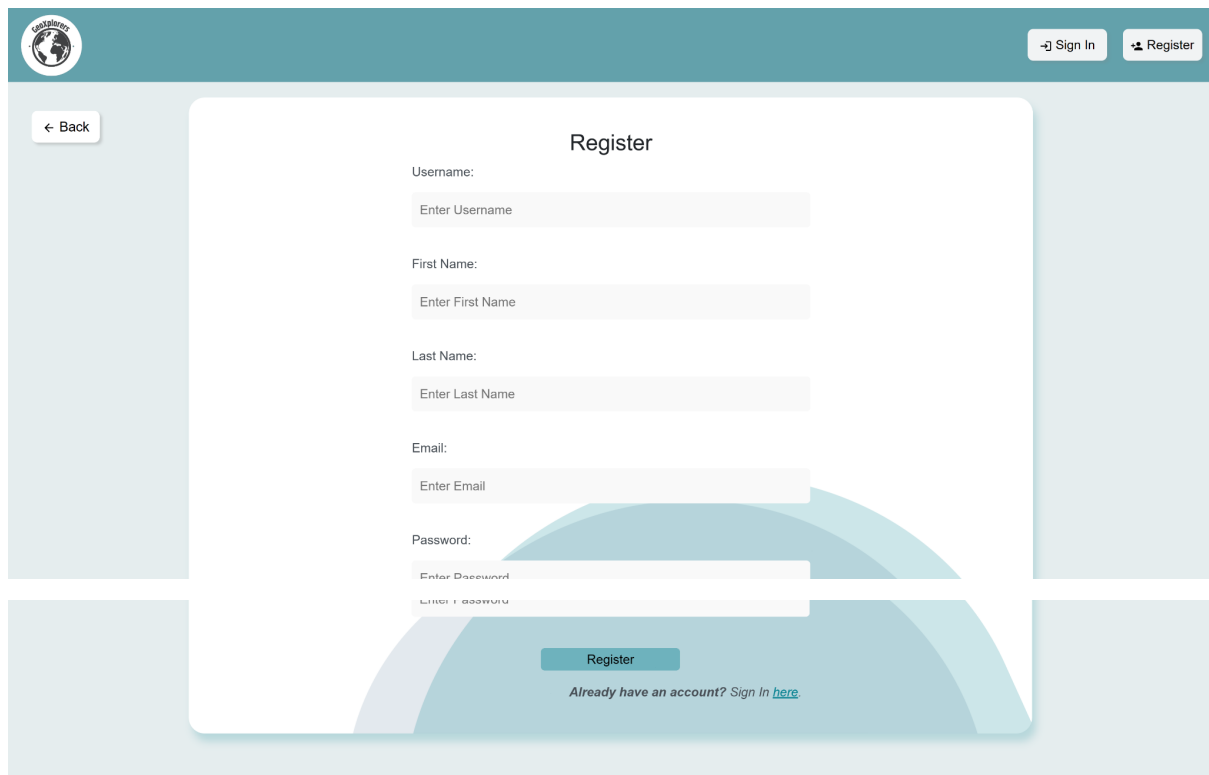


- This page contains the core functionality of the web application.
- A user that is not logged in has the same privileges on the search functionality with a logged in user, apart from the fact that he does not have the ability to save his searches.
- A user can search using only text or categories or keywords or radius, or a combination of any of the above. Considering the rest are self-explanatory, the radius creates a circle with a default distance around the user. The user can then expand or shorten the circle around him.
- The Search API works in a way that the user can change the parameters and the request is still being called using a requestBody. This makes it easier for a user to share a search to a friend.
- The Markers that appear after a successful search showcase the title of the point above them and by hovering above them you get to see the description of that Item.

- It is worth noting that the whole functionality has been built around the Google Maps API.

Register Page:

The Register Page display:

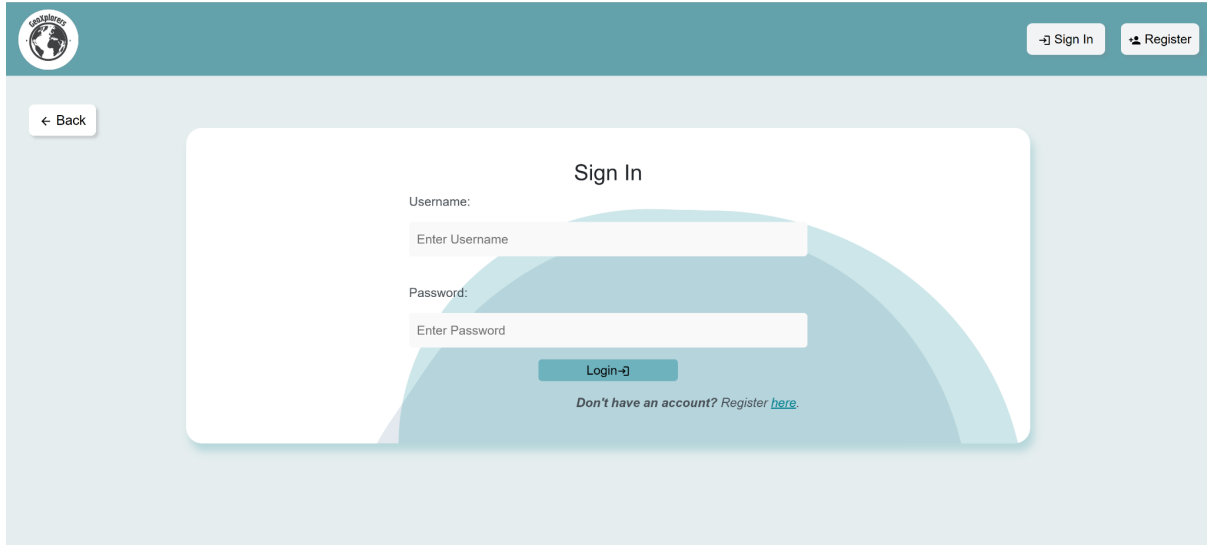


The screenshot shows a web application interface for a registration page. At the top, there is a teal header bar containing a circular logo on the left and two buttons, 'Sign In' and 'Register', on the right. Below the header, the main content area has a light blue background. On the left side of this area is a 'Back' button. Centered is a white card titled 'Register'. Inside the card, there are five input fields with placeholder text: 'Enter Username', 'Enter First Name', 'Enter Last Name', 'Enter Email', and 'Enter Password'. Below these fields is a teal 'Register' button. At the bottom of the card, there is a link that says 'Already have an account? Sign In [here](#)'.

- The Register Screen allows users to register by providing their username, password, first name, last name, and email.
- When the registration form is submitted, the register function is called. It sends a POST request to the user registration API endpoint using Axios, saves the user data in local storage, updates the user context, and navigates the user to the home page upon successful registration.

Login Page:

The Login Page display:

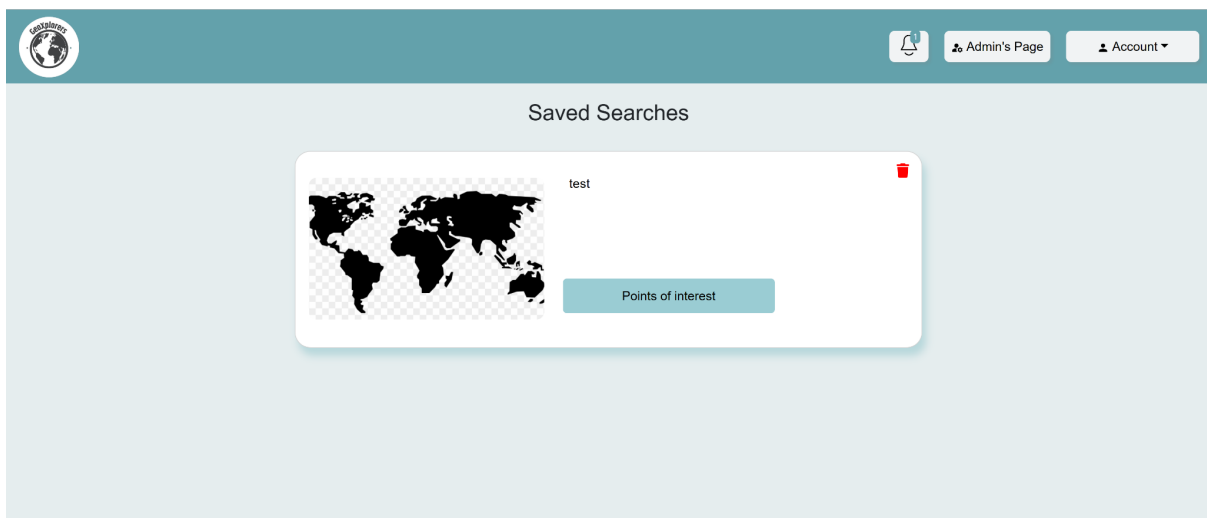


The screenshot shows the Login Page of an application. At the top, there is a teal header bar containing a circular logo on the left and two buttons, "Sign In" and "Register", on the right. Below the header, on the left, is a "Back" button. The main content area features a white card titled "Sign In". Inside the card, there are two input fields: "Username:" with a placeholder "Enter Username" and "Password:" with a placeholder "Enter Password". Below these fields is a teal "Login" button. At the bottom of the card, there is a link: "Don't have an account? Register [here](#)."

- The Login Page allows users to log in by providing their username and password.
- When the login form is submitted, the login function is called. It sends a POST request to the login API endpoint using Axios, sends the provided username and password in the request payload, and handles the response.
- If the login is successful, the user data is stored in local storage, the user context is updated, and the user is navigated to the home page. If the login fails, an error alert is displayed.

Saved Searches Page:

The Saved Search Page display:



The screenshot shows the Saved Searches Page. The header bar is teal and contains a circular logo on the left, a bell icon, and two buttons: "Admin's Page" and "Account". The main content area is light blue and titled "Saved Searches". It features a white card with a world map icon on the left, the text "test" in the center, and a teal button labeled "Points of interest" at the bottom right. A small red trash icon is located in the top right corner of the card.

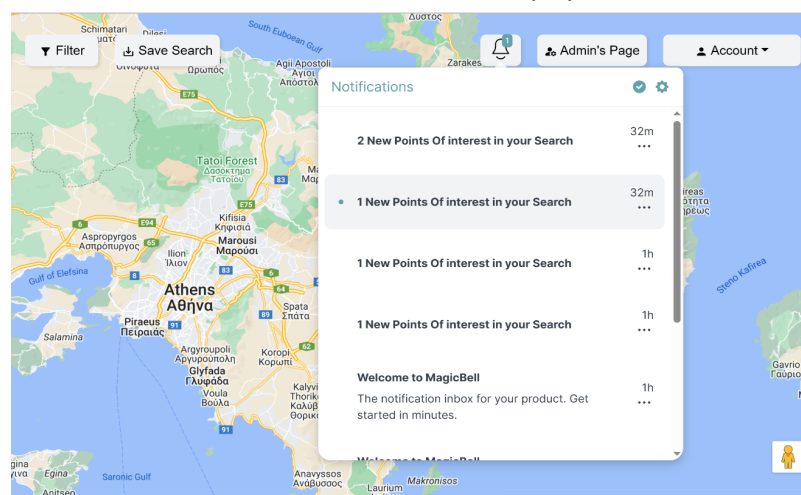
The Save Searches Page is a feature that allows users to delete and view their saved search items.

- The component displays the user's saved searches as cards with summarized details. Users can expand/collapse the text, navigate to item details, and delete searches
- If the user clicks on the button or the image of a saved search, they are being navigated to the Home Page that now displays that search.
- Visual feedback includes loading skeletons and toast notifications.
- If no searches exist, a message with an image is shown, instead.

Notifications:

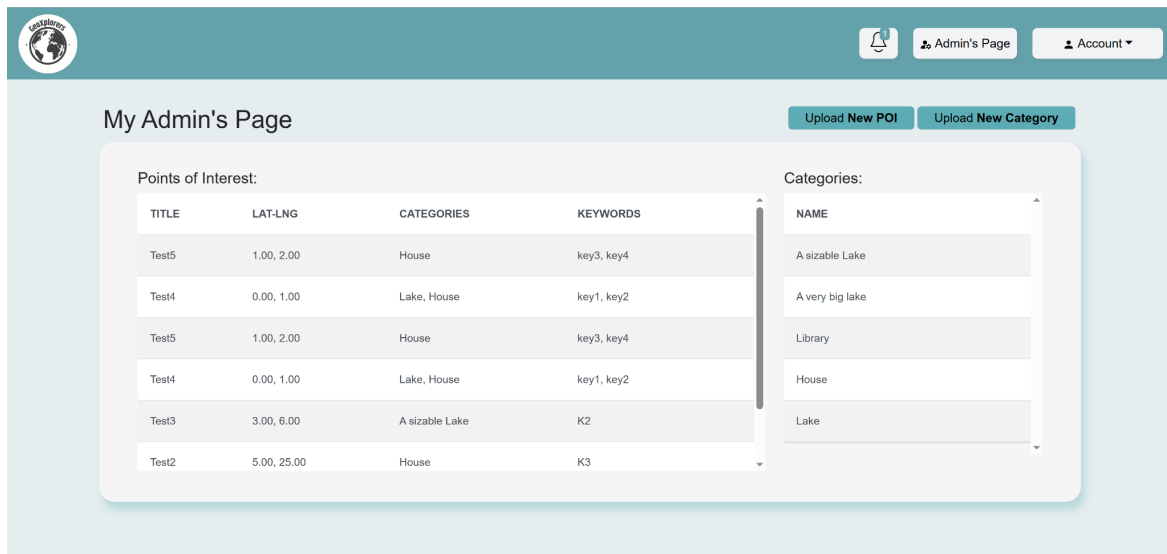
- Users receive notifications with new points of interest from searches they have saved in real-time and they receive an email regarding the same notification.
- Users have access to a notification inbox where they can view and manage all their notifications in one central place. The inbox provides a comprehensive overview of new and past notifications. The built-in notification functionality works only for logged in users and can be viewed at the top right of the screen.

The notification inbox display:



Admin Page:

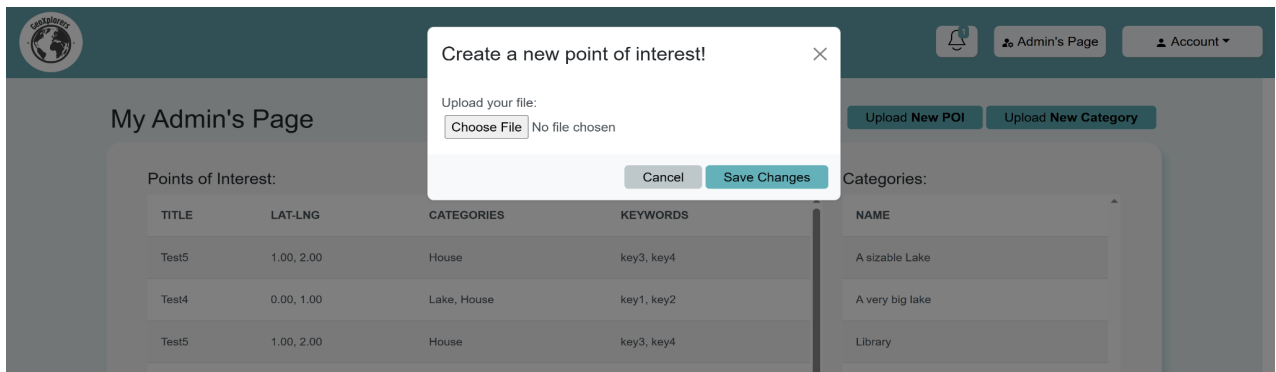
The Admin Page display:



The Admin's Page allows the staff users (users with admin privileges) to perform various administrative tasks related to points of interest and categories, allowing the staff users to manage POIs and categories by uploading new data and viewing the existing data. More specifically:

- Upon component initialization, the component checks if the user is logged in and they have admin privileges. Although the admin page button only appears to users with admin privileges, we wanted to ensure that even if someone with no admin privileges finds access to the admin's page url, then that user will be **redirected to the homepage**.
- The component retrieves the list of POIs and categories from the API using the `getPois()` and `getCtgs()` functions respectively. The API requests include an authentication token to ensure that only authenticated admins (`is_staff == true`) can access the data.
- The component includes two modals, one for uploading new categories and another for uploading new POIs. The modals are shown and hidden based on the user's interactions with the corresponding buttons.
- The Admin Page allows users to upload **tcv files** containing data for creating new categories or POIs. The uploaded file is sent to the corresponding API endpoint using the `saveFile()` function.

The Admin Page display when prompted to create new POI:



Upon successful upload, the modals are closed and a success message is displayed.

- The component displays a table of POIs and categories, showing relevant details such as title, coordinates, categories, ect., so the user can look at the already uploaded POIs or categories and upload accordingly making the user's experience easier and pleasant.