Aung Paing Soe, Kyaw Soe Han, William Ngo, Trong Le

Professor Abucejo

CS 160, Section 01

15 May 2025

Final Project: Description/Requirements

Feature Descriptions

Feature 1: Device Suggestion During User Input

To make sure that the device that the user selects to find a fix through the application is actually a valid device, the input is sanitized by calling a GET API request to the iFixIt API. This API call returns a list of categories that are available on iFixIt. Then, after filtering what the user has input thus far, a suggestion box will appear that allows users to choose their device of interest. This would help the application recommend clear and correct information to the user without any confusion if the user accidentally misspells a device or makes up an imaginary device.

Even with the API call, there is potential of there being a fault/bug here due to the user ignoring the suggestions coming from iFixIt, if our team had time to work on this in the future, we would work on some script that will double check to make sure that the user only accepts the selection from the suggestion box rather than able to enter whatever input the user provides.

Feature 2: Troubleshooting Guide With Gemini

Given a valid device and a problem, we aim to utilize the LLMs like Gemini to recommend to users how and why their devices are not working as expected. By understanding why their devices have a problem, it will help them understand what to do in other parts of our application: to either repair their current device or recycle and replace their device with a new one.

Feature 3: Repair Guide via iFixIt

When the user chooses to repair their device after looking through the troubleshooting guide, there will be a button that allows them to proceed to looking at potential repair guides for their issues. Upon clicking the button, the user will be directed to the iFixIt website, where a search query of their device is already performed, and all repair guides associated with their device can be found. Then, using what is recommended in the troubleshooting guide, the user can choose a specific repair guide tailored to their issues.

Feature 4: Local Recycling Centers & Repair Shops Locator

After the user has decided that their device can no longer be fixed on their own, they can choose to locate an electronic disposal/recycling service to responsibly discard their device or find a repair shop to send their device to. The map page has a search bar where the user can enter an address, place name, zip code, or city to find the nearest service locations. There are additional search features that allow the user to pick between repair and recycling centers, select the search radius, filter by rating, and hide locations that are not currently open. The map implements live search, meaning that changing the search filters will automatically update the map pins and search results without the user needing to click the search button again. Changing the search radius will also automatically adjust the map zoom accordingly.

The map integration and search feature uses Google's Maps and Places API. Our team used a free version that allows unlimited Places queries but has limited search results. For every location look-up, only the most relevant 20 results are returned. This means that even in large search radii, there will only be 20 results. The only way to mitigate this issue is to upgrade to a premium plan, which may be an option if we decide to expand our project in the future.

Requirements and Dependencies

Set up instructions:

1. Run `git clone` https://github.com/aungpaingsoe3/ewaste_identifier.git in the terminal to get the project source code.
2. Enter `cd client` to move to the `client` directory.
3. Create a `.env.local` file in the client folder and include the following:
    a. `GOOGLE_MAPS_API_KEY=<insert-valid-api-key-here>`
    b. `NEXT_PUBLIC_GOOGLE_MAPS_API_KEY=<insert-prev-api-key-here>`
    c. `NEXT_PUBLIC_GEMINI_API_KEY=<insert-valid-gemini-key-here>`
4. Run `npm install` to resolve some dependencies that are used in this project, such as:
    a. `@react-google-maps/api`
    b. `React-markdown`
    c. `@google/genai`
5. Run `npm run build` to create an optimized production build.
6. Run `npm start` to begin testing the application.