Report 5
sdmay-25-44

FixIt

Benjamin Muslic
William Griner
Jonathan Duron
Mohamed Elaagip

Prof. Berk Gulmezoglu

Weekly Summary

This week, our team focused on resolving issues with reading DTC codes. We were unable to make the Arduino Uno work for this as of now, so we switched to using the ESP32 with a WiFi-enabled ELM327, which allowed us to successfully read the codes. Additionally, we conducted more research on the UI side to improve user interaction. The objective for the week was to keep the project moving forward despite delays from midterms and jobs, and the switch to new hardware was essential to getting the system to function properly.
We also researched how the entire team can deploy the app onto each of our devices. The best way we can share our app is through apple flight test and the google play store.

Past Week Accomplishments

Benjamin Muslic

After struggling recently to try to specifically extract the DTCs from his car, he was able to figure it out. He had to make some changes in the hardware to replicate an experiment. For now he is putting the Arduino Uno aside and using an ESP32 instead which looks like the following image:

He educated us that the ESP32 is in fact a much more powerful micro controller. It has a speed of 240MHz compared to Arduino Uno 16MHz. It also has built in WiFi and Bluetooth which is perfect for our intention.

Next, he acquired a new connector. A generic Wi-Fi OBD reader. These ones are bare bones and primarily used for development situations such as the one we are experiencing now. It looks like:

So he hooked up the OBD reader to the OBD port in his car and installed respective libraries for the ESP32. With the assistance of the library, he was able to figure out how to program checking DTCs although he had to make some open-source contributions to accept Wi-Fi instead of Bluetooth.

```
    Serial.println("Connected to ELM327");

    // Demonstration of calling currentDTCCodes in blocking mode
    Serial.println("Performing DTC check in blocking mode...");

    myELM327.currentDTCCodes();
    if (myELM327.nb_rx_state == ELM_SUCCESS)
    {
        Serial.println("Current DTCs found: ");

        for (int i = 0; i < myELM327.DTC_Response.codesFound; i++)
        {
            Serial.println(myELM327.DTC_Response.codes[i]);
        }
        delay(10000); // Pause for 10 seconds after successful fetch of DTC codes.
    }
    else if (myELM327.nb_rx_state != ELM_GETTING_MSG)
    {
        myELM327.printError();
    }
}
```

On the Arduino IDE serial monitor we saw the chars coming in and the programs deciphering in action to ultimately find a code that was triggering his check engine light. It is code P0410:

```
Delimiter found.
All chars received: 4302041004104300
ELMduino: Found code: P0410
```

According to research it means a malfunction in the secondary air injection system. There are countless fixes across different car makes and models. This is the barrier we are trying to eliminate for our app. Advising the user what it is and what is the optimal

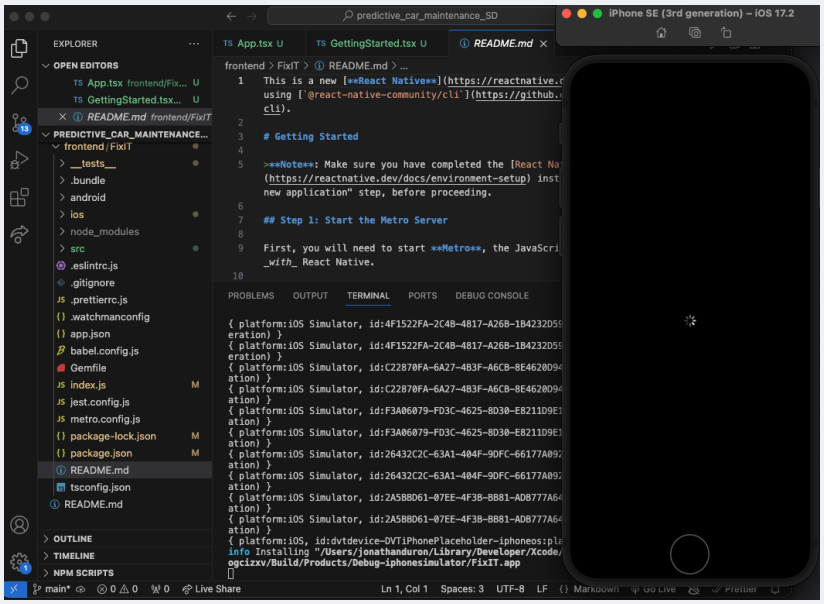
fix according to his/her vehicle. Ben was able to get this info relayed to the serial monitor on the Arduino IDE but he said the goal is to display on the phone. He will investigate using additional hardware to make this happen so we can ultimately have those codes displayed in the app for our software to analyze. But he let us know this is significant progress as we are not reliant on existing readers made by different companies.

Issues: He is still a bit confused on how to rely on that information to the phone. Also, the OBD-II and ESP32 are talking via Wi-Fi at the moment, and we hope this does not become an issue in the future. If it is an issue, we must figure out how to use Bluetooth, but Ben has had trouble getting it to connect

Jonathan Duron

As I did some research on how I can start distributing this app to my teammates there are two options. One being I install the app by plugging in their devices onto my laptop and running the app on my machine. Or I get an apple and android play store that will allow me to upload app onto their flight test store. Having the apps added to the flight test will allow me to internally send it to devices by them giving me their emails.

Issues: Emulator can take up to 5-10 minutes just to boot up. It usually stays the screen below and takes almost 10 minutes to load. I'm trying to do some research into why.



Would also tend to crash sometimes.

```
error Failed to launch the app on simulator, An error was encountered processing the command (domain=
FBSOpenApplicationServiceErrorDomain, code=5):
Simulator device failed to launch org.reactjs.native.example.FixIT.
The system shell (SpringBoard:43118) probably crashed.
Underlying error (domain=FBSOpenApplicationServiceErrorDomain, code=5):
        The request to open "org.reactjs.native.example.FixIT" failed. The system shell (SpringBoard:
43118) probably crashed.
        The system shell (SpringBoard:43118) probably crashed.
```

The Derived data from the emulator also takes up so much space on my laptop which leades me to run out of storage on my machine.

Mohamed Elaagip

This week, I focused on enhancing the user interface (UI) design for our app to improve user interaction and experience. My work involved several key activities:

- UI Design and Skeleton Development: I dedicated time to developing a comprehensive UI skeleton for the app. This involved creating wireframes and mockups that outline the basic structure and layout of the app's interface. The goal was to ensure that the design is intuitive and user-friendly, facilitating easy navigation and interaction.
- Market Research: I conducted thorough research into current market trends and competitor apps to identify best practices in UI design. This research helped me understand what users expect from similar applications and how we can differentiate our app by offering a superior user experience.
- Design Tools Utilization: I utilized design tools such as Figma to create detailed mockups. These tools allowed me to experiment with different design elements, color schemes, and typography to find the most visually appealing and functional design for our app.

William Griner

This week I assisted in developing the UI design with Mohamed. I did research on the psychology of effective UIs and discussed with Mohamed on how we could implement these principles into our design. One such principle is cognitive load reduction. This principle focuses on minimizing the mental effort required for users to understand and interact with the app. For our app, this would mean using clear, concise instructions, step-by-step guidance, and intuitive visuals (like diagrams or icons) to help users quickly grasp the complex topics regarding why their car is broken, without feeling overwhelmed. I also collaborated with Moe while creating our Figma mockup, again focusing on a user-friednly and simple experience

Plans for the upcoming week

Benjamin Muslic

Ben is going to try to relay that DTC's he was able to extract with his code to the phone. On the phone he will try to find some way utilize the Bluetooth and/or Wi-Fi to connect to the ESP32 and see what is exactly also displayed on the serial monitor in the Arduino IDE. In this case, the end user is not required to have a laptop on them.

Jonathan Duron

I plan to start communicating and implementing the calls for the backend calls to allow users to login onto the app. I will do this by talking to the backend crew and asking them how we can set up the backend to allow users login.

Mohamed Elaagip

- Refinement of UI Elements: I plan to refine the UI elements based on feedback from the team and potential users. This will involve adjusting layouts, improving color contrast, and ensuring that all interactive elements are easily accessible.
- Collaboration with Development Team: I will work closely with the development team to ensure that the UI design is effectively translated into code. This collaboration will help address any technical limitations early in the process, ensuring a smooth integration of design and functionality.

William Griner

- further investigate and implement web scraping, cloud infrastructure, fine-tuning of LLMs, and the database structure we'd need for ML purposes.
- Continue working on UI features and implementation.

Individual Contribution

Benjamin Muslic

Hours this week: 8
Cumulative: 41

Jonathan Duron

Hours this week: 6
Cumulative: 30

Mohamed Elaagip

Hours this week: 6
Cumulative: 36

William Griner

Hours this week: 5
Cumulative: 32