ECE:4890, ECE Senior Design

Smart OBD Diagnostic System (SODS)

Group 3: Cole Arduser, Luke Farmer, Sam Loecke, Samuel Nicklaus

Abstract

SODS allows users to access comprehensive, real-time vehicle diagnostics and detailed post-trip analyses. It features a physical dashboard for in-car live data display and a web application for thorough analysis and review of vehicle performance and health. The project's goal is to make informed vehicle maintenance and performance decisions accessible to all car owners.

Introduction

Rationale

Smart OBD Diagnostic System (SODS) enhances vehicle safety and maintenance by providing real-time diagnostics and comprehensive post-trip analysis in a user-friendly format, bridging the gap between advanced vehicle technology and everyday usability.

Goals and Objectives

Enhance Diagnostics: Enable real-time monitoring from the vehicle's OBD II port to encourage proactive maintenance.

User-Friendly Design: Develop intuitive in-car and web interfaces for easy access to vehicle data.

Comprehensive Analysis: Offer detailed historical analytics to identify trends and prevent issues.

Ensure Scalability and Security: Design the system to incorporate future technological advancements and data security.

Requirements, Constraints and Standards

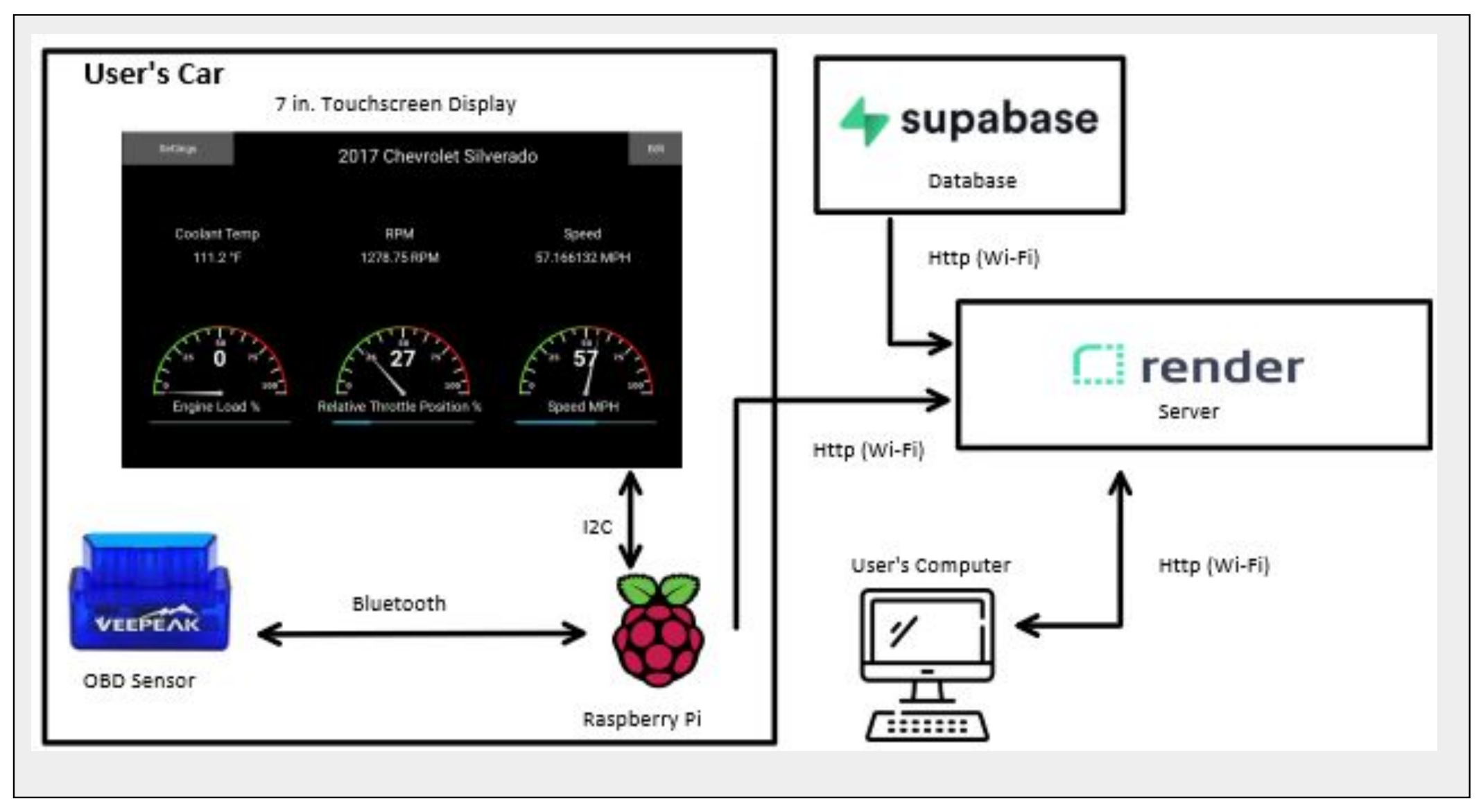
Major System Requirements:

- L1.1: System collects data from the car's OBD II port.
- L1.2: System saves data in a local database.
- L1.3: System provides an interactive touchscreen for viewing live data.
- L1.4: System displays and assesses engine trouble codes.
- L1.5: System uploads data to a web server.
- L1.6: System manages and stores data from vehicles.
- L1.7: System utilizes an efficient data storage system.
- L1.8: System offers a web application for data viewing and analysis.

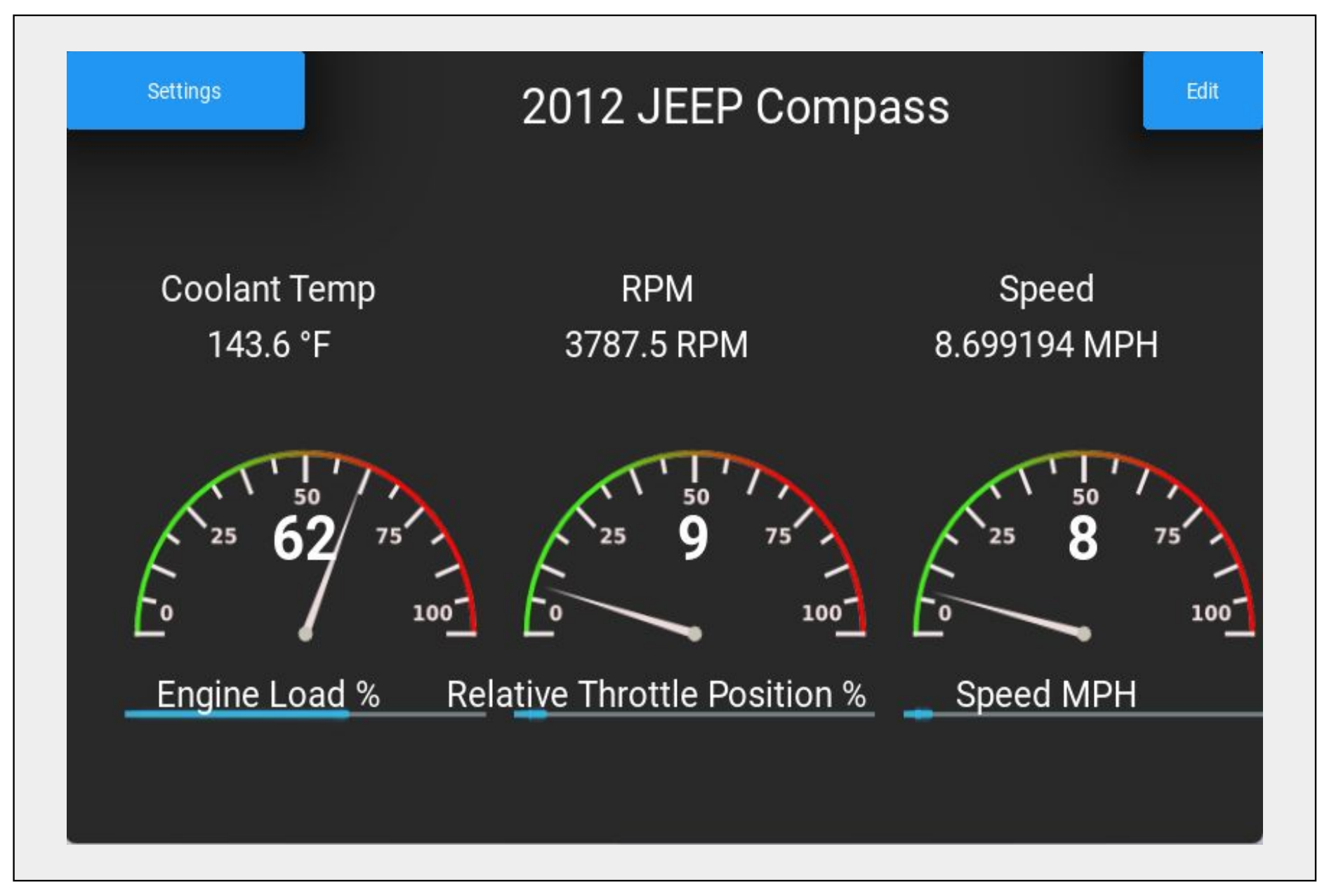
Bluetooth OBD sensor that is plugged into the car and connected to Raspberry Pi.

Design and Implementation

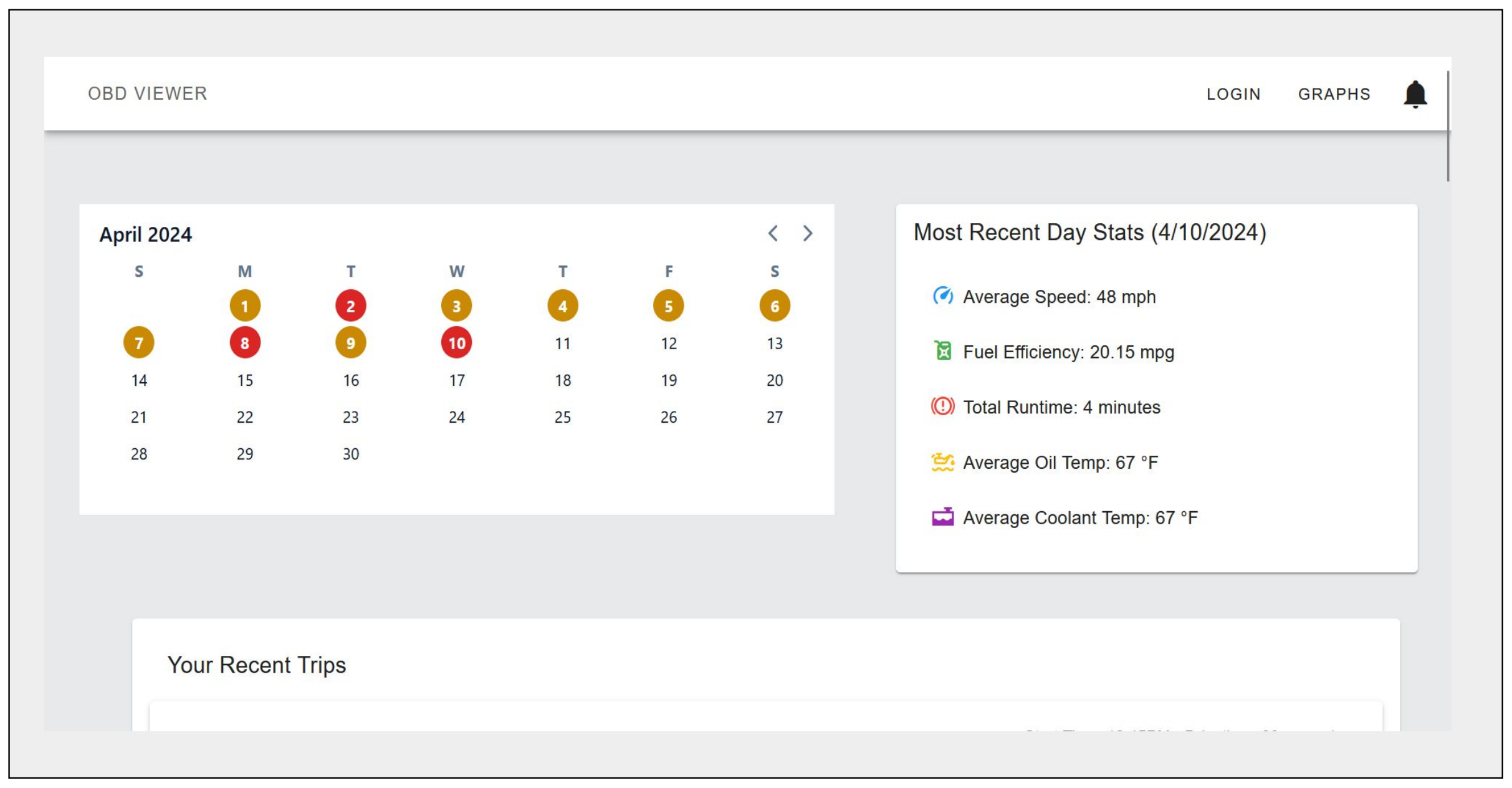
The design of our system includes in car components as well as online components. It all starts with the Bluetooth OBD sensor connected to the user's car's OBD port. This sensor communicates with the Raspberry Pi via Bluetooth. The Raspberry Pi runs the data collection software as well as the in car live data dashboard display. The Raspberry Pi then communicates with the online server via HTTP calls. The online server is hosted on Render and receives the data from the Pi. The server then communicates with both the client and the database. The database is hosted on Supabase.



System overview of our Smart OBD System.



Main Display Screen for the In-Car Heads Up Display



April 2024

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

Most Recent Day Stats (4/10/2024)

Average Speed: 48 mph

Fuel Efficiency: 20.15 mpg

Total Runtime: 4 minutes

Average Oil Temp: 67 °F

Average Coolant Temp: 67 °F

Your Recent Trips

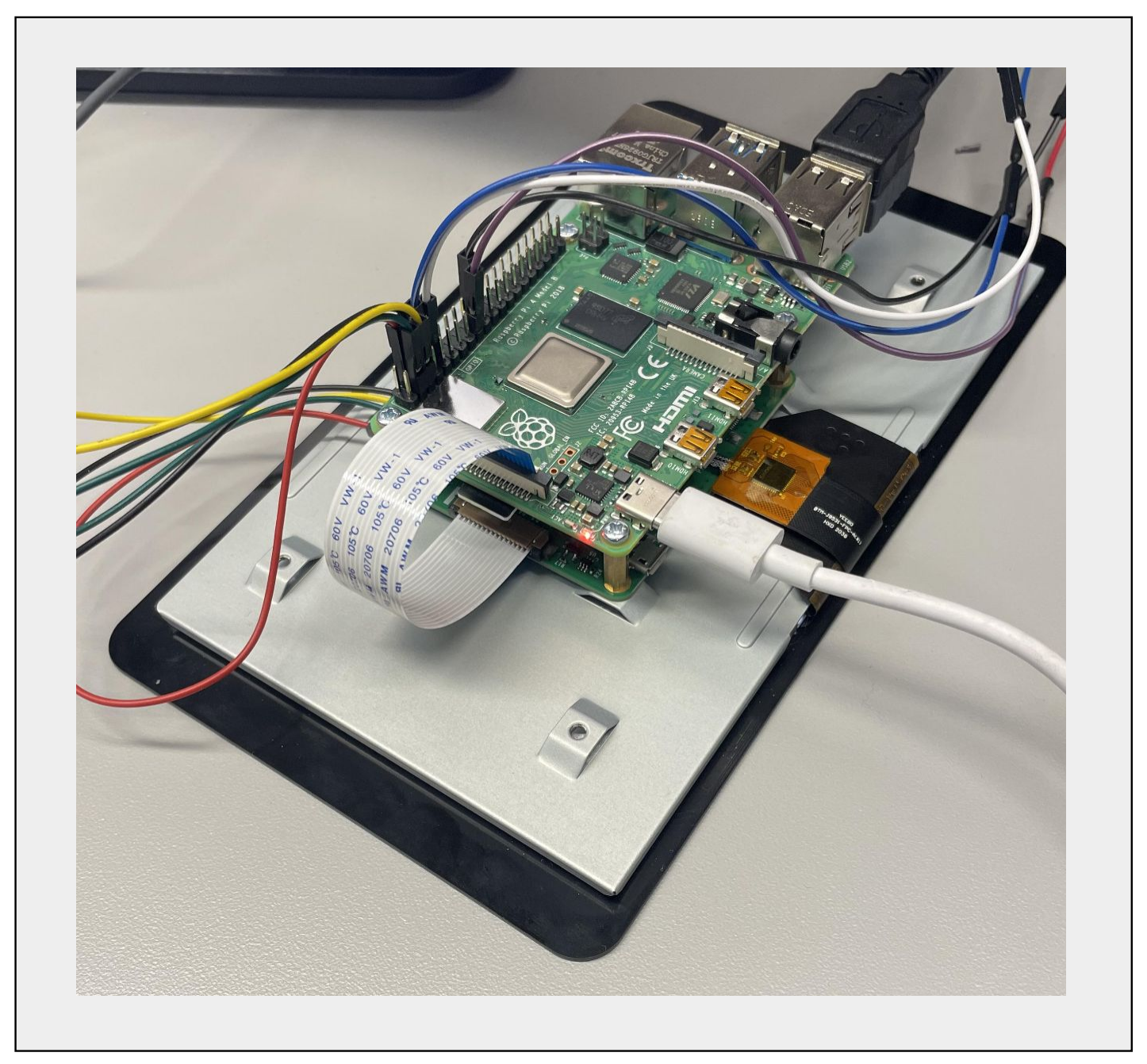
Home screen of the web client, features include calendar view for daily stats (top left), most recent drive stats (top right), and list of most recently completed trips (bottom)

Testing

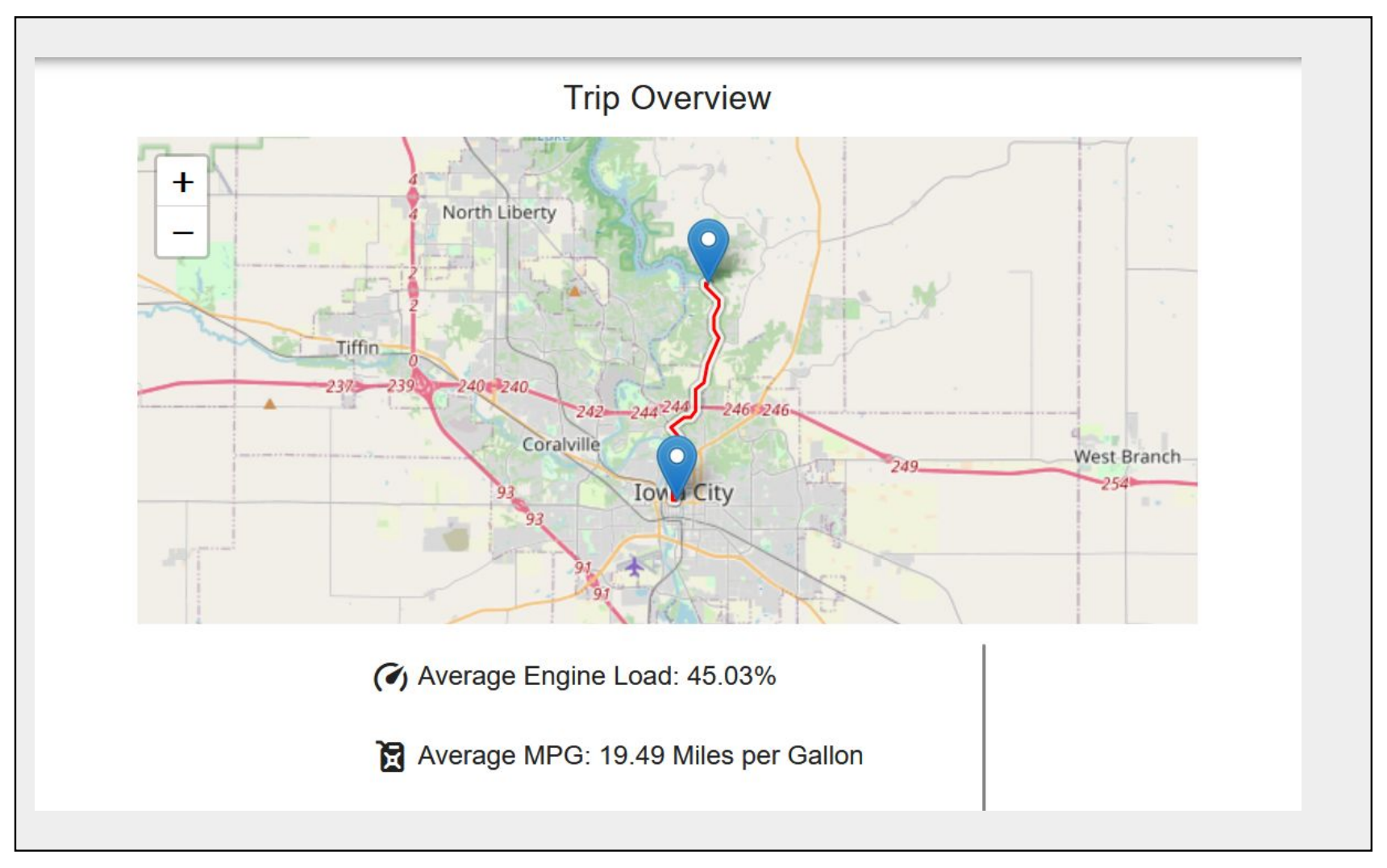
Test ID	Test Description	Expected Outcome	Actual Outcome	Pass/Fail
TC001	Verify system's capability to collect data from the car's OBD II port continuously.	Data from all available OBD II parameters is collected without interruption.	Data successfully collected without interruption.	Pass
TC002	Confirm that the system saves all collected data to the local database accurately.	All transmitted data is accurately saved in the database.	All data accurately saved in the local database.	Pass
TC003	Test the responsiveness and accuracy of the live data display on the touchscreen interface.	Interface responds within 250 ms and displays accurate data.	Response time within 200 ms; data displayed accurately.	Pass
TC004	Evaluate the system's ability to display engine trouble codes and assess their severity.	System accurately displays trouble codes and provides a correct severity assessment for each.	Trouble codes and severity accurately displayed.	Pass
TC005	Test the system's ability to upload collected data to the specified web server.	Data is successfully uploaded and matches the local database entries.	Data uploaded successfully and verified on the server.	Pass
TC006	Ensure the web server correctly collects and stores received data.	Web server receives all data packets and correctly stores them in the database.	Data correctly received and stored by the web server.	Pass
TC007	Verify the database's efficiency in data storage and retrieval processes.	Operations complete within reasonable time frames and without excessive resource consumption.	Database operations efficient and within expected resource usage.	Pass
TC008	Assess the web application for viewing, comparing, and analyzing collected data.	All features function as intended, with data accurately represented.	Web application functions correctly; data representation accurate.	Pass

	Test Case ID	TC001	TC002	TC003	TC004	TC005	TC006	TC007	TC008	Total Test Cases
Requirement ID										
L1.1		X	X							2
L1.2		X	X							2
L1.3			X	X	X					3
L1.4				X	X					2
L1.5						X	X			2
L1.6						X	X	X		3
L1.7					X	X		X		3
L.1.8								X	X	2

Testing Traceability Matrix



Raspberry Pi attached to the display screen



Average Engine Load: 45.03%

Average MPG: 19.49 Miles per Gallon

Trip overview page, showing the user trip data such as average engine load, MPG, and runtime. Also maps out the user's route, they took

Project Outcome

Critical Assessment

Successfully integrated real-time and post-trip vehicle diagnostics in a user-friendly system, meeting key project objectives and requirements.

Deviations

Adjusted the power source specifications to enhance compatibility across various car models

Team Contributions

- Samuel Nicklaus: Server and front-end development
- Cole Arduser: Emulation environment, server hosting.
- Sam Loecke: OBD-Raspberry Pi integration, backend GUI.
- Luke Farmer: Hardware procurement, GUI development.

Challenges

- Database Access
- GUI Development due to Kivy

Skills Acquired

Developed key skills in documentation, requirements engineering, systematic testing, and project management.

Project Management

Throughout our project, we conducted bi-weekly check-ins with the professor and teaching assistant and used a Gantt chart to track tasks and timelines. We utilized GitHub for both bug tracking and pull request management, ensuring that any changes pushed to the main branch were reviewed and approved by at least two team members. Regular code reviews also helped maintain group understanding of the full stack. This experience showed us how important careful planning is when managing big projects. It also made clear that having organized systems in place helps a lot with keeping track of progress and getting things done smoothly.

Conclusion

Our senior design project successfully developed a smart OBD sensor for vehicles, featuring an integrated head unit and web interface. This enhancement significantly improves upon traditional vehicle diagnostics and user interaction. With an estimated commercial cost of $100.

Acknowledgements

Thanks to Dr. Tyler Bell for lending us a Raspberry Pi