DRIVAID

A smart driving monitor

Group E1: Samraj Kalkat, Ryan Vimba, Reid Yesson

Application Area

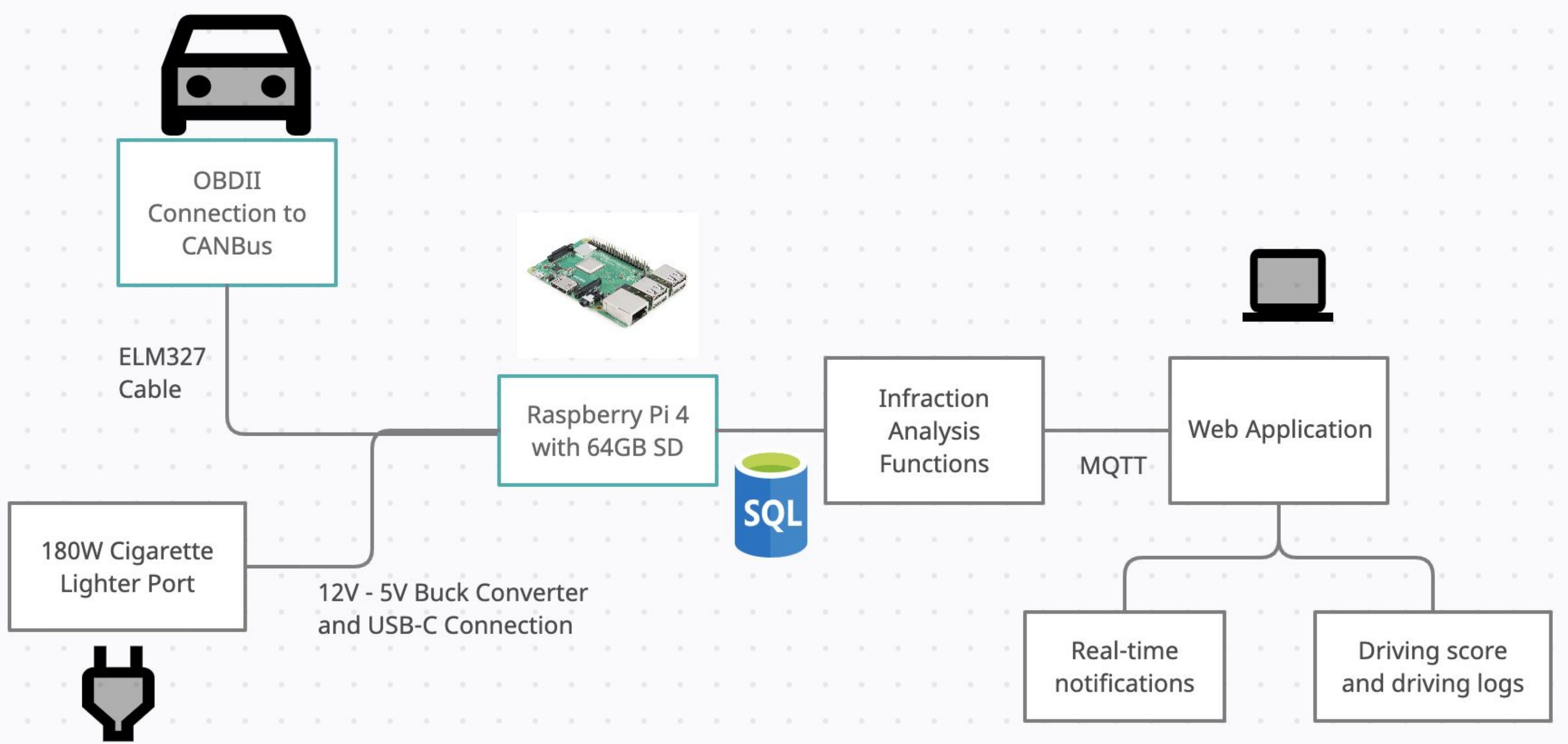
Why?

- Drivers need feedback to improve their driving
- Lack of feedback and constructive criticism available

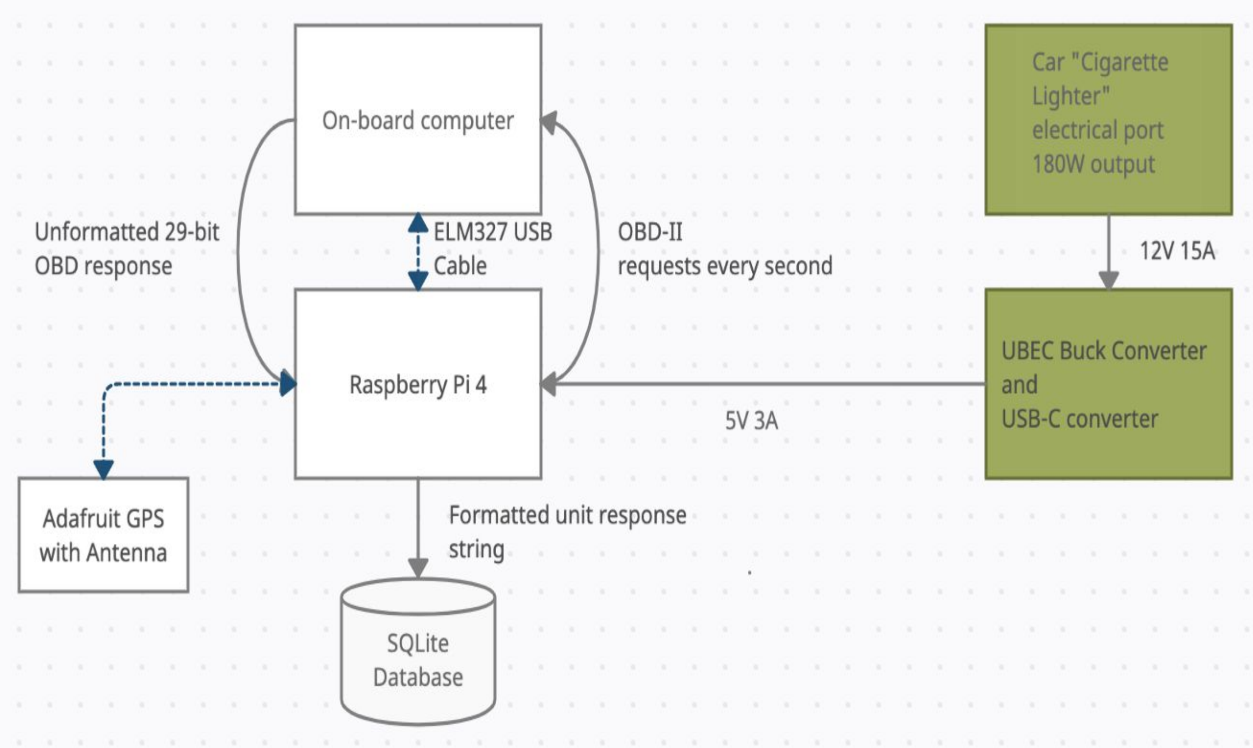
Solution

- Detect and identify poor driving habits
- Notify driver in real time of infractions
- Log and share driving records / data

Solution Approach


OBDII Connection to CANBus
ELM327 Cable
180W Cigarette Lighter Port
12V - 5V Buck Converter and USB-C Connection
Raspberry Pi 4 with 64GB SD
SQL
Infraction Analysis Functions
MQTT
Web Application
Real-time notifications
Driving score and driving logs

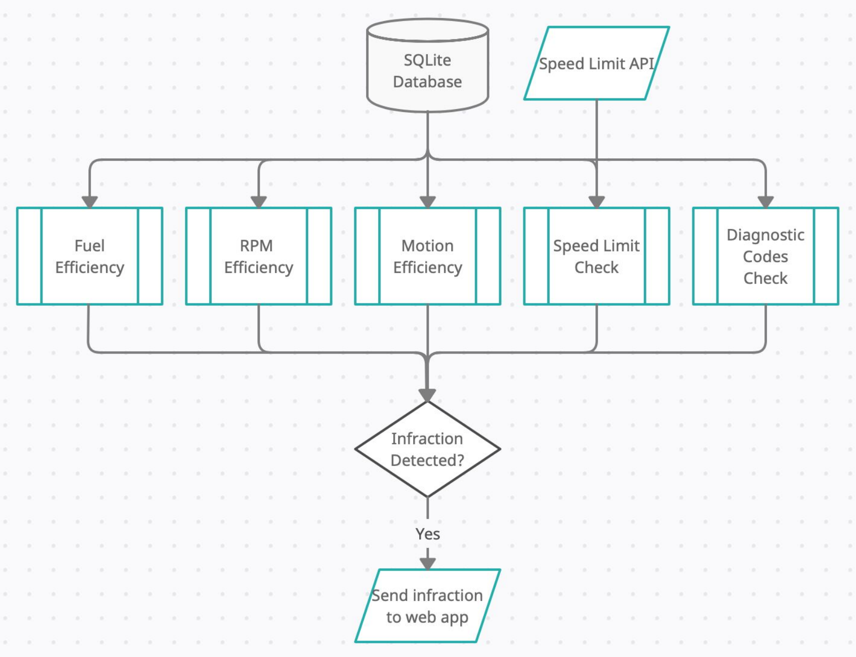
Complete Solution - Signals and OBD-II

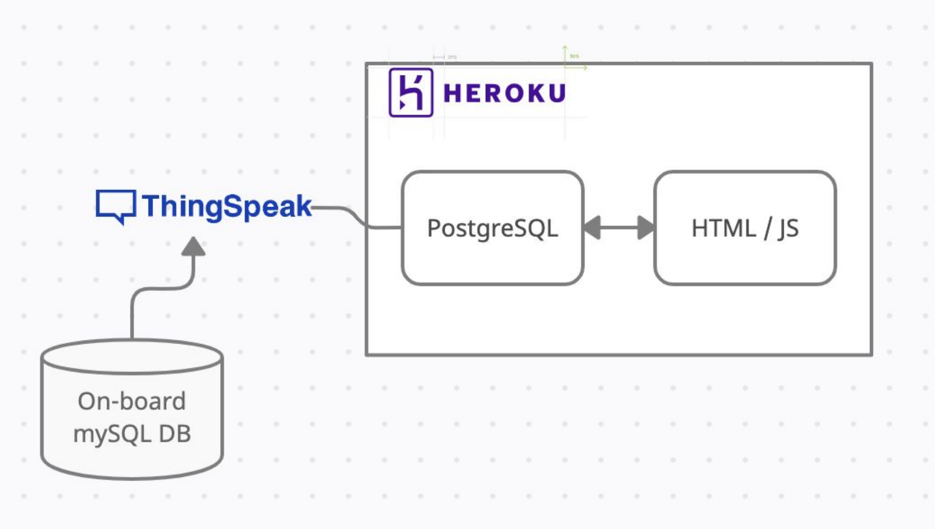

On-board computer
Car "Cigarette Lighter" electrical port 180W output
Unformatted 29-bit OBD response
ELM327 USB Cable
OBD-II requests every second
12V 15A
UBEC Buck Converter and USB-C converter
Raspberry Pi 4
5V 3A
Adafruit GPS with Antenna
Formatted unit response string
SQLite Database

CHANGES:

- We swapped the PiCan2 after not receiving data despite communication with the founder and creator among many repeated attempts
- We also added an antenna to our GPS hardware

Complete Solution - Infraction Detection and Website



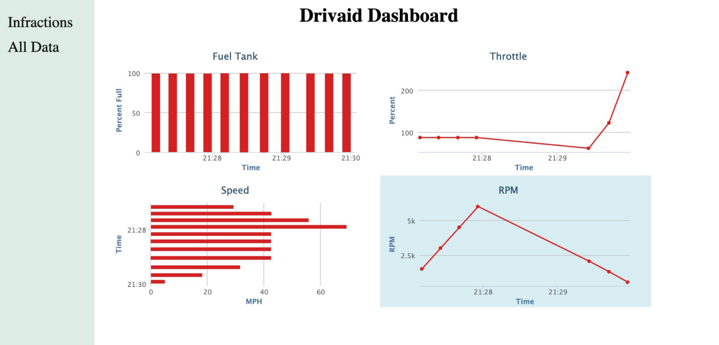


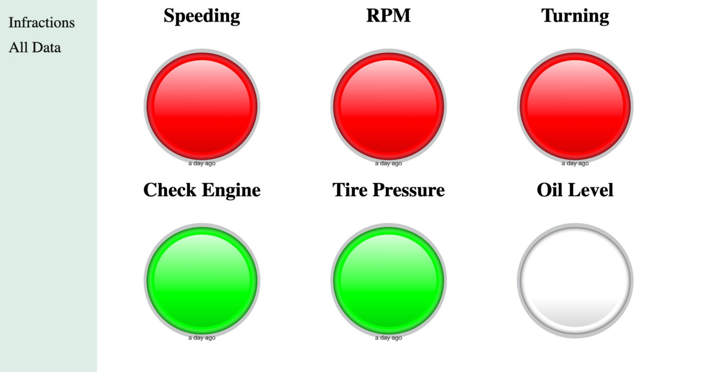
CHANGES:

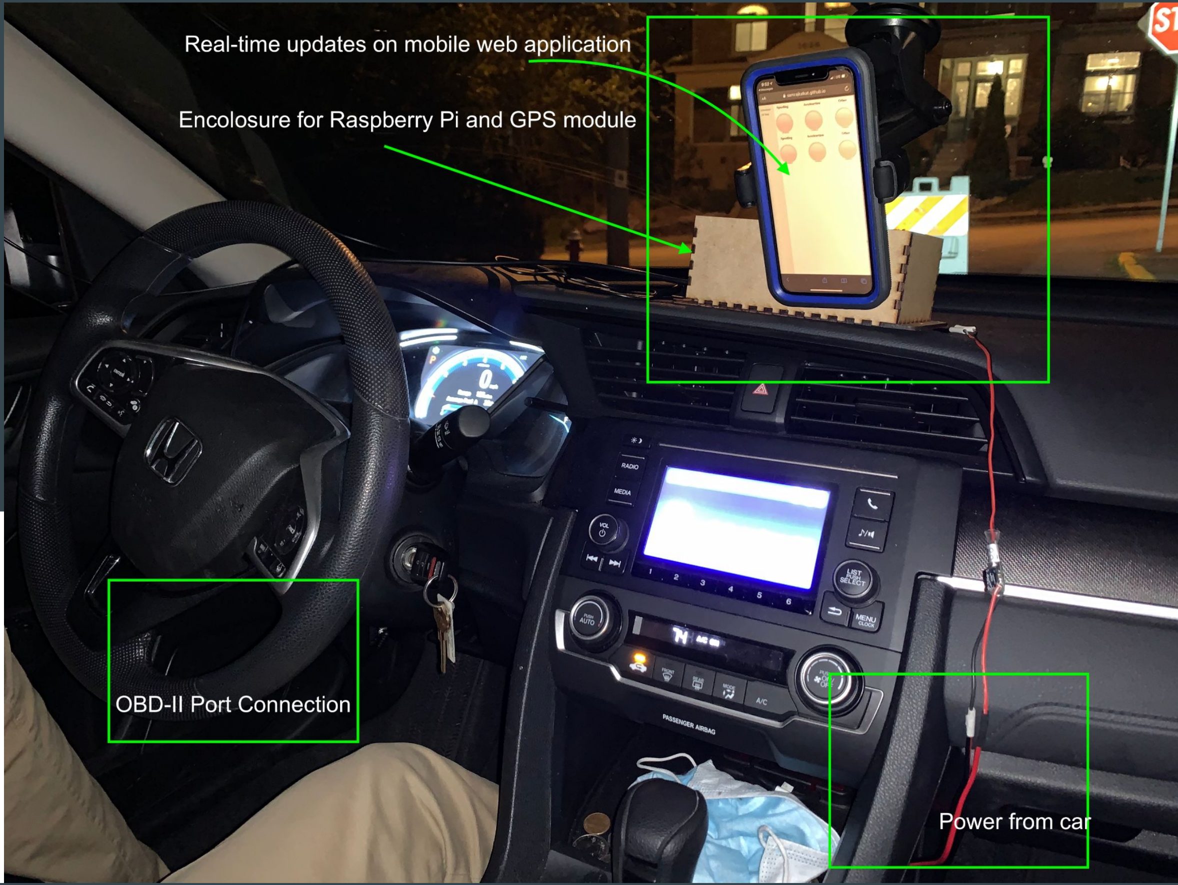
Different measurements on Infraction detection due to unreadable data.

Different data storage approach for website.

Complete Solution






Real-time updates on mobile web application

Encolosure for Raspberry Pi and GPS module

OBD-II Port Connection

Power from car

Testing, Metrics, and Validation (OBD-II Logger)

Requirement	Testing / Metrics/ Verification	Results
Request and receive data from OBD-II across timestamps	Comparing vehicle dashboard to database file and web app	ECU returned data in controlled driving test in parking lot, alongside non-driving tests
Handling zeroed data returned by the onboard computer/ECU	Viewing database file after zero detection in moving and non-moving positions	No seen spikes in web application after timestamp control during driving test
Data formatting correlates with SQLite	Error checking in ELM Logging file and creating dummy improperly formatted data to be checked before sending to DB	Python types match up and improper formatting is caught before DB and replaced with previous timestamp data point
Arduino receives 5V 3A from UBEC Converter	Multimeter recording verification	No current recordings above 3A, voltage steady at 5.2V, power steady at 15W regardless

Testing, Metrics, and Validation (Algorithms and DB)

Requirement	Testing / Metrics/ Verification	Results
Speed Limit Check Accuracy	Find longitude and latitude points with speed limits of 25, 35, and 55 mph and verify they return correct limit	Success, locations on Forbes, Fifth, and highway all returned correct speed limits
Fuel Efficiency accuracy	Go for a 10 minute drive and compare fuel efficiency calculated to the car’s fuel efficiency	Success, result calculated was within 5 miles of fuel efficiency displayed from the car
RPM Efficiency accuracy	Run RPM check on data set with 100 points that bring the RPM above 6500	Success, check found exactly 100 infractions
Motion Efficiency accuracy	Run motion efficiency check on data set with 50 points that bring acceleration above 1.5 m/s^2 and 50 that bring it below $-1.5 m/s^2$	Success, check found exactly 100 infractions

Testing, Metrics, and Validation (MQTT and Website)

Requirement	**Testing / Metrics/ Verification**	**Results**
Data is sent and updates within 5 seconds on the website	Sending test data and observing the response time	Success, the website updates consistently with an average of ~5 seconds
Data is properly stored on the site without data corruption or faults	Data stream test script to the hosting service and observing that data is correctly places	Success. The data sent from the script consistently matches the data being stored

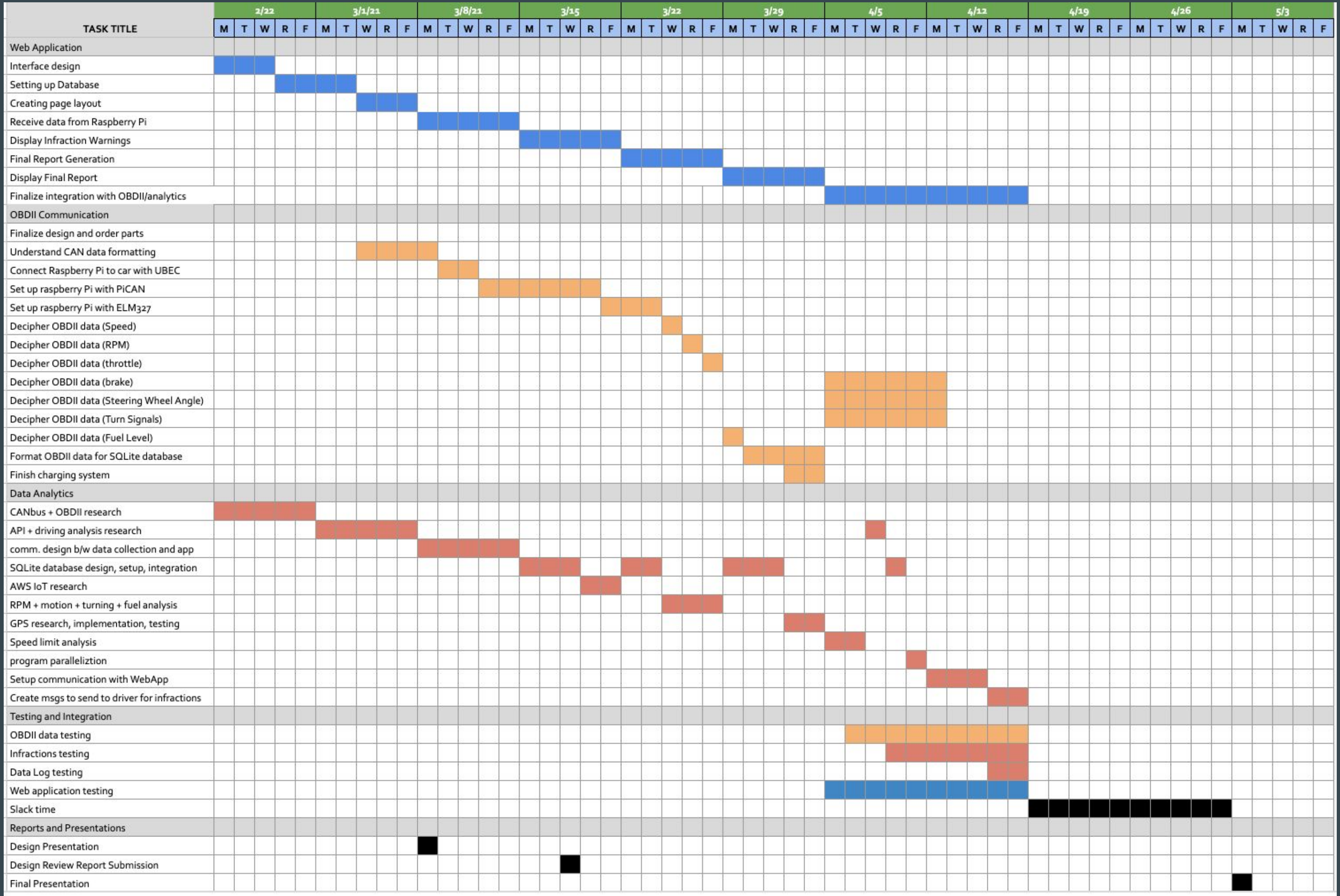
Testing Results (Integrated System)

- Driving and stationary tests showed correct data transfer from the OBD-II port to the web application
- Ran a driving test in an empty parking lot and recreated inefficient driving events (accelerate then brake constantly over short distance) to see if fuel efficiency light shows up on Web App
- In park, we revved engine to see if RPM light turned red
- Passenger-side database monitoring allowed us to tweak some of the algorithms after we saw that they were too sensitive to small changes in acceleration/throttle

Trade-offs

Tradeoff	Issue	Solution
PiCAN	Not able to view data using the PiCAN board. Tested the board and contacted support	Switched to ELM327 usb interface
Steering wheel angle & turn signal	Could not read the steering wheel angle or turn signals. These values cannot be accessed using standard OBD-II codes	Removed these metrics and replaced with readable data
Odometer	Similarly, odometer is restricted due to security and cannot be accessed via OBD-II contrary to original belief	Instead, we calculate accurate mileage by using speed and timestamps to compute a distance
Google Places API	Could not get speed limits without special license	Pivoted to OpenStreetMaps API to access speed limits (more limited dataset)

Project Management



	2/22					3/1/21					3/8/21					3/15					3/22					3/29					4/5					4/12					4/19					4/26					5/3				
TASK TITLE	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
Web Application																																																							
Interface design	🟦	🟦	🟦																																																				
Setting up Database				🟦	🟦	🟦	🟦																																																
Creating page layout								🟦	🟦	🟦																																													
Receive data from Raspberry Pi											🟦	🟦	🟦	🟦	🟦																																								
Display Infraction Warnings																🟦	🟦	🟦	🟦	🟦																																			
Final Report Generation																					🟦	🟦	🟦	🟦	🟦																														
Display Final Report																										🟦	🟦	🟦	🟦	🟦																									
Finalize integration with OBDII/analytics																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦															
OBDII Communication																																																							
Finalize design and order parts																																																							
Understand CAN data formatting								🟧	🟧	🟧	🟧																																												
Connect Raspberry Pi to car with UBEC												🟧	🟧																																										
Set up raspberry Pi with PiCAN														🟧	🟧	🟧	🟧	🟧	🟧																																				
Set up raspberry Pi with ELM327																				🟧	🟧	🟧																																	
Decipher OBDII data (Speed)																							🟧																																
Decipher OBDII data (RPM)																								🟧																															
Decipher OBDII data (throttle)																									🟧																														
Decipher OBDII data (brake)																															🟧	🟧	🟧	🟧	🟧	🟧																			
Decipher OBDII data (Steering Wheel Angle)																															🟧	🟧	🟧	🟧	🟧	🟧																			
Decipher OBDII data (Turn Signals)																															🟧	🟧	🟧	🟧	🟧	🟧																			
Decipher OBDII data (Fuel Level)																										🟧																													
Format OBDII data for SQLite database																											🟧	🟧	🟧	🟧																									
Finish charging system																													🟧	🟧																									
Data Analytics																																																							
CANbus + OBDII research	🟥	🟥	🟥	🟥	🟥																																																		
API + driving analysis research						🟥	🟥	🟥	🟥	🟥																							🟥																						
comm. design b/w data collection and app											🟥	🟥	🟥	🟥	🟥																																								
SQLite database design, setup, integration																🟥	🟥	🟥			🟥	🟥				🟥	🟥	🟥						🟥																					
AWS IoT research																			🟥	🟥																																			
RPM + motion + turning + fuel analysis																							🟥	🟥	🟥																														
GPS research, implementation, testing																													🟥	🟥																									
Speed limit analysis																															🟥	🟥																							
program paralleliztion																																			🟥																				
Setup communication with WebApp																																				🟥	🟥	🟥																	
Create msgs to send to driver for infractions																																							🟥	🟥															
Testing and Integration																																																							
OBDII data testing																																🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧															
Infractions testing																																		🟥	🟥	🟥	🟥	🟥	🟥	🟥															
Data Log testing																																							🟥	🟥															
Web application testing																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦															
Slack time																																									⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛					
Reports and Presentations																																																							
Design Presentation											⬛																																												
Design Review Report Submission																		⬛																																					
Final Presentation																																																			⬛				