Cruise Control
Software Development Document
V.0.11

Team SCJB:

Sunmin Lee

Cassie Ball

Julia Nelson

Seung (Ben) Eom

"I pledge my honor that I have abided by the Stevens Honor System."

Table of Contents

1. Executive Summary
2. Introduction
    a. Customer need
    b. Major features
    c. Reliability
    d. Performance
3. Requirements
    a. Functional Requirements
        i. User Inputs
        ii. Sensors
        iii. EMS
        iv. Feedback from Outputs
    b. Systems Requirements
    c. Non-Functional Requirements
        i. Performance
        ii. Reliability
        iii. Availability
        iv. Maintainability
        v. Security
4. Requirements Analysis Model
    a. UML Use Cases & Diagrams
        i. Use Case 1
        ii. Use Case 2
        iii. Use Case 3
        iv. Use Case 4
        v. Use Case 5
        vi. Use Case 6
        vii. Use Case 7
    b. UML Class-Based Modeling
    c. UML CRC Model Index Card
    d. UML Activity Diagram
    e. UML Sequence Diagram
    f. UML State Diagram
5. Software Architecture
    a. Architecture Type
    b. Components, Connectors, & Constraints
    c. Control Management

    d. Data Architecture
    e. Diagrams
    f. Issues
6. Code
7. Tests
8. Issues

1. Executive Summary

This project is to develop a cruise control that is used to automatically maintain a car's speed without the user having their foot on the gas pedal. Cruise control is a feature offered in the majority of modern cars to provide user convenience specifically for long drives on highways. Some major features that are needed to make the cruise control work include turning the program on and off, a dashboard display of the speed of the car, buttons to increase and decrease the speed as necessary, and so on. Our goal is to successfully implement these major features with no chance of failure in order to increase the reliability of our cruise control software and enhance the overall performance of the cruise control system.

2. Introduction

High-Level Customer Need

Cruise control is a highly sought after feature in vehicles that electronically interacts with the engine's management system to maintain its speed. Originally invented to save gas by reducing gas-wasting surges on the accelerator, cruise control provides an opportunity for the driver to rest their foot and safely adjust their positioning, while also helping to prevent speeding tickets on long roads and highways.

Major Features

Our implementation of cruise control needs a number of standard features in order to work properly. The first major feature we need is to take input that will turn cruise control on and off. In addition, cruise control needs to be able to shut off when the user presses the brake and, when the car turns off, the entire program should stop running. We need another input that sets the speed that the cruise control will maintain. To see the speed that we are maintaining we will need a dashboard display of a speedometer that shows the current speed. Input to increase and decrease the set speed of the cruise control will also be implemented so the driver does not have to turn off cruise control and try to get the desired speed by pressing the gas. There will be a logging system that writes data to an excel spreadsheet as soon as cruise control is turned on. It will track any changes in the program such as when it turns on, being set, the current speed, and when it is turned off or deactivates. This will provide a better understanding of how the cruise control program is running and if there is a problem, it will be easier to understand where and why it occurred.

High-Level Reliability

As the reliability of our implementation of cruise control is directly related to the safety of the passengers, it is crucial to develop highly reliable software to give the passengers a sense of security. Ultimately, our goal will be for our implementation to have a success rate of 99.99% for hardware and 99.999% for software of the cruise control. To achieve this goal, we have to make sure that all of the major features mentioned above, such as turning the cruise control on and off, and setting the speed the passenger wants to maintain the vehicle at, should work 99.99% of the time. With our

agile development process, we will successfully implement all the features with a low probability of failure and test the reliability of our implementations via series of tests and experiments, to make sure that we have reached our goals.

High-Level Performance

The cruise control system will have to have optimal performance while the car is in motion. Optimal performance means that the vehicle will reach the cruise control speed that is set by the driver as soon as possible. The cruise control will also have to maintain the speed that the driver decided to set by the cruise control button until the driver turns off the engine or turns off the cruise control option. It is important that once the cruise control is turned on, and the speed has been set, that the vehicle does not dip above nor below the speed. A fluctuation of the speed of the car is an indicator of a low-level performance, which should be avoided and be accounted for when implementing the cruise control system in the vehicle.

3. Requirements

Functional Requirements

User-Inputs

FR-I-1. System shall accept the driver's activation of the system from a turn on input. The cruise control shall then turn on and display to the driver that cruise control has been turned on.

FR-I-2. System shall accept the driver's deactivation of the system from a turn off input. The cruise control shall then turn off completely and the cruise control display icon will turn off.

FR-I-3. System shall accept every setting of speed that is set by the driver. On a set speed input, cruise control shall set and maintain the speed of the car at the current speed shown on the speedometer.

FR-I-4. System shall accept the driver's increase of the set speed. On an increased speed input, cruise control shall increase the set speed by 1 mph.

FR-I-5. System shall accept the driver's decrease of the set speed. On a decreased speed input, cruise control shall decrease the set speed by 1 mph.

FR-I-6. On an acceleration pedal input, the cruise control shall accept the user's increase of set speed and the car shall speed up. If the driver wants to set the speed of the car to the speed in which he accelerated, he can hit the set input again and the car will maintain this new speed. If the driver does not hit set and he releases the accelerator, the speed of the car will decelerate back to its original set speed.

FR-I-7. On a brake input, cruise control shall turn off the set speed but the cruise control system shall still be turned on, and shall wait for a new set speed.

FR-I-8. When the car turns off, cruise control shall turn off completely.

Sensors

SN-1. The cruise control system shall have a sensor to detect the speed of the car.

SN-2. The cruise control system shall have a sensor to detect the time and use the sensor that detects the speed of the car to help log data to the log.

SN-3. The system shall receive information from sensors about the brake application continuously every 0.5 seconds

SN-4. The system shall provide an approval request to the sensors before activation

EMS

EMS-1. The system shall receive the current speed from the EMS continuously every 1 second

EMS-2. The system shall provide an activation request to the throttle (EMS)

EMS-3. The system shall provide a deactivation request to the throttle (EMS)

Feedback from Output

FR-O-1. Logging of car data shall begin within 1 second of the car turning on. Cruise control hardware shall provide visual feedback to the user on activation.

FR-O-2. When the vehicle shuts down, cruise control hardware shall provide visual feedback to the user.

FR-O-3. Once the speed has been set by the user, the cruise control hardware shall provide visual feedback on it.

FR-O-4. The cruise control shall perform an automatic shutdown after one minute if cruise control has been turned on, but a speed has not been set.

System Requirements

SR-1. System shall accept all inputs within 2 seconds after the engine starts.

SR-2. System shall accept electric power from the alternator

SR-3. System shall accept direct current from the car battery to support logging after engine is shut down

SR-4. The system shall receive the time and date from the car's clock every second

SR-5. The system shall provide an adjustment (increase or decrease) request to the throttle up to the allowable positions of speed

SR-6. The system shall provide a physical interface for technicians to access in the case where maintenance is needed.

SR-7. The system software shall be configurable by the technician

SR-8. The system hardware shall have a 99.99% availability

SR-9. The system software shall have a 99.99% availability

SR-10. The system shall have an accuracy range of plus-minus 1 of the set speed

SR-11. The system shall adhere to the safe and secure communication protocol specified by the manufacturer, industry, and government

SR-12. The system shall only be accessible only to authorized dealers via hard-wired interface and be password protected.

Non-Functional Requirements

Performance

NF-P-1. The cruise control system shall display that it is turned on within 100ms of a turn on input from the driver.

NF-P-2. The cruise control system shall turn off within 100ms of a turn off input.

NF-P-3. The cruise control system shall deactivate the set speed within 100ms of a brake input.

Reliability

NF-R-1. The cruise control system shall be available to use 99.99% of the time.

Availability

NF-A-1. The cruise control system shall be ready to use without delay 99.99% of the time.

NF-A-2. The cruise control system shall be ready to turn on within 2 seconds after the car is turned on and the engine starts running.

NF-A-3. The cruise control system shall only be able to maintain a speed once the car is driving at a minimum speed of 25 mph.

Maintainability

NF-M-1. The cruise control shall be able to maintain itself until the car becomes unable to move. If maintenance is needed, the cruise

control unit shall be able to signal to the driver whenever the system is unavailable and cruise control shall not be activated.

NF-M-2. The technician shall be able to view all of the logged data from the cruise control system.

Security

NF-S-1. The cruise control unit shall not be connected to the internet, so it will not be vulnerable to cyber attacks. The only vulnerability it has is if there is a physical harm against the unit. If the cruise control system were to fall to a physical attack that caused any controls or sensors to malfunction outside optimal operation, the cruise control system would alert the driver and deny usage.

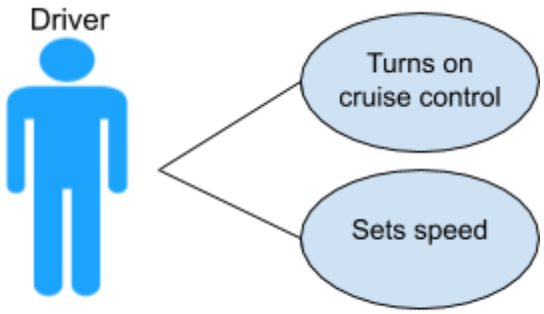
4. Requirements Analysis Modeling

UML Use Cases

Use case 1: Usage of turn on and set speed
Primary actor: The driver
Goal in context: The cruise control system turns on and the user sets speed
Precondition: The car is in good condition, and the driver is trying to use cruise control in an appropriate condition (driving at a speed of over 25 mph)
Trigger: Driver decides to use cruise control on a highway

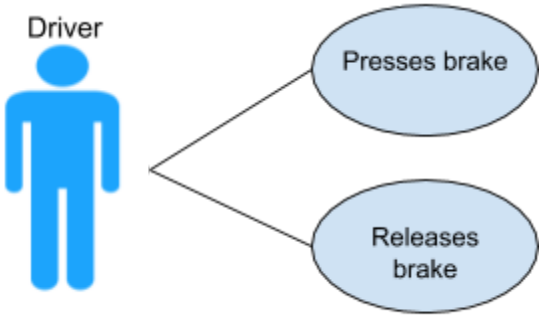
1. The driver powers on cruise control through the user interface.
2. Cruise control starts writing to a log.
3. Cruise control gives visual feedback to show that it has been activated.
4. The driver sets speed to the current speed of the car.
5. Cruise control requests the speed from sensors.
6. Sensors provide approval to set cruise control at current speed.
7. Cruise control requests the Engine Management System to set the speed at current position for desired speed.
8. The Engine Management System is set to the indicated speed, and is visually shown to the driver.
9. Cruise control detects inputs from the driver (including increase speed, decrease speed, accelerator, brake, and turn off) and adjusts speed, deactivates, or turns off cruise control completely depending on which input is given.
10. The set speed is maintained regardless of whether the driver is going uphill or downhill.
11. Speed is continuously reported to the cruise control system.

Exception: If there is something wrong with the cruise control system, it will not turn on and the user will have to get it fixed before he can use it again.



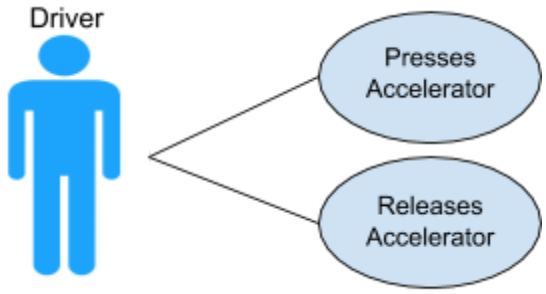
Use case 2: Usage of brake with cruise control engaged.
Primary actor: The driver.
Goal in context: Cruise control shall disengage when brake is applied.
Precondition: The car is in good condition, and the driver has activated cruise control and set the speed.
Trigger: Driver must brake to avoid potential collision.

1. While using cruise control, the driver sets foot on the brake pedal.
2. Cruise control set speed is deactivated and the car decelerates.
3. Cruise control logs the brake pedal input and the changes in the speed of the car.
4. Cruise control gives visual feedback to show that the car is decelerating.
5. Cruise control remains turned on until the user requests a speed to be set or the user turns off the cruise control system completely by using a turn off input.
6. Speed is continuously reported to the cruise control system.



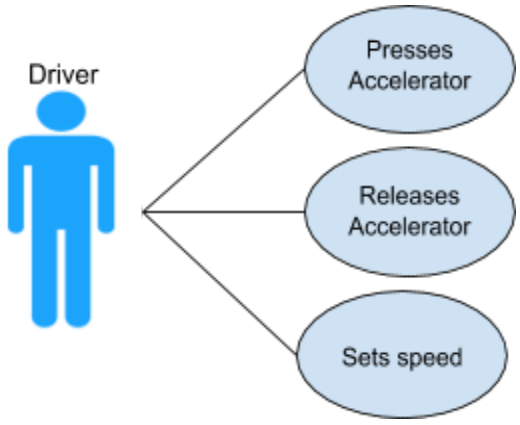
Use case 3: Usage of accelerator with cruise control engaged.
Primary actor: The driver.
Goal in context: Cruise control shall allow acceleration and then re-engage at the originally set speed when the accelerator is released.
Precondition: The car is in good condition, and the driver has activated cruise control and set the speed.
Trigger: Driver wants to use the accelerator to pass another car on the road.

1. While using cruise control, the driver sets foot on the accelerator pedal.
2. Cruise control shall allow the acceleration of the vehicle.
3. Cruise control logs the acceleration pedal input and the changes in the speed of the car.
4. Cruise control gives visual feedback to show that the car is accelerating.
5. The driver releases the accelerator and the vehicle begins to decelerate.
6. Cruise control logs the acceleration pedal input and the changes in the speed of the car.
7. Cruise control requests the speed from sensors.
8. Cruise control shall read the speed log and decelerate until the previous set speed is reached.
9. Cruise control shall request the Engine Management System to maintain the speed at the current position.
10. Cruise control shall display visual feedback of the current speed to the driver.
11. Speed is continuously reported to the cruise control system.



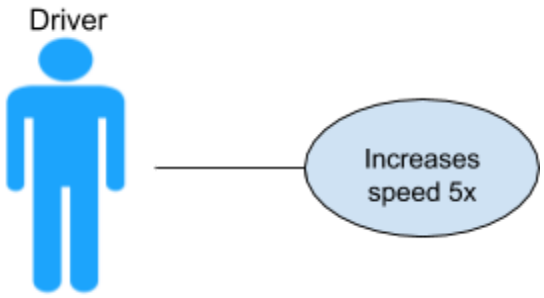
Use case 4: Usage of accelerator when driver sets speed
Primary actor: The driver.
Goal in context: Cruise control shall allow acceleration and on set speed input, sets the speed of the car to the new speed the driver accelerated to.
Precondition: The car is in good condition, and the driver has activated cruise control and set the speed.
Trigger: Driver wants to use the accelerator to pass another car on the road.

1. While using cruise control, the driver sets foot on the accelerator pedal.
2. Cruise control shall allow the acceleration of the vehicle.
3. Cruise control logs the acceleration pedal input and the changes in the speed of the car.
4. Cruise control gives visual feedback to show that the car is accelerating.
5. The driver releases the accelerator and sets the speed at the current speed.
6. Cruise control logs the acceleration pedal input and the changes in the speed of the car.
7. Cruise control requests the speed from sensors.
8. Cruise control shall read the speed log and decelerate until the previous set speed is reached or the user requests to set the speed.
9. The driver requests to set the speed at the current speed.
10. Sensors provide approval to set cruise control at current speed.
11. Cruise control requests the Engine Management System to set the speed at current position for desired speed.
12. The Engine Management System is set to the indicated speed, and is visually shown to the driver.
13. Cruise control shall display visual feedback of the current speed to the driver.
14. Speed is continuously reported to the cruise control system.



Use Case 5: Usage of the increase speed button while cruise control is on and speed is set.
Primary actor: The driver.
Goal in context: Cruise control shall increment the set speed by 1 mph with each input.
Precondition: The car is in good condition, and the driver has activated cruise control and set the speed.
Trigger: Driver wants to increase the currently set speed 5 mph, without using the accelerator pedal.

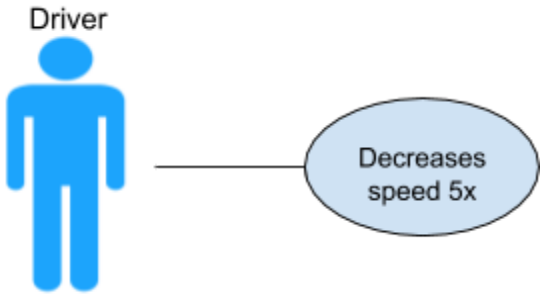
1. While cruise control is activated and currently set at a speed, the driver triggers the increase speed input on the cruise control interface.
2. Cruise control requests speed values from sensors.
3. Sensors provide approval to set cruise control at the incremented speed.
4. Cruise control logs the increase speed input and the changes in the speed of the car.
5. Cruise control requests the Engine Management System to increase the speed by 1 mph.
6. The Engine Management System is set to 1 mph over the original set speed.
7. Cruise control shall display visual feedback of the current set speed to the driver.
8. Speed is continuously reported to the cruise control system.
9. The user repeats step 1 and the cruise control system repeats steps 2-7.



Use Case 6: Usage of the decrease speed button while cruise control is on and speed is set.
Primary actor: The driver.
Goal in context: Cruise control shall decrement the set speed by 1 mph with each input.
Precondition: The car is in good condition, and the driver has activated cruise control and set the speed.
Trigger: Driver wants to decrease the currently set speed 5 mph.

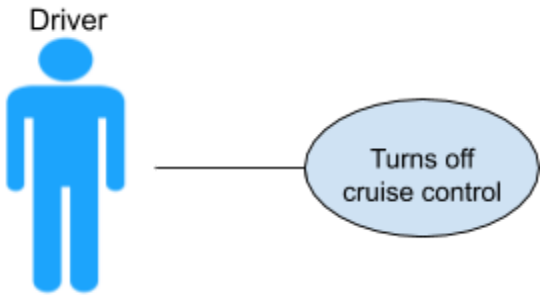
1. While cruise control is activated and currently set at a speed, the driver triggers the decrease speed input on the cruise control interface.
2. Cruise control requests speed values from sensors.
3. Sensors provide approval to set cruise control at the decreased speed.
4. Cruise control logs the decrease speed input and the changes in the speed of the car.
5. Cruise control requests the Engine Management System to decrease the speed by 1 mph.
6. The Engine Management System is set to 1 mph under the original set speed.
7. Cruise control shall display visual feedback of the current set speed to the driver.
8. Speed is continuously reported to the cruise control system.
9. The user repeats step 1 and the cruise control system repeats steps 2-7.

Exception: If the driver is trying to decrease speed under 25 mph, the Engine Management System will not adjust the speed and cruise control shall display visual feedback regarding the speed not being updated.



Use Case 7: Usage of the turn off input
Primary actor: The driver.
Goal in context: Cruise control shall turn off completely on a turn off input
Precondition: The driver has cruise control turned on.
Trigger: Driver wants to turn off cruise control completely.

1. While cruise control is on, the driver triggers the turn off input.
2. If the car is driving at a set speed, the set speed is deactivated and the car decelerates.
3. Cruise control turns off completely.
4. Cruise control gives visual feedback to show that the car is decelerating.
5. Cruise control gives visual feedback to show cruise control has been turned off completely.
6. If cruise control is on, but no set speed has been activated, the cruise control system turns off completely.
7. Cruise control gives visual feedback to show cruise control has been turned off completely.



UML Class-Based Modeling

Cruise Control
Time Current speed
turnOn() setSpeed() increaseSetSpeed() decreaseSetSpeed() accelerateSpeed() disengageSpeed() TurnOff()

UML CRC Model Index Card

Class: turnOn()	
Activates the cruise control system Keeps the cruise control on standby, waiting for set speed Keeps the cruise control on standby, waiting for the increase speed Keeps the cruise control on standby, waiting for the decrease speed Keeps the cruise control on standby, waiting for the cruise control off	Set speed Increase speed Decrease speed Cruise Control Off

Class: turnOff()	
Deactivates the cruise control system Disables the “cruise control on”	Cruise Control On

Class: setSpeed()	
Sets the speed of the cruise control system Sets the new speed to the increased speed that is set by driver Sets the new speed to the decreased speed that is set by driver	Increase speed Decrease speed

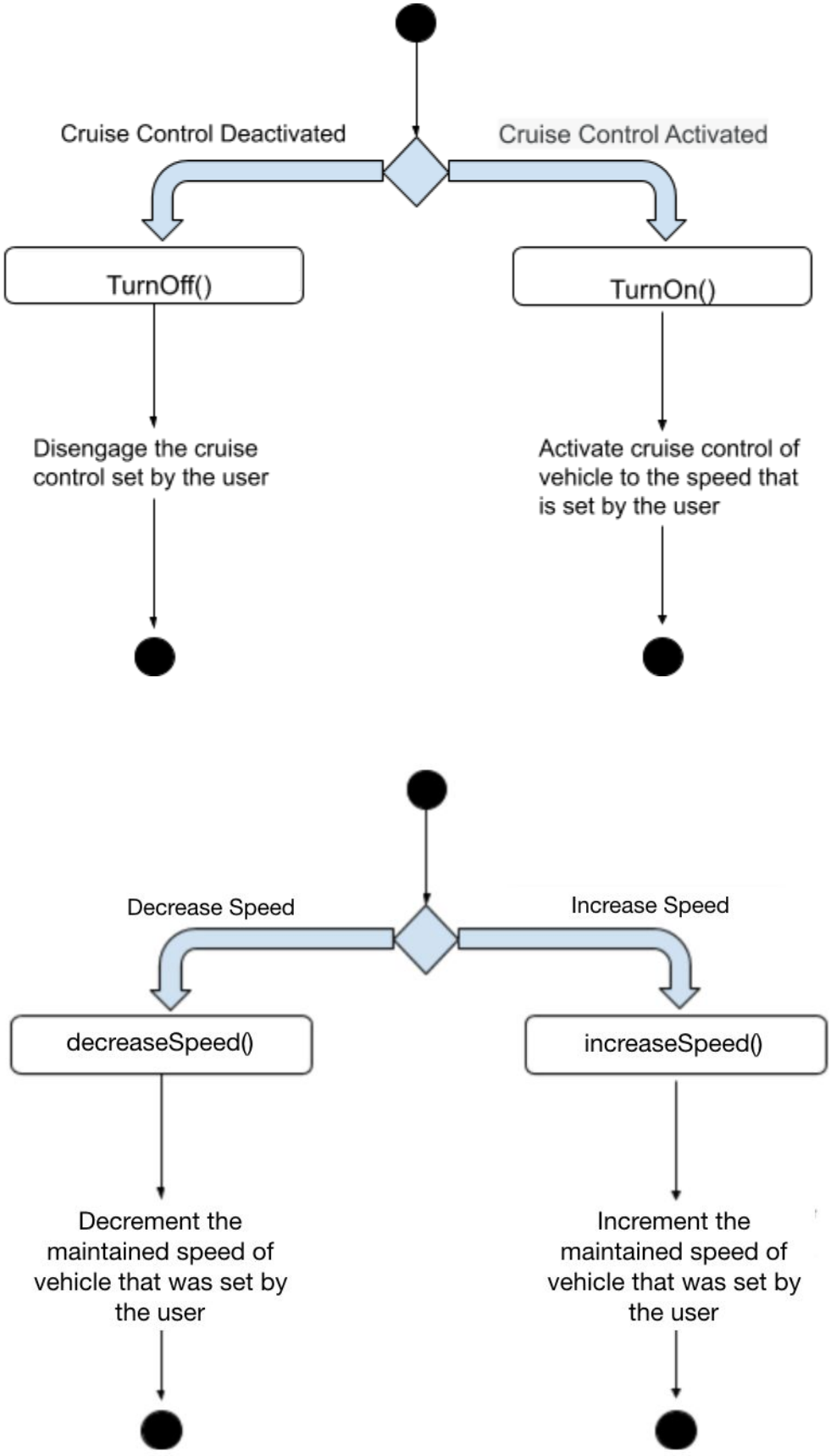
Class: increaseSetSpeed()	
Increases the desired speed that can be set by the driver	

Class: decreaseSetSpeed()	
Decreases the desired speed that can be set by the driver	

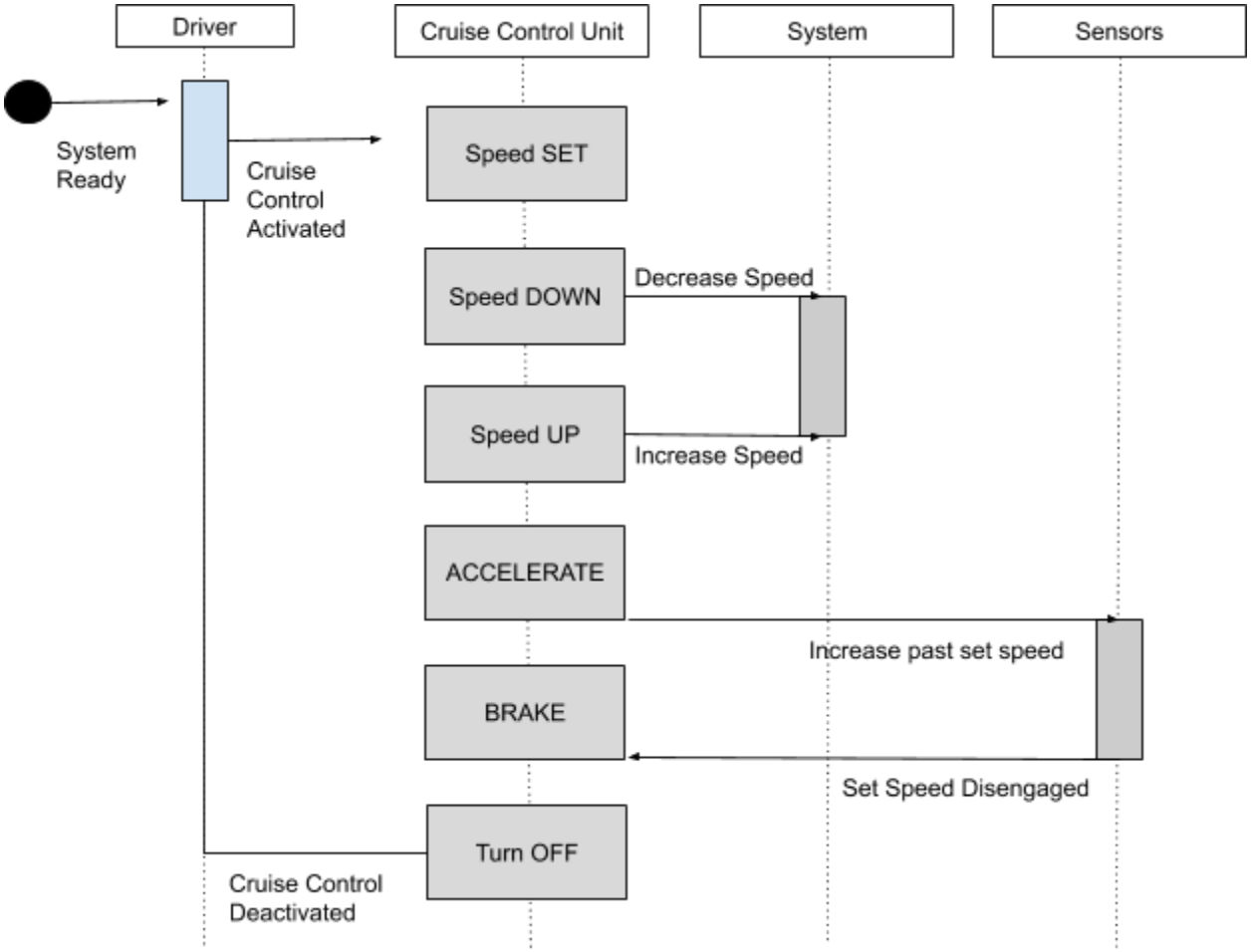
Class: accelerateSpeed()	
Increases the speed of the car Speed is increased past the speed that is set by the cruise control system After the acceleration pedal is released, cruise control shall maintain originally set speed upon deceleration.	Accelerator pedal

Class: disengageSetSpeed()	
Vehicle will start to decelerate, but cruise control will still be active Cruise control system will still stay activated The vehicle will not have the set speed anymore	Brake pedal

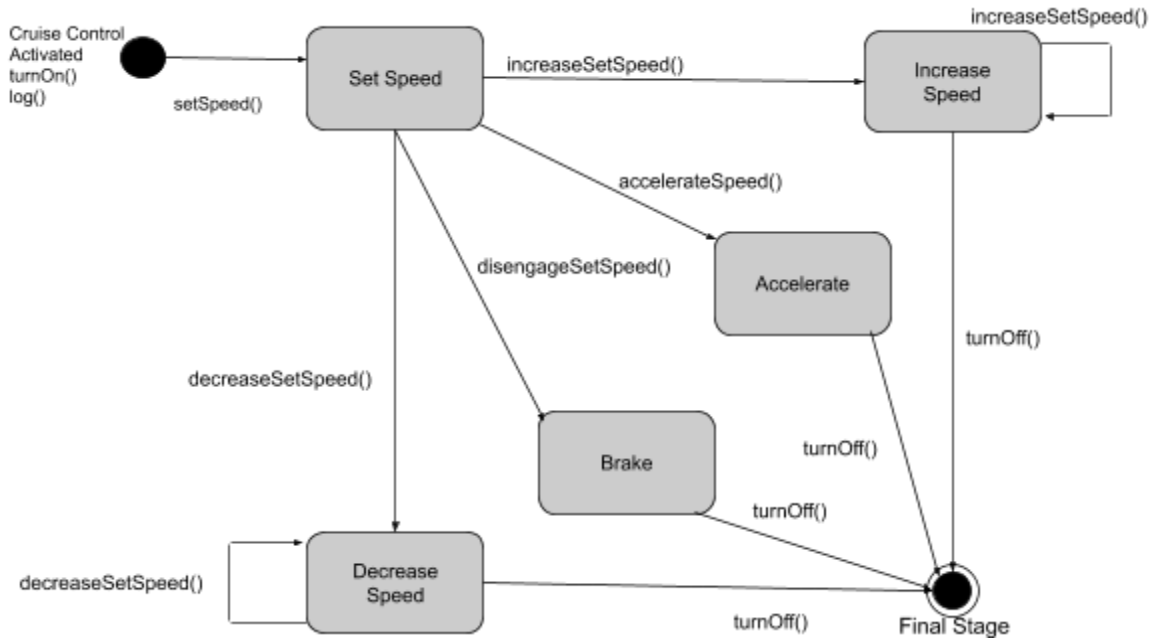
UML Activity Diagram



UML Sequence Diagram



UML State Diagram



5. Cruise Control Software Architecture

Architecture Type: Object-Oriented

While discussing different architectural styles for the Cruise Control Software, our team weighed the pros and cons mainly for the most relevant styles, Object-Oriented and Data Flow Architecture. We found that the Cruise Control system utilizes the concept of a class that contains multiple components, therefore the **Object-Oriented Architecture** is the most suitable and have listed the pros and cons of the styles below.

Object-Oriented Architecture

Pros:

- This architecture allows us to split the multiple components of the cruise control, to clearly state and understand the methods that are only available to the driver (accelerateSpeed(), setSpeed(), etc.) and the methods that are available to the system.
- Creates a flowchart to show how one class interacts with another in an organized manner.

Cons:

- This style does not show the layers of the architecture, such as the application layer and the user interface layer.

An architectural style that we considered that does show the internal layers that Object-Oriented does not, is the Layered architecture style. Although the Layered style shows this information, the Data Flow architecture seemed to be a better style to consider for the Cruise Control system.

Data Flow Architecture

Pros:

- It provides a clear direction and a division between all the possible paths that the pipes can lead to
- It simplifies the system maintenance of the project because it has split all of the filters and pipes
- Supports sequential and parallel execution

Cons:

- Because all of the paths have been laid out, it is hard to work with this architecture in a dynamic way.

Object-Oriented and Data Flow Architecture seemed to be the two most suitable for the Cruise Control system. The Data Flow architecture is suitable for this system because it gives a clear understanding of how one pipe can lead to another filter. It is also easier to follow the paths, and will show how the data that is passed from one filter will lead to another filter. However, the Object-Oriented architecture allows the cruise control system to be visualized in terms of the classes that are working in this cruise control system that represent all of the components of the system. This style also clearly lays out which methods are limited to which class, and the methods unavailable to that class making the Object-Oriented style a better fit for the Cruise Control system.

Components, Connectors, & Constraints

Every architectural style utilizes *components* that perform functions that are required, *connectors* that enable communication between these components, and *constraints* that determine how the components can be used within the system. In object-oriented programming, the components are the necessary data and functions that must be used to manipulate the data. These are our objects, the Driver, Brake, Accelerator, System On/Off, the Set Speed, and Increase/Decrease Speed. The connectors allow communication between components through message passing. The connectors necessary for our cruise control system are the EMS sensor, accelerator and brake sensors, and our user input sensor. The constraint to object-oriented programming is the necessity of classes and interfaces. Each component receives and sends specific data to other components in order to function.

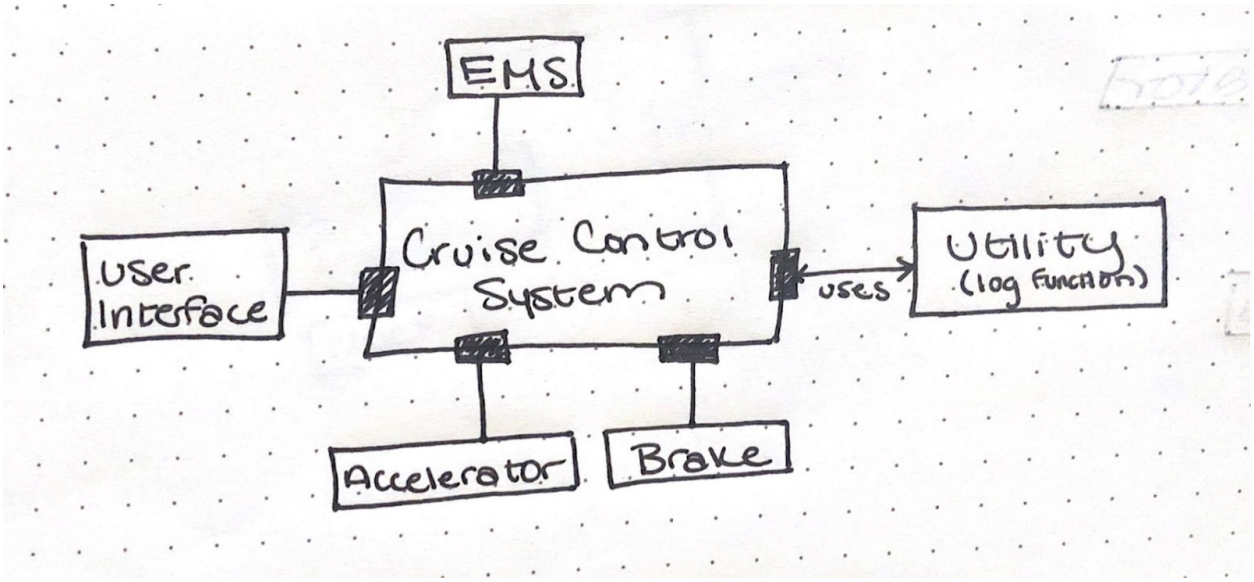
Control Management

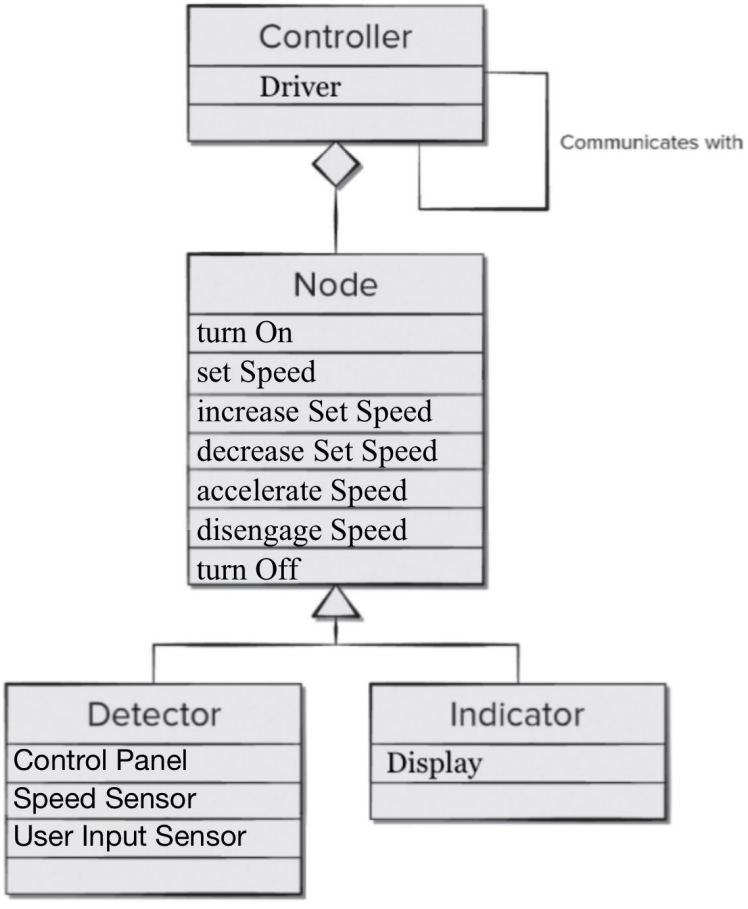
Control is managed by the user first. In the architectures that we are following for this project, the control within the architecture first starts with the input from the user that is interacting with the software. The architecture then states that the control is then passed around the architecture through the uses of the pipes and the methods that are available to the architecture, depending on which architecture is being used. Our system records the time of any change of state in our system on the log. This allows the system itself or a utility worker to properly read the chain of events of the cruise control operating.

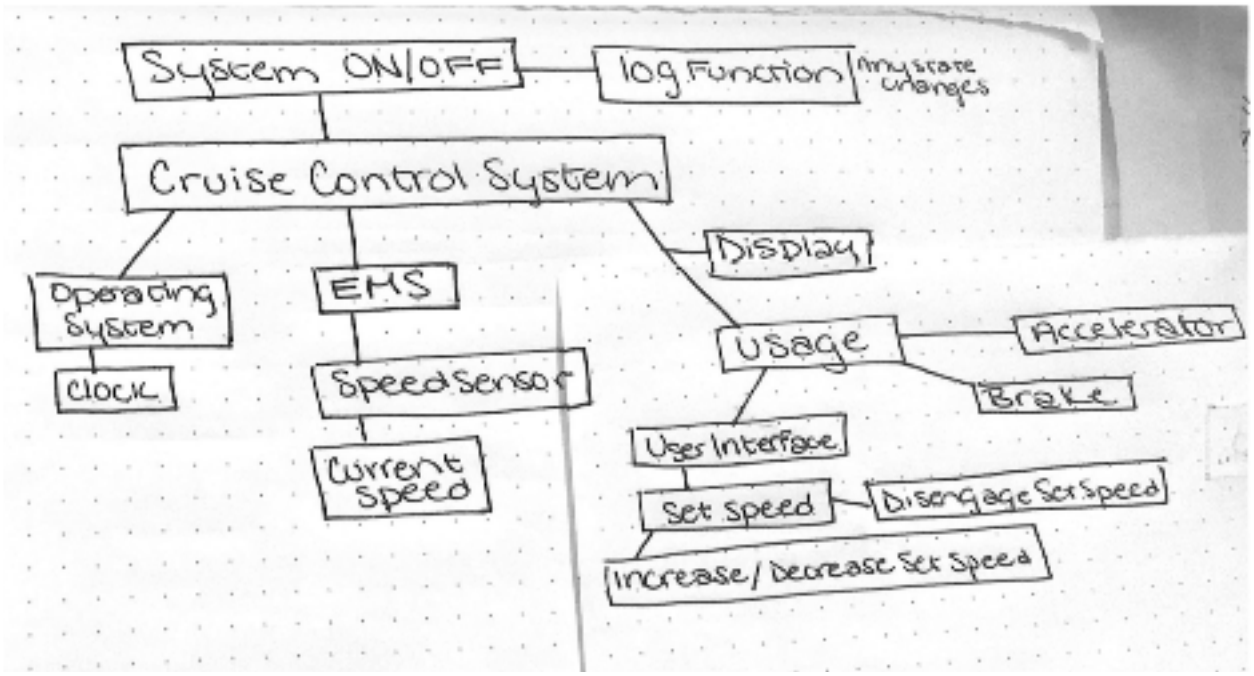
Data Architecture

Data architecture is a set of rules and policies that govern the data that is collected on its usage, storage, and its management within its organization and database systems. With a weak data architecture, it will be more difficult for the program to adapt and enhance.

Diagrams


EMS
Cruise Control System
User Interface
uses
Utility
(log Function)
Accelerator
Brake


Controller
Driver
Communicates with
Node
turn On
set Speed
increase Set Speed
decrease Set Speed
accelerate Speed
disengage Speed
turn Off
Detector
Control Panel
Speed Sensor
User Input Sensor
Indicator
Display


System ON/OFF
log Function
any state changes
Cruise Control System
Operating System
Clock
EMS
Speed Sensor
Current Speed
Display
Usage
Accelerator
Brake
User Interface
Set speed
Disengage Set Speed
Increase / Decrease Set Speed

6. Cruise Control Code

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <errno.h>

/*
todos
1. Fix the accelerate speed so that it allows for release and set speed
       if it releases, it should stedility back to the set speed
       once the set speed input is given, set the speed
2. disengageSetSpeed()
3. logger()
4. Error handling, not very in depth
5. Come up with test cases, test cases need to be as per our requirements in our doc
6. Make list of issues

*/

//Global variables

//This variable will contain the set_speed of the vehicle
int set_speed;

//This
int current_speed;
char state[10];
bool onOff = false;
bool accelerator = true;
bool end_disengage = false;

void turnOn(){
   scanf("%s", state);
```

```
    if (strcmp(state, "start") == 0) {
        onOff = true;
    }
}
void setSpeed(int newSpeed){
    if (newSpeed > 150) {
        printf("Max Speed = 150\n");
        set_speed = 150;
        printf("The new speed is: %d\n\n", set_speed);
    }
    else{
        set_speed = newSpeed;
        printf("The new speed is: %d\n\n", set_speed);
    }
    current_speed = set_speed;
}
void increaseSetSpeed(){
    set_speed++;
    printf("The new speed is: %d\n\n", set_speed);
}
void decreaseSetSpeed(){
    set_speed--;
    printf("The new speed is: %d\n\n", set_speed);
}

void accelerator_helper (int signal, siginfo_t *siginfo, void * context){
    //Listen for the ctrl c signal, print to console the appropriate message
    if(signal == SIGINT) {
        if(accelerator == true){
            accelerator = false;
            printf("Decelerating... \n\n");
            //accelerateSpeed();
        }
        else{
            accelerator = true;
            printf("Accelerating... \n\n");
            //accelerateSpeed();
        }
```

```
        fflush(stdout);
    }
    //Listen for the ctrl z signal, print to console the appropriate message
    else if(signal == SIGTSTP) {
        printf("Setting speed\n\n");
        setSpeed(current_speed);
        fflush(stdout);
    }
}
void accelerateSpeed(){
    struct sigaction act;

    memset (&act, '\0', sizeof(act));

    act.sa_sigaction = &accelerator_helper;

    act.sa_flags = SA_SIGINFO;

    //Handler for SIGINT
    if(sigaction(SIGINT, &act, NULL) != 0) {
        perror("Sigaction error");
        exit(EXIT_FAILURE);
    }

    //Handler for SIGTSTP
    if(sigaction(SIGTSTP, &act, NULL) != 0) {
        perror("Sigaction error");
        exit(EXIT_FAILURE);
    }

    //current_speed = set_speed;
    //printf("Value of acce %d", accelerator);

    while(accelerator == true){
        current_speed++;
        printf("Current speed: %d\n\n", current_speed);
        sleep(1);
    }
```

```
   while(accelerator == false){
       if (current_speed == set_speed) {
           return;
       }
       current_speed--;
       printf("Current speed: %d\n\n", current_speed);
       sleep(1);
   }
}

void disengageSpeed(){
   current_speed = set_speed;
   set_speed = 0;
   while ((end_disengage == false) && current_speed != 0){
       current_speed--;
       printf("The new speed is: %d\n\n", current_speed);
       sleep(2);
   }
}
void turnOff(){
   onOff = false;
}
void logger(char command[]){
   FILE * file;

   //Create a new file if not existing already,
   //appends to file if it exists.
   file = fopen("cruise_control.log", "a");

   //Error check for opening file
   if(file == NULL) {
       perror("Error opening file");
       exit(1);
   }

   //Write to log of commands
   fprintf(file, "%s\n", command);
   fprintf(file, "Current speed: %d\n\n", set_speed);
```

```
    //Close file
    fclose(file);
}
int main(){
    printf("Type 'start' to start.\n\n");

    while (onOff != true) {
        turnOn();
    }

    printf("\nWelcome to cruise control!\n\n");

    //Variable for the input
    char option[256];
    int arguments;

    while (onOff) {
        char option[10];

        printf("Options for cruise control:\n");
        printf("1. Type ss for setting a new speed\n");
        printf("2. Type iss for increasing the speed\n");
        printf("3. Type dss for decreasing the speed\n");
        printf("4. Type as to accelerate\n");
        printf("5. Type ds to disengage\n");
        printf("6. Type off to turn off\n\n");

        arguments = read(0, option, 255);

        if(arguments < 0) {
            //Make an exception for signals
            if (errno == EINTR) {
                continue;
            }
            else{
                printf("Error while reading input.\n");
                exit(1);
            }
        }
```

```
        else {
            //Add the null byte character at the end of array before passing to
functions
            option[arguments - 1] = '\0';

            //Pass to the function to log the commands to cs392_shell.log
            logger(option);
        }



        if (strcmp(option, "ss") == 0){
            char temp[10];
            int temp_value;

            printf("Please give a number.\n\n");

            scanf("%s", temp);

            temp_value = atoi(temp);
            setSpeed(temp_value);
        }

        if (strcmp(option, "iss") == 0){
            increaseSetSpeed();
        }

        if (strcmp(option, "dss") == 0){
            decreaseSetSpeed();
        }

        if (strcmp(option, "as") == 0){
            accelerateSpeed();
        }

        if (strcmp(option, "ds") == 0){
            set_speed = 0;
            while(current_speed > set_speed){
                current_speed = current_speed - 5;
```

```
                printf("Speed is %d\n\n", current_speed);
                sleep(1);
            }
        }

        if (strcmp(option, "off") == 0){
            turnOff();
        }

    }

    return 0;
}
```

7. Cruise Control Tests

1. *Usage of turn on and set speed* *INPUT*---- **“start, ss 30”**
    - Start
        - turns on the cruise control
    - set speed
        - prompt the user for a speed to set the cruise control to

2. *Usage of the brake with cruise control engaged* *INPUT* ---- **“start, ss 30, ds”**
    - Start
        - turns on the cruise control
    - set speed
        - prompt the user for a speed to set the cruise control to
    - brake (disengage set speed)
        - applies brake, resets set speed but doesn’t turn off the cruise control. Car will slow down until the current speed hits 0.

3. *Usage of accelerator with cruise control engaged* *INPUT* --- **“start, ss 30, as, CTRL-C**
    - Start
        - turns on the cruise control
    - set speed
        - prompt the user for a speed to set the cruise control to
    - accelerate/release

        - accelerates until the user has released the accelerator(CTRL-C). Once the user has released the accelerator pedal, the car will decrease speed until it hits set speed.

4. *Set NEW set speed during usage of accelerator* *INPUT* ---- **"start, ss 30, as, CTRL-Z, CTRL-C"**
    - Start
        - turns on the cruise control
    - set speed
        - prompt the user for a speed to set the cruise control to
    - accelerate/set speed/release
        - accelerates until the user has released the accelerator (CTRL-C). If the user chooses to set the speed to current speed while accelerating, they may do so by setting speed again (CTRL-Z). Once the user has released the accelerator pedal, the car will decrease speed until it hits set speed.

5. *Usage of increasing speed while cc on & speed is set* *INPUT* --- **"start, ss 40, iss, iss, iss, iss, iss"**
    - Start
        - turns on the cruise control
    - set speed
        - prompt the user for a speed to set the cruise control to
    - increase set speed
        - increments the set speed by 1mph.

6. *Use of decreasing speed while cc on & speed is set* *INPUT* ---- **"start, ss 40, dss, dss, dss, dss, dss"**
    - Start
        - turns on the cruise control
    - set speed
        - prompt the user for a speed to set the cruise control to
    - decrease set speed
        - decrements the set speed by 1mph.

7. *Usage of the turn off* *INPUT* ---- **"start, ss 30, off"**
    - Start
        - turns on the cruise control
    - set speed
        - prompts the user for a speed to set the cruise control to
    - turn off
        - cruise control turns off

8. Cruise Control Issues

1. For the general use case, the quality attributes are the availability rate and the reliability rate of the cruise control system. For example, use case 1 follows through the user turning on the cruise control and setting the speed. The quality attributes in this case will be the reliability rate of the functions that the user is trying to use, such as the cruise control turning on, turning off, and the success of the cruise control being able to set the speed and relay that data to the cruise control unit. All the measurable attributes of this system will be mainly the reliability and the availability of the system overall in a given time period.

2. The role of a system architecture is to map out the structure and the organization of the project. In our case of the cruise control project, we decided that the object-oriented architecture and the data flow architecture seemed the most applicable to our project. The object-oriented architecture will map out how objects in the class relate to each other, or access each other through the given class methods. The data flow architecture will map out how the filters pass around data through the use of pipes. And by observing how these components of the architecture are connected to each other, it will show what each component is responsible for, which in turn will show what exactly are the requirements of this project.

3. We identified the microkernel architecture pattern that is being used for this system, which is similar to our system's architecture in that there is an entry point to the cruise control system, which is equivalent to the initial turning on input from the user. Once the turn on input has been initiated by the user, that is when the architecture can then follow the flow of the architecture and move on to the different aspects and modules available for use by the architecture, such as the set speed and the increase/decrease speed. This architecture pattern is very fluid, which is how the structure of our project is.

4. With our implementation of the microkernel pattern for our project, when making changes in the plug-in modules, it will minimize the changes in the central system, which is useful in our project because like demonstrated in use case 2, there might be a situation in which the driver would try to break, but this change in module is not supposed to alter the main system. Also, using the microkernel pattern benefits as it opens the possibility of testing

each module individually, which shown through our multiple test cases, it is crucial to test every function individually to make sure that all of them work.

5. Some issues that we found from the architecture pattern that was used in our design was that we do not fully take advantage of the plug-in feature that is mainly used in the microkernel pattern. While we envisioned that in the future, we will add features to the project other than the core cruise control features that we already have (on, off, set speed, increase speed, decrease speed), we do not currently have any plug-ins. As for quality issues in our project, our defects exist in the availability and reliability of the cruise control unit, because it is not 100% available and reliable at all times.

6. The Formal Review allowed our group to identify multiple minor errors involving the specificity, and wording or phrasing of certain details that need to be corrected. These corrections involve minor details added by members of the team.

7. A major issue that we found involved our Class-based modeling. We did not properly name our classes, giving them names of functions. This issue also involves not showing the relationship between these classes. To solve this, our group must rename our classes to better develop our diagram, showing relationships.

8. We faced a few issues during the development of the accelerateSpeed function while coding. Upon revisit, we were able to complete the function so that it allows for release and set speed if it releases, while stabley returning to previous set speed, if the set speed input is given, it sets the speed.

9. We originally could not figure out what exactly needed to be tested. Upon revisit, we were able to come up with the test cases by working from the requirements listed in the document including the use cases.