After you have chosen a topic, create a brief software requirements specification (SRS) document for the project. The SRS document should contain at least the following sections:

* Introduction (with purpose and scope)
* Overall description (with product perspective, product functions, and user characteristics)
* Specific requirements

**Introduction**

This is a computer car racing game (Car Racer) which can be playing by many users. Racing track has some obstacles that need to avoid.

Description of the game:

Car Racer is a simple racing game where players control a car using keyboard inputs and navigate through predefine track while avoiding obstacles and collecting points. Car images can be created separately and import to the game.

Here's a basic outline of how the game can be coded using Python:

1. Set up the game window and import necessary libraries (e.g., Pygame).

2. Create a car object and set its initial position on the screen.

3. Create a track with different obstacles and points using rectangles or images.

4. Implement keyboard controls to move the car left or right.

5. Add collision detection to check if the car hits any obstacles or collects points.

6. Keep track of the player's score and display it on the screen.

7. Add a timer to measure the player's time to complete the track.

8. Implement game over conditions, such as running out of time or hitting too many obstacles.

9. Display a game over screen with the final score and an option to restart the game.

10. Add sound effects and background music to enhance the gaming experience.

This is a basic framework for the game,

**Improvement ideas:**

* If it hits a obstacles speed will be reduce by half
* When the cars are very close , back car can get speed boost if DRS Zone is define.
* If car overtakes by cutting the path, it has to give the position back
* If car cuts the path more than 3 times, it reduce the speed by half
* Add many cars on the race

Python libraries like Pygame make it relatively easy to create 2D games involving cars. With a little creativity and coding, you can bring the Car Racer game to life!

import pygame

import random

# Step 1: Set up the game window and import necessary libraries

pygame.init()

window\_width = 800

window\_height = 600

game\_window = pygame.display.set\_mode((window\_width, window\_height))

pygame.display.set\_caption("Car Game")

# Step 2: Create a car object and set its initial position on the screen

car\_image = pygame.image.load("car.png")

car\_width = 50

car\_height = 100

car\_x = window\_width // 2 - car\_width // 2

car\_y = window\_height - car\_height - 10

car\_speed = 5

# Step 3: Create a track with different obstacles and points

obstacle\_width = 50

obstacle\_height = 50

obstacle\_x = random.randint(0, window\_width - obstacle\_width)

obstacle\_y = -obstacle\_height

obstacle\_speed = 3

point\_width = 30

point\_height = 30

point\_x = random.randint(0, window\_width - point\_width)

point\_y = -point\_height

point\_speed = 2

# Step 4: Implement keyboard controls to move the car left or right

def move\_car(direction):

if direction == "left":

car\_x -= car\_speed

elif direction == "right":

car\_x += car\_speed

# Step 5: Add collision detection to check if the car hits any obstacles or collects points

def check\_collision():

global car\_x, car\_y, obstacle\_x, obstacle\_y, point\_x, point\_y obstacle\_rect = pygame.Rect(obstacle\_x, obstacle\_y, obstacle\_width, obstacle\_height)

point\_rect = pygame.Rect(point\_x, point\_y, point\_width, point\_height) car\_rect = pygame.Rect(car\_x, car\_y, car\_width, car\_height)

if car\_rect.colliderect(obstacle\_rect):

return "obstacle"

elif car\_rect.colliderect(point\_rect):

return "point"

else:

return None

# Step 6: Keep track of the player's score and display it on the screen

score = 0

font = pygame.font.Font(None, 36)

# Step 7: Add a timer to measure the player's time to complete the track

clock = pygame.time.Clock()

start\_time = pygame.time.get\_ticks()

time\_limit = 60000 # 60 seconds

# Step 8: Implement game over conditions

game\_over = False

# Step 10: Add sound effects and background music

pygame.mixer.music.load("background\_music.mp3")

pygame.mixer.music.play(-1) # -1 means play the music on loop

collision\_sound = pygame.mixer.Sound("collision\_sound.wav")

point\_sound = pygame.mixer.Sound("point\_sound.wav")

while not game\_over:

for event in pygame.event.get():

if event.type == pygame.QUIT:

game\_over = True

elif event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

move\_car("left")

elif event.key == pygame.K\_RIGHT:

move\_car("right")

game\_window.fill((255, 255, 255))

# Move obstacles and points

obstacle\_y += obstacle\_speed

point\_y += point\_speed

# Check if the car hits any obstacles or collects points

collision\_result = check\_collision()

if collision\_result == "obstacle":

game\_over = True

elif collision\_result == "point":

score += 1

point\_x = random.randint(0, window\_width - point\_width)

point\_y = -point\_height

# Draw car, obstacles, and points on the screen

game\_window.blit(car\_image, (car\_x, car\_y))

pygame.draw.rect(game\_window, (255, 0, 0), (obstacle\_x, obstacle\_y, obstacle\_width, obstacle\_height))

pygame.draw.rect(game\_window, (0, 255, 0), (point\_x, point\_y, point\_width, point\_height))

# Display score on the screen

score\_text = font.render("Score: " + str(score), True, (0, 0, 0))

game\_window.blit(score\_text, (10, 10))

# Display timer on the screen

elapsed\_time = pygame.time.get\_ticks() - start\_time

remaining\_time = max(0, time\_limit - elapsed\_time)

timer\_text = font.render("Time: " + str(remaining\_time // 1000), True, (0, 0, 0))

game\_window.blit(timer\_text, (window\_width - 100, 10))

# Check game over conditions

if remaining\_time <= 0:

game\_over = True

pygame.display.update()

clock.tick(60)

# Step 9: Display a game over screen with the final score and an option to restart the game

game\_over\_text = font.render("Game Over", True, (0, 0, 0))

score\_text = font.render("Final Score: " + str(score), True, (0, 0, 0))

restart\_text = font.render("Press R to Restart", True, (0, 0, 0))

while True:

game\_window.fill((255, 255, 255))

game\_window.blit(game\_over\_text, (window\_width // 2 - 100, window\_height // 2 - 50))

game\_window.blit(score\_text, (window\_width // 2 - 100, window\_height // 2))

game\_window.blit(restart\_text, (window\_width // 2 - 120, window\_height // 2 + 50))

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_r:

# Restart the game

car\_x = window\_width // 2 - car\_width // 2

car\_y = window\_height - car\_height – 10

obstacle\_x = random.randint(0, window\_width - obstacle\_width)

obstacle\_y = -obstacle\_height

point\_x = random.randint(0, window\_width - point\_width)

point\_y = -point\_height

score = 0

start\_time = pygame.time.get\_ticks()

game\_over = False

pygame.mixer.music.play(-1)

# Restart the background music

break

Make sure to replace the file names `"car.png"`, `"background\_music.mp3"`, `"collision\_sound.wav"`, and `"point\_sound.wav"` with the appropriate file names for your game.