Functional Requirements Document  
Car Racing Game Prototype

Contents

[1. Introduction 1](#_Toc193956773)

[1.1 Purpose 1](#_Toc193956774)

[1.2 Scope 2](#_Toc193956775)

[1.3 Document Structure 2](#_Toc193956776)

[2. Key Functional Requirements 2](#_Toc193956777)

[2.1 Cars Move Along the Track and Determine the Winner 2](#_Toc193956778)

[2.2 Car Speed Influenced by Multiple Factors 2](#_Toc193956779)

[2.3 Obstacles Affect Race Dynamics 2](#_Toc193956780)

[2.4 Pit Stop (Planned Feature) 3](#_Toc193956781)

[2.5 Race Management and Results Calculation 3](#_Toc193956782)

[3. Non-Functional Requirements 3](#_Toc193956783)

[3.1 Visual Representation 3](#_Toc193956784)

[3.2 Randomized Data 3](#_Toc193956785)

[3.3 Scalability 3](#_Toc193956786)

[3.4 Performance 3](#_Toc193956787)

[4. Conclusion 3](#_Toc193956788)

# 1. Introduction

## 1.1 Purpose

This document defines the functional requirements for the Car Racing Game Prototype. The game simulates a car race with dynamic elements affecting gameplay. Main objectives include:  
• Enabling cars to move between multiple stops on the track.  
• Calculating total race time for each car while introducing randomness for realism.  
• Visually displaying the race process and determining the winner.

## 1.2 Scope

This document outlines the primary functional aspects of the game prototype built in Java using JavaFX. It covers UI components, game logic, and model structures. This version does not include future features like pit stop logic or driver strategy.

## 1.3 Document Structure

• Section 2: Key functional requirements  
• Section 3: Non-functional requirements  
• Section 4: Summary and future direction

# 2. Key Functional Requirements

## 2.1 Cars Move Along the Track and Determine the Winner

Description  
• Two cars race from a unique start stop to a unique end stop.  
• Each track consists of multiple stops and all stops must be visited.  
• The system calculates total race time and declares the fastest car as the winner.  
System Behavior  
• On start, cars begin moving along predefined paths.  
• Each car’s speed and position are updated per frame.  
• Winner is determined based on total completion time.

## 2.2 Car Speed Influenced by Multiple Factors

Description Car performance depends on:  
• Engine: Determines acceleration and max speed (e.g., V6, V8).  
• Wheel: Influences acceleration and stability (e.g., Standard, Carbon Fiber).  
System Behavior  
• Speed updates every frame based on acceleration.  
• Max speed is constrained by the engine.  
• Wheel type impacts performance on turns or obstacles.

## 2.3 Obstacles Affect Race Dynamics

Description Obstacles appear on the track and affect performance:  
• SlipperyObstacle (e.g., oil spill): may cause sliding.  
• Positions are randomly generated per track.  
System Behavior  
• Collision with obstacle triggers slide animation.  
• Speed reduction is applied, and status is logged in UI.  
• Obstacles have icons (e.g., cones) to indicate position.

## 2.4 Pit Stop (Planned Feature)

Note: Pit stops for fuel, tires, and repairs are planned as future extensions, not in the current version.

## 2.5 Race Management and Results Calculation

Description  
• Game flow is managed by RaceController, which controls per-frame updates, obstacle handling, and completion checks.  
• Core game data (track, cars, results) is stored in a shared GameData model.  
• Completion times and winners are calculated automatically.  
System Behavior  
• RaceController pulls from GameData and updates car position and speed.  
• Race ends when both cars reach destination.  
• UI displays result logs and winner.

# 3. Non-Functional Requirements

## 3.1 Visual Representation

• Includes 3 UI screens: Difficulty, Car Selection, Race View.  
• Custom images for cars, obstacles, and tracks.  
• Background image enhances immersion.

## 3.2 Randomized Data

• Obstacle placement is randomized.  
• Car starting and ending stops vary by game run.

## 3.3 Scalability

• Codebase supports new engines, wheels, and obstacles.  
• Future expansion can include weather and driver logic.

## 3.4 Performance

• JavaFX Timeline ensures smooth rendering.  
• UI updates occur in real-time without lag.

# 4. Conclusion

This document summarizes the Car Racing Game Prototype, developed with JavaFX and MVC principles. Current features include dual-car racing, obstacle logic, audio support (AudioController), and a complete multi-stage user interface. Future extensions may involve pit stops, weather, and analytics.