CAR RACING GAME

A PROJECT REPORT
for
Mini Project
Session (2024-25)

Submitted by
SATYJEET KUMAR-202410116100187
SAURABH KUMAR-202410116100188
SANKET PUNDHIR-202410116100183
SANIDHYA GARG-202410116100182

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr .Vipin Kumar
Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(April - 2025)

CERTIFICATE

Certified that SATYJEET KUMAR (202410116100187), SAURABH KUMAR (202410116100188), SANIDHYA GARG (202410116100182), and SANKET PUNDHIR (202410116100183) have carried out the project work titled "Car Racing Game" (Mini Project-II) for the Master of Computer Application program at Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow, under my supervision. The project report embodies original work and studies carried out by the students themselves. The contents of the project report do not form the basis for the award of any other degree to the candidates or to anybody else from this or any other University/Institution.

Dr. Vipin Kumar
Assistant Professor

Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Akash Rajak
Dean

Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

The **Car Racing Game** is a Java-based mini project developed as part of the curriculum for the Master of Computer Applications (MCA) program. This project aims to provide users with an engaging and interactive experience of a virtual racing environment while demonstrating fundamental concepts of object-oriented programming, graphical user interface (GUI) design, event handling, and game logic development. The game features a 2D top-down racing interface where players control a car, navigating through a dynamically scrolling track while avoiding obstacles and other vehicles. The core mechanics involve collision detection, speed control, scoring systems, and level progression. The design emphasizes real-time responsiveness and smooth gameplay, achieved through multithreading and optimized rendering techniques using Java's built-in libraries such as AWT and Swing. Beyond entertainment, the project also serves as an educational tool to understand core programming concepts and the development cycle of interactive applications. It showcases the integration of logic, graphics, and user interaction in a cohesive structure, making it a valuable learning experience for aspiring developers.This game demonstrates the application of technical skills in a practical scenario and highlights the potential for further enhancement, such as the inclusion of multiplayer functionality, advanced AI opponents, or mobile platform adaptation.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Vipin Kumar** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Akash Rajak, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Satyjeet Kumar
Sanket Pundhir
Saurabh Kumar
Sanidhya Garg

TABLE OF CONTENTS

Chapter 1
INTRODUCTION

1.1 Project Description

The **Car Racing Game** is a Java-based 2D racing game developed using the **Swing** and **AWT** libraries. This desktop application offers a simple yet engaging user experience, where the player maneuvers a car along a vertical scrolling road while avoiding randomly generated opponent vehicles. The primary goal of the game is to survive as long as possible while accumulating points, with the game's difficulty increasing progressively over time.

The game begins with a start screen, followed by real-time gameplay where the player uses the left and right arrow keys to navigate the car between predefined lanes. Opponent cars appear from the top of the screen at random intervals and speeds, simulating traffic on a busy road. A collision between the player's car and any opponent vehicle ends the game, triggering a game- over prompt with the option to restart or exit.

This project serves as a foundational introduction to game development using Java. It effectively demonstrates concepts such as:

- **GUI design** using Swing components (`JPanel`, `JFrame`, `JButton`)
- **Keyboard event handling** via the `KeyListener` interface
- **Graphics rendering** using the `paintComponent()` method
- **Timer-based animations** with `javax.swing.Timer`
- **Object movement and collision detection** using geometric shapes

Two versions of the game are implemented:

- `CarRacingGame.java`: A functional version with difficulty selection.
- `ImproveGame.java`: An enhanced version with improved UI, smoother road animation, and better collision logic.

The project aims to provide an interactive platform to apply object-oriented programming principles, understand real-time user interaction, and gain hands-on experience in developing a Java-based GUI application. It lays the groundwork for future enhancements such as sound effects, high score tracking, and multiplayer capabilities.

1.2 Project Scope

The scope of the **Car Racing Game** project encompasses the design, development, and testing of a basic 2D racing game using **Java** and **Swing**. This project focuses on building a fully functional standalone desktop application that demonstrates core game development principles and Java GUI programming skills.

In Scope:

1. Development of a 2D car racing game interface using Java Swing.
2. Implementation of player controls using keyboard input.
3. Real-time game loop and movement of opponent vehicles.
4. Collision detection between the player's car and opponent cars.
5. Scoring system based on survival time or distance covered.
6. Game over and restart functionalities.
7. Difficulty scaling with score progression (speed increase).
8. Start screen and game over screen.
9. Basic UI elements (buttons, labels, background graphics).

Out of Scope

10. Multiplayer functionality (local or online).
11. Use of 3D graphics or advanced physics engines.
12. Mobile or web-based version of the game.
13. High score persistence or database integration.
14. Audio effects or background music.

Future Scope:

Although not part of the current version, the following features can be considered for future enhancements:

a. Adding sound effects and background music.
b. Introducing multiple levels or game modes.
c. Saving and displaying high scores.
d. Mobile adaptation using Android SDK or JavaFX.
e. Power-ups, fuel meters, or additional obstacles

1.3 Project Overview

The **Car Racing Game** is a desktop-based 2D game developed using **Java** and the **Swing GUI toolkit**. The game is designed to simulate a fast-paced car racing environment where the player must avoid incoming opponent vehicles while navigating a vertically scrolling road. The objective is to score as high as possible by surviving longer without collisions.

This project serves both as an educational tool to understand Java programming concepts and as an entertainment application. It emphasizes **real-time gameplay mechanics**, **responsive user input**, and **graphical rendering**, making it a compact but complete example of a Java-based game application.

The game architecture includes:

A **main game loop** that updates the game state and redraws graphics.
Event listeners to capture and respond to user input.
Timers to manage opponent car movement and screen refresh rates.

Chapter 2

Feasibility Study

The feasibility study for the **Car Racing Game** project evaluates the practicality of developing and deploying the game in terms of technical, operational, economic, and schedule factors. The goal is to ensure that the project is viable and achievable with the available resources, tools, and time constraints.

1. Technical Feasibility

The project is technically feasible as it relies on widely-used and well-documented technologies such as:

1. **Java SE**: A robust programming language with extensive support for GUI and game development.
2. **Swing & AWT**: Native libraries for creating graphical user interfaces and handling events.
3. **Standard IDEs (e.g., IntelliJ, Eclipse)**: Readily available environments for Java development.

All necessary development tools are open-source or freely available, making technical implementation achievable even on basic hardware.

2. Operational Feasibility

The game is user-friendly, easy to operate, and provides immediate feedback through score tracking and game over prompts. It requires minimal setup to run, making it feasible for end-users or evaluators to operate without technical training. Restart and difficulty selection features enhance usability and gameplay experience.

3. Economic Feasibility

The cost of development is minimal, as the entire game is developed using free tools:

4. Java Development Kit (JDK)
5. IntelliJ IDEA Community Edition / Eclipse
6. No paid libraries or hardware are needed

This makes the project **economically viable** for individual developers or students.

4. Schedule Feasibility

The project was completed within a manageable time frame, typically spanning a few weeks. The development was broken down into:

- Planning and Design
- Core Mechanics Implementation
- UI Enhancements and Event Handling
- Testing and Debugging
- Documentation

Chapter 3
Project Objective

The primary objective of this project is to develop a **Car Racing Game** using **Java programming language**, focusing on enhancing programming skills, game design, and interactive graphical interface development. The project will serve as both a learning tool and a platform for showcasing advanced concepts in Java, including object-oriented programming (OOP), game mechanics, event handling, GUI development, and real-time rendering.

Detailed Objectives:

1. **Game Design and Concept**:
    - Develop a dynamic racing environment where players can control a car and race against opponents.
    - Implement various track designs, environments, and levels to create an engaging experience for users.
    - Allow for single-player or multiplayer modes where players can race against AI-controlled cars or other players.
    - Provide different difficulty levels that adjust the speed and intelligence of opponent cars.
    - Include features like speed boosts, obstacles, and power-ups that add complexity to the game.
2. **Object-Oriented Programming (OOP)**:
    - Utilize OOP principles like **inheritance**, **polymorphism**, **encapsulation**, and **abstraction** to organize game entities (e.g., cars, tracks, power-ups).
    - Design **Car** and **Track** classes, with appropriate attributes (e.g., speed, handling, position) and methods (e.g., accelerate, brake, turn).
    - Implement **GameManager** and **RaceController** classes to manage the game's flow, including starting a race, keeping track of the score, and controlling the racing session.
3. **Graphical User Interface (GUI)**:
    - Use Java's **Swing** or **JavaFX** to create an interactive and responsive user interface.
    - Design screens for the main menu, race track selection, gameplay, and result screen.
    - Provide visual feedback such as score display, lap time, player health, and racing position.
    - Create animations for car movement, race track rendering, and game transitions.
4. **Real-Time Rendering**:
    - Implement continuous, smooth rendering for the movement of cars, obstacles, and environment features during the race.
    - Utilize **Java 2D Graphics API** for rendering the race track, cars, and other dynamic objects.
    - Optimize game performance for smooth gameplay, ensuring that the game runs at a steady frame rate even with multiple moving objects on the screen.
5. **Event Handling and Player Interaction**:
    - Capture player inputs using **keyboard events** (e.g., arrow keys or WASD) for car movement and other controls (e.g., brakes, boosts).
    - Implement **mouse events** for menu selections and other interactive elements.
    - Handle real-time events, such as collisions, car acceleration, and power-ups, in a responsive manner.
6. **AI Opponents and Difficulty Levels**:
    - Design AI-controlled cars that follow the track and adapt to the player's speed and behavior.
    - Implement various difficulty levels by adjusting the speed, reaction time, and pathfinding of AI-controlled cars.
    - Develop algorithms for AI decision-making, such as avoiding obstacles, overtaking the player, and taking corners efficiently.

7. **Game Physics**:
    - Incorporate basic physics principles to simulate car movement, acceleration, and deceleration.
    - Add collision detection and response (e.g., handling the car's behavior when it hits an obstacle or another car).
    - Implement realistic car handling, such as varying levels of friction on different surfaces (e.g., tarmac, grass, mud).
8. **Sound Effects and Music**:
    - Integrate sound effects for car movement, collisions, and other in-game events (e.g., engine sound, tire screeching).
    - Include background music that enhances the racing atmosphere.
    - Allow for the user to toggle sound settings or adjust volume levels.
9. **Scorekeeping and Leaderboards**:
    - Implement a system to track player performance, including lap times, race completion times, and position in the race.
    - Display scores and rank players based on their race results.
    - Store high scores or allow for leaderboard functionality to compare performances across different game sessions.
10. **Testing and Debugging**:
    - Perform extensive testing to ensure all game mechanics work as expected (e.g., car movement, collision detection, event handling).
    - Debug and refine the game's performance to eliminate any lag or graphical issues.
    - Ensure that the game is stable and free from crashes or unexpected behaviors.
11. **Documentation and Reporting**:
    - Create detailed documentation covering the game's design, architecture, and code implementation.
    - Write a comprehensive report describing the entire game development process, including challenges faced and solutions implemented.

Provide user instructions, explaining how to play the game, con

Chapter 4
Hardware and Software Requirement

Hardware Requirements:

1. **Processor**: Intel Core i3 (minimum), Intel Core i5 (recommended).
2. **RAM**: 4 GB (minimum), 8 GB (recommended).
3. **Graphics Card**: Integrated graphics (minimum), Dedicated GPU (recommended).
4. **Storage**: 1 GB free (minimum), 5 GB free (recommended).
5. **Display**: 1280 x 720 resolution (minimum), 1920 x 1080 resolution (recommended).
6. **Operating System**: Windows 7/macOS 10.10 (minimum), Windows 10/macOS 10.15 (recommended).

Software Requirements:

1. **Java Development Kit (JDK)**: Version 8 (minimum), JDK 17 (recommended).
2. **IDE**: Eclipse, IntelliJ IDEA, or NetBeans.
3. **GUI Framework**: JavaFX or Swing.
4. **Graphics Software**: GIMP (minimum), Photoshop (recommended).
5. **Sound Software**: Audacity (minimum), Adobe Audition (recommended).
6. **Version Control**: Git (GitHub/GitLab).
7. **Database (optional)**: SQLite, MySQL (for storing high scores).

Chapter 5
Project Flow

The project flow for developing the **Car Racing Game** can be broken down into several phases, starting from planning to final testing and documentation. Below is the detailed flow of the project:

1. Project Planning and Requirement Analysis

Objective: Define the scope and gather requirements for the game.

- **Game Concept**: Decide on the type of racing game (e.g., arcade-style, realistic, single-player, multiplayer).
- **Game Features**: List key features such as track designs, car mechanics, AI opponents, power-ups, score system, levels, and menus.
- **Tools and Technologies**: Finalize the hardware and software requirements, IDEs (e.g., IntelliJ IDEA, Eclipse), libraries (JavaFX or Swing), and tools for asset creation (e.g., GIMP, Audacity).
- **Target Audience**: Determine who will play the game, which may guide your design choices.
- **Timeline**: Set milestones and deadlines for different phases of the development.

2. Game Design

Objective: Design the game mechanics, UI layout, and flow.

- **Track Design**: Create rough sketches or wireframes of the race tracks. Decide if there will be multiple tracks and if these will vary in complexity.
- **Car Design**: Design car models (2D sprites or simple images) and define car behaviors (e.g., speed, acceleration, handling).
- **AI Design**: Plan how AI will behave on the track (e.g., speed, intelligence, pathfinding).
- **User Interface (UI)**: Design the layout for the game menus, score display, pause screen, and HUD.
- **Game Flow Diagram**: Develop a flowchart outlining the sequence of events (e.g., main menu → race selection → race start → game over/score screen).

3. Development Setup

Objective: Set up the development environment and basic project structure.

- **Create the Project**: Set up the project structure in the chosen IDE (e.g., IntelliJ, Eclipse).
- **Version Control**: Initialize Git for version control and create a repository on GitHub/GitLab for project management and collaboration.
- **Dependencies**: Install and configure necessary libraries (e.g., JavaFX or Swing for UI, JUnit for testing).

4. Core Game Development

Objective: Implement the core features of the game.

a. Car Class and Mechanics

- Implement the **Car class**, defining properties like position, speed, acceleration, and handling.
- Implement car controls (e.g., acceleration, deceleration, steering) using **keyboard inputs**.
- Add movement logic to update the car's position on the screen as per user inputs.

b. Track and Environment

- Create the **Track class** to define the race track layout, including track boundaries, obstacles, and checkpoints.
- Implement collision detection (e.g., if the car goes off-track or hits an obstacle).
- Design different terrains (e.g., road, grass) with varied friction and effects on car speed.

c. AI Opponents

- Implement the **AI class** for AI-controlled cars with pathfinding algorithms (e.g., following the track, avoiding obstacles).
- Implement different levels of difficulty by adjusting AI speed, decision-making, and responsiveness.

d. Graphics Rendering

- Use **JavaFX** or **Swing** to render the track, cars, and other elements in real-time.
- Implement smooth rendering for car movement and track updates.
- Design animations for car actions (e.g., turning, accelerating, braking).

e. Game Loop and Event Handling

- Implement the **game loop** to control the game's flow, handling updates for each frame (e.g., car movement, AI updates, rendering).
- Handle keyboard and mouse events for player interaction (e.g., car control, menu navigation).

5. User Interface (UI) and Menus

Objective: Create the graphical user interface and menu system.

- **Main Menu**: Implement the main menu with options to start a new game, select difficulty, view instructions, and quit the game.
- **Game HUD**: Display real-time information on screen, such as the player's position, speed, lap time, and remaining laps.
- **Pause Menu**: Allow players to pause the game and view options like resume, restart, or quit.
- **Results Screen**: Show the final score, lap times, and rankings after the race ends.

6. Sound and Music

Objective: Add sound effects and background music to enhance the gameplay experience.

- **Sound Effects**: Implement sound effects for car movement (engine, tires screeching, brakes), collisions, power-ups, and other in-game events.
- **Background Music**: Add background music during gameplay and the menu screens.
- **Sound Control**: Provide options to mute or adjust the volume of sounds and music.

7. Scoring and Leaderboards

Objective: Implement a scoring system and display player rankings.

- **Score Calculation**: Track and calculate the player's race time, position, lap times, and any bonuses (e.g., power-ups).
- **Leaderboards**: Store high scores and rankings, either in a local file or a database (optional).
- **Display Scores**: Show the current score, lap times, and position during the race and at the results screen.

8. Testing and Debugging

Objective: Test and refine the game for performance and stability.

- **Unit Testing**: Write unit tests for core functionalities like car movement, collision detection, and AI behavior.
- **Debugging**: Identify and fix any bugs or glitches (e.g., car physics, rendering issues, unresponsive controls).
- **Playtesting**: Test the game in different scenarios to check gameplay balance, AI difficulty, and track design.
- **Performance Optimization**: Ensure smooth gameplay by optimizing the game loop, rendering, and asset loading.

9. Documentation

Objective: Document the game design, development process, and usage.

- **User Manual**: Write instructions on how to play the game, control the car, and understand the scoring system.
- **Developer Documentation**: Document the game's architecture, key classes, and code implementation.
- **Testing Report**: Document any testing procedures, test results, and bug fixes.

1. Final Release

Objective: Finalize the game and prepare for release.

- **Build and Package**: Create an executable JAR file for the game that can be run on any system with Java installed.
- **Versioning**: Assign a version number (e.g., v1.0) and release the final build.
- **Distribution**: Distribute the game through platforms like GitHub or via direct download.

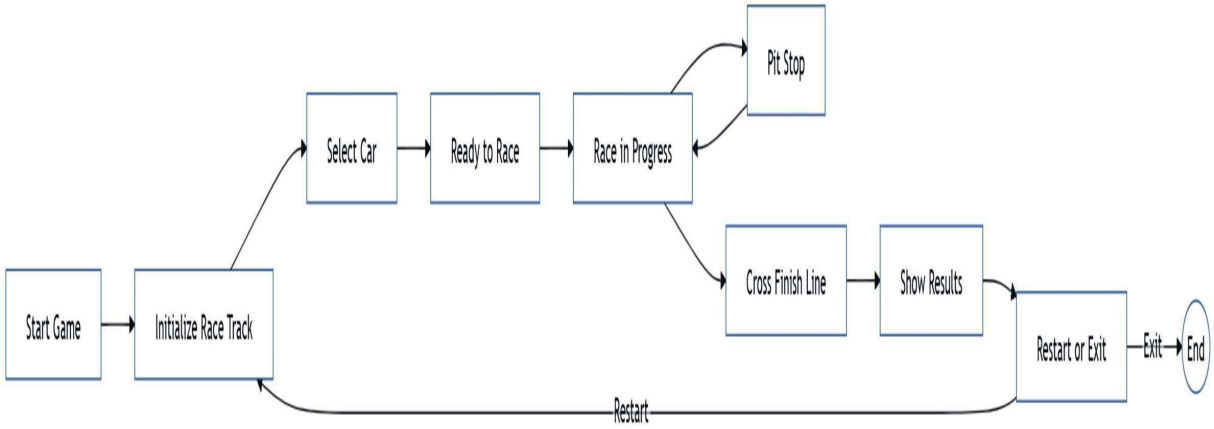
2. Post-Release (Optional)

Objective: Continuously improve the game after release.

- **Feedback Collection**: Collect feedback from players and identify potential improvements or bugs.
- **Updates**: Implement updates to fix bugs, add new features, or enhance gameplay.
- **Expansion**: Add new tracks, cars, or features to keep the game engaging.

✓ FlowChart:

– Flowchart is a diagrammatic representation of sequence of logical steps of a program. Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.

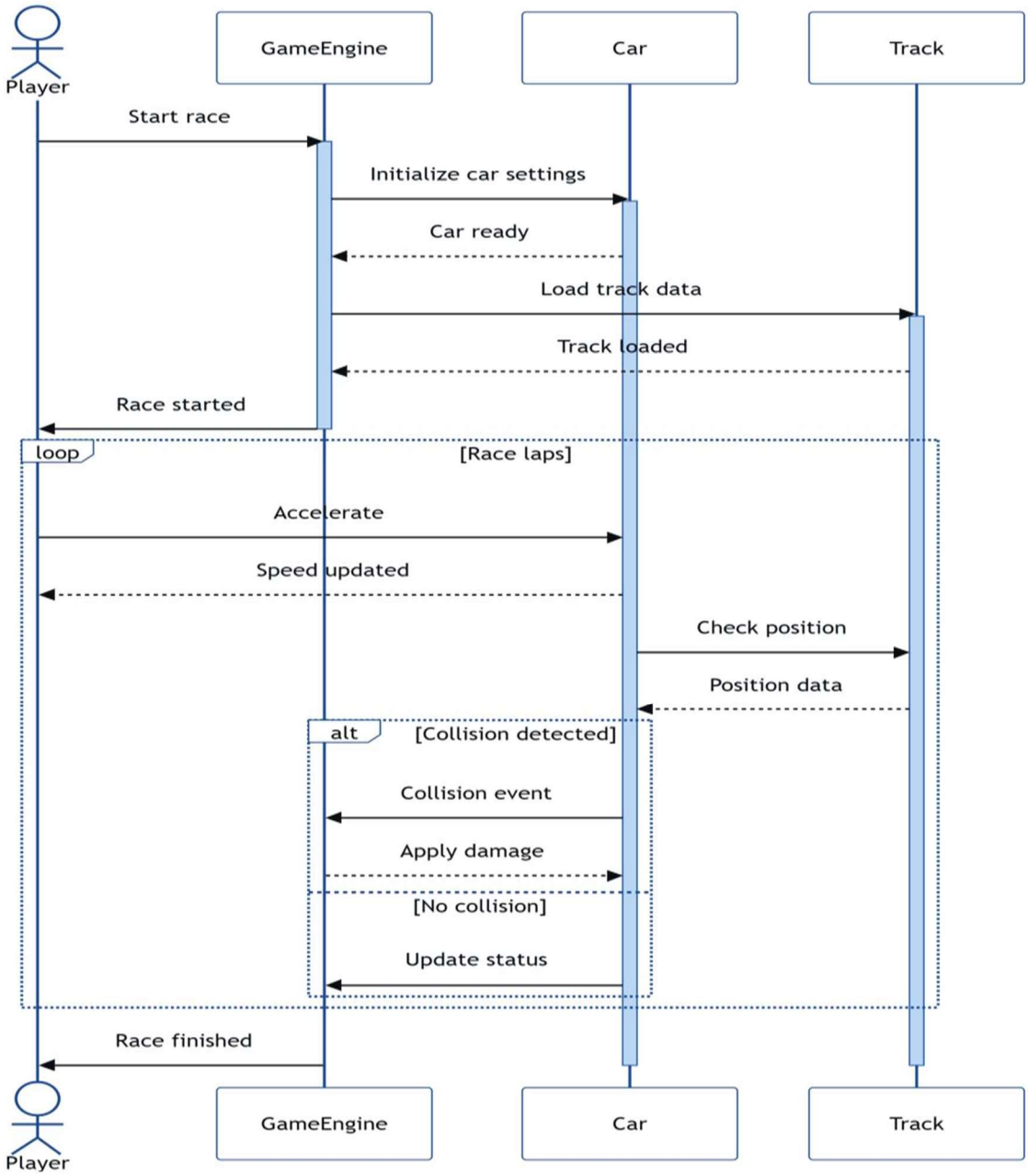


1. **Start**
    - Beginning of the development process.
2. **Project Planning & Requirement Analysis**
    - Define game concept, features, tools, and technologies.
    - Set project timeline and milestones.
3. **Game Design**
    - Design game mechanics, UI layout, and game flow.
    - Plan car behavior, track design, AI, and difficulty levels.
4. **Development Setup**
    - Create project structure in the IDE.
    - Initialize version control (Git).
    - Install necessary libraries (e.g., JavaFX, Swing).
5. **Core Game Development**
    - **Car Class and Mechanics**: Implement car movement, controls, speed, and handling.
    - **Track and Environment**: Design track layout, obstacles, and surface types.
    - **AI Opponents**: Implement AI behavior (pathfinding, difficulty levels).
    - **Graphics Rendering**: Implement real-time rendering of car movement and track.
    - **Game Loop and Event Handling**: Handle continuous game updates and user input.
6. **User Interface (UI) and Menus**
    - Implement main menu, in-game HUD, and pause menu.
    - Add results screen and score display.
7. **Sound and Music Integration**
    - Add sound effects (e.g., car sounds, collisions).
    - Integrate background music.
8. **Scoring & Leaderboard System**
    - Implement score calculation (time, position, laps).
    - Display scores and create leaderboards.
9. **Testing and Debugging**
    - Perform unit testing on car mechanics, AI, and collisions.
    - Playtest for game balance and performance.
    - Debug any issues or bugs.
10. **Documentation**
    - Write user manual and developer documentation.
    - Document testing and bug fixes.
11. **Final Release**
    - Package and distribute the game (create an executable JAR).
    - Version the final release (e.g., v1.0).
12. **Post-Release Updates (Optional)**
    - Collect user feedback.
    - Fix bugs and implement new features.
13. **End**
    - End of the development process.

✓ Sequence Diagram :

- ➢ Purpose of a Sequence Diagram To model high-level interaction among active objects within a system. To model interaction among objects inside a collaboration realizing a use case. It either models' generic interactions or some certain instances of interaction

.

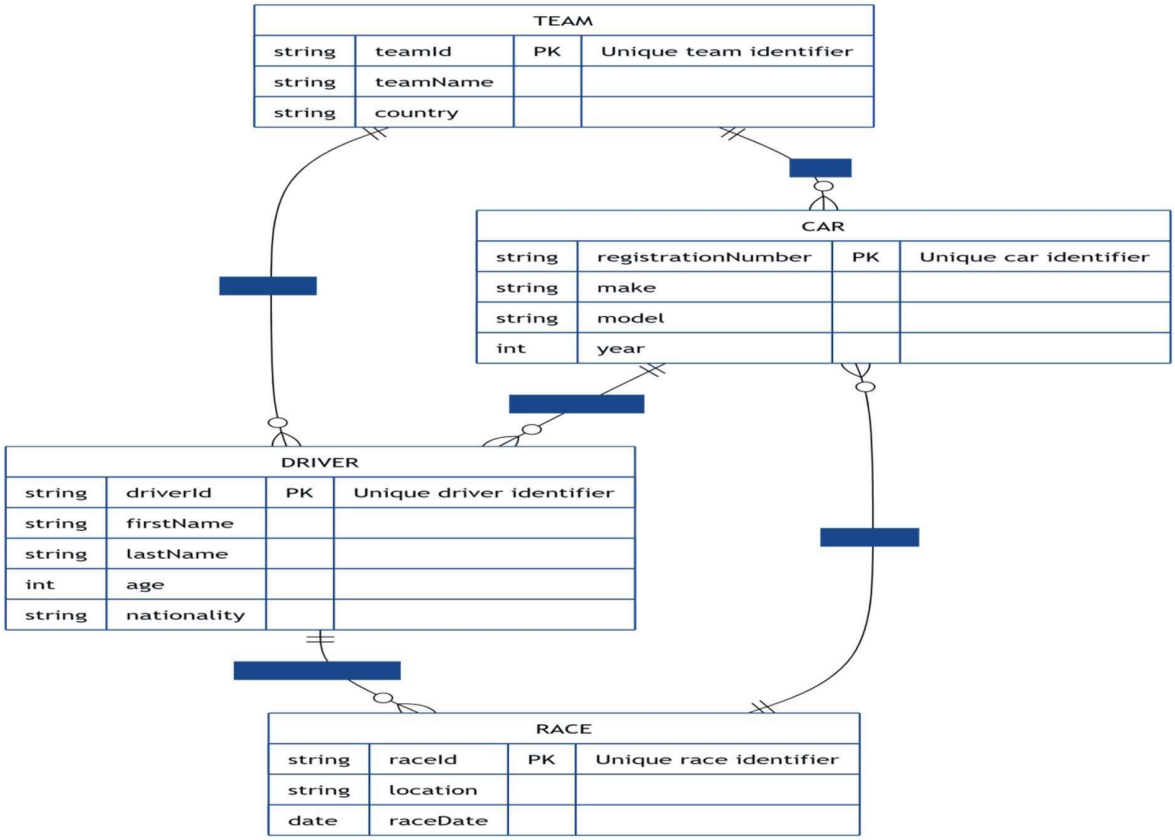


✓ Entity Relationship Diagram:

➢ ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.

➢ It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

➢ In ER modelling, the database structure is portrayed as a diagram called an entity-relationship diagram.



9. Project Outcome

The **Car Racing Game** project aims to develop a fully functional, engaging, and interactive game using Java. the outcome of this project can be broken down into several key deliverables and results, each contributing to the success of the project from both a technical and a user-experience perspective. Below is a detailed breakdown of the expected project outcomes.

1. Functional Car Racing Game

Objective: A fully operational car racing game where players can race cars against AI opponents or in multiplayer mode.

- **Key Features:**

  - **Playable Cars:** Players can control cars using the keyboard (e.g., arrow keys or WASD for movement).
  - **AI Opponents:** The game includes AI-controlled cars with varying levels of difficulty.
  - **Race Tracks:** Multiple race tracks with unique designs and features (e.g., obstacles, curves).
  - **Power-Ups and Obstacles:** Power-ups (e.g., speed boosts) and obstacles (e.g., oil slicks, barriers) are integrated into the track to add challenge and variety.
  - **Game Modes:** Single-player mode (against AI) and multiplayer mode (if implemented).

Outcome: Players experience smooth gameplay with responsive car controls, engaging AI, and challenging tracks that make the game interesting and replayable.

2. Interactive and Responsive User Interface (UI)

Objective: A well-designed and user-friendly interface that allows players to navigate the game smoothly.

- **Key Features:**

  - **Main Menu**: Players can start a new game, adjust settings, or quit the game.
  - **In-Game HUD**: Displays key information like current lap time, total race time, player's position, and speed.
  - **Pause Menu**: Allows players to pause the game, resume, or restart.
  - **Results Screen**: Displays race results, including lap times, rankings, and total race time.

Outcome: A clear, intuitive UI that enhances the player experience by providing essential game information and allowing easy navigation between different screen

3. Sound Effects and Background Music :

Objective: Integrate immersive sound effects and music to enhance the gaming experience.

- **Key Features**:
  - **Sound Effects**: Engine sounds, tire screeches, collisions, and other in-game actions.
  - **Background Music**: Music that plays during the race and menu screens to keep the player engaged.
  - **Sound Control**: Options to adjust or mute sounds and music from the game settings.

Outcome: A more immersive experience where the sound enhances the overall atmosphere of the game, making it more exciting and realistic.

4. Scoring and Leaderboard System :

Objective: Implement a scoring system that tracks player performance and ranks them based on race completion.

➢ **Key Features**:

- **Race Scoring**: Players are awarded points based on race completion time, position, and lap times.
- **Leaderboards**: Display top scores and rankings, allowing players to compare their performance against others.
- **Save and Display Scores**: Track high scores for replay value.

Outcome: Players are motivated to improve their performance through the inclusion of scoring and leaderboards, adding replayability to the game.

5. High-Quality Game Graphics :

Objective: Deliver visually appealing graphics for cars, tracks, and the environment.

➢ **Key Features**:
2D/3D Graphics: Depending on the project scope, the game may have 2D graphics for cars and tracks, or 3D graphics for a more realistic visual experience.
Animations: Smooth animations for car movements, collisions, and other dynamic game elements.
Track and Environment Design: Creative and unique track layouts with varied terrains, such as roads, grass, and dirt, to enhance gameplay.

Outcome: Visually appealing game with clear and smooth animations that keep players engaged. The environment and cars are well-designed and contribute to the overall game feel.

6. Performance and Stability :

Objective: Ensure the game runs efficiently on a variety of systems and is free from major bugs or performance issues.

- **Key Features**:
  - **Smooth Gameplay**: The game runs at a consistent frame rate (e.g., 30 FPS or 60 FPS) even with multiple cars and obstacles on the screen.
  - **Minimal Bugs and Crashes**: Through rigorous testing, bugs and crashes are minimized, ensuring a stable experience.
  - **Optimized Performance**: The game is optimized for various hardware configurations, making it accessible to a broad range of players.

Outcome: The game provides a smooth and enjoyable experience, free from performance issues or crashes, ensuring players can focus on gameplay without distractions.

7. Documentation :

Objective: Provide comprehensive documentation that explains how to play the game and describes its technical implementation.

- **Key Features**:
  - **User Manual**: Instructions for players on how to start the game, control the car, and understand the scoring system.
  - **Developer Documentation**: Details on the game's architecture, key classes, and design decisions.
  - **Testing Report**: Information on the testing process, issues encountered, and how they were resolved.

Outcome: Clear documentation ensures that users can easily play the game and developers can maintain or extend the game if needed.

8. Game Testing and Quality Assurance

Objective: Ensure that the game is bug-free, balanced, and provides a positive player experience.

- **Key Features**:
  - **Unit Testing**: Test individual components, such as car movement, collision detection, and AI behavior.
  - **Playtesting**: Conduct playtests to identify issues in game balance, difficulty, and overall fun factor.
  - **Bug Fixing**: Identify and resolve bugs, graphical glitches, or gameplay issues that may disrupt the user experience.

Outcome: A polished game with minimal bugs and well-balanced mechanics, ensuring players have a smooth and enjoyable experience.

9. Final Game Release

Objective: Package and distribute the game for others to play.

- **Key Features**:

  Executable File: Generate a final executable version of the game (e.g., JAR file) that can be run independently on any system with Java installed.
  Distribution: Make the game available for download via platforms like GitHub or other distribution channels.

  Outcome: The game is successfully released and is ready for players to download, play, and enjoy.

10. Post-Release Feedback and Updates

Objective: Gather feedback from players to improve the game.

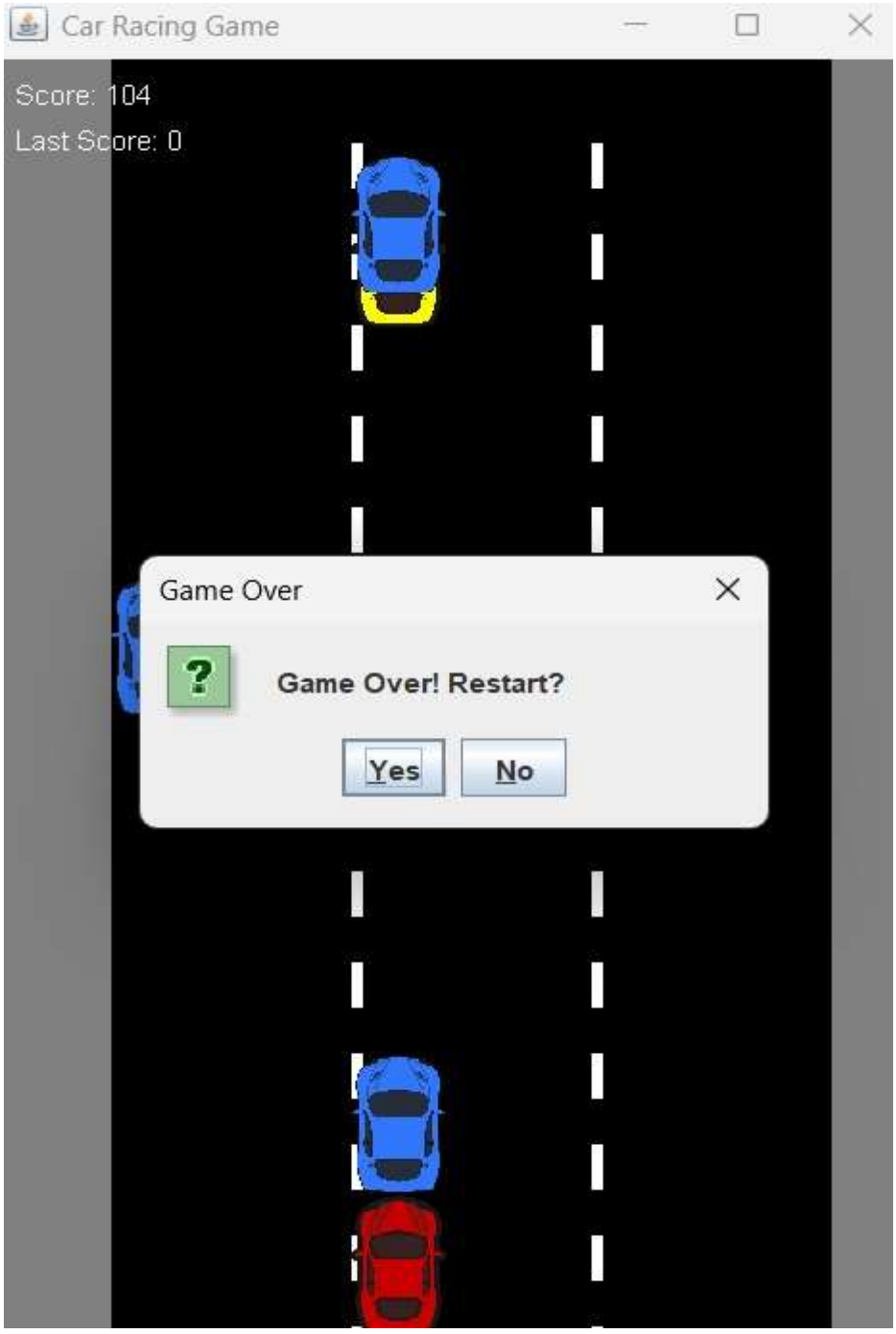
- **Key Features**:
  - **Feedback Collection**: Collect feedback from players about game difficulty, enjoyment, and bugs.
  - **Post-Release Updates**: Fix any remaining bugs, add new features, or adjust gameplay based on user feedback.

Outcome: Continuous improvement of the game through updates, ensuring that it remains engaging for players and fixing any lingering issues.

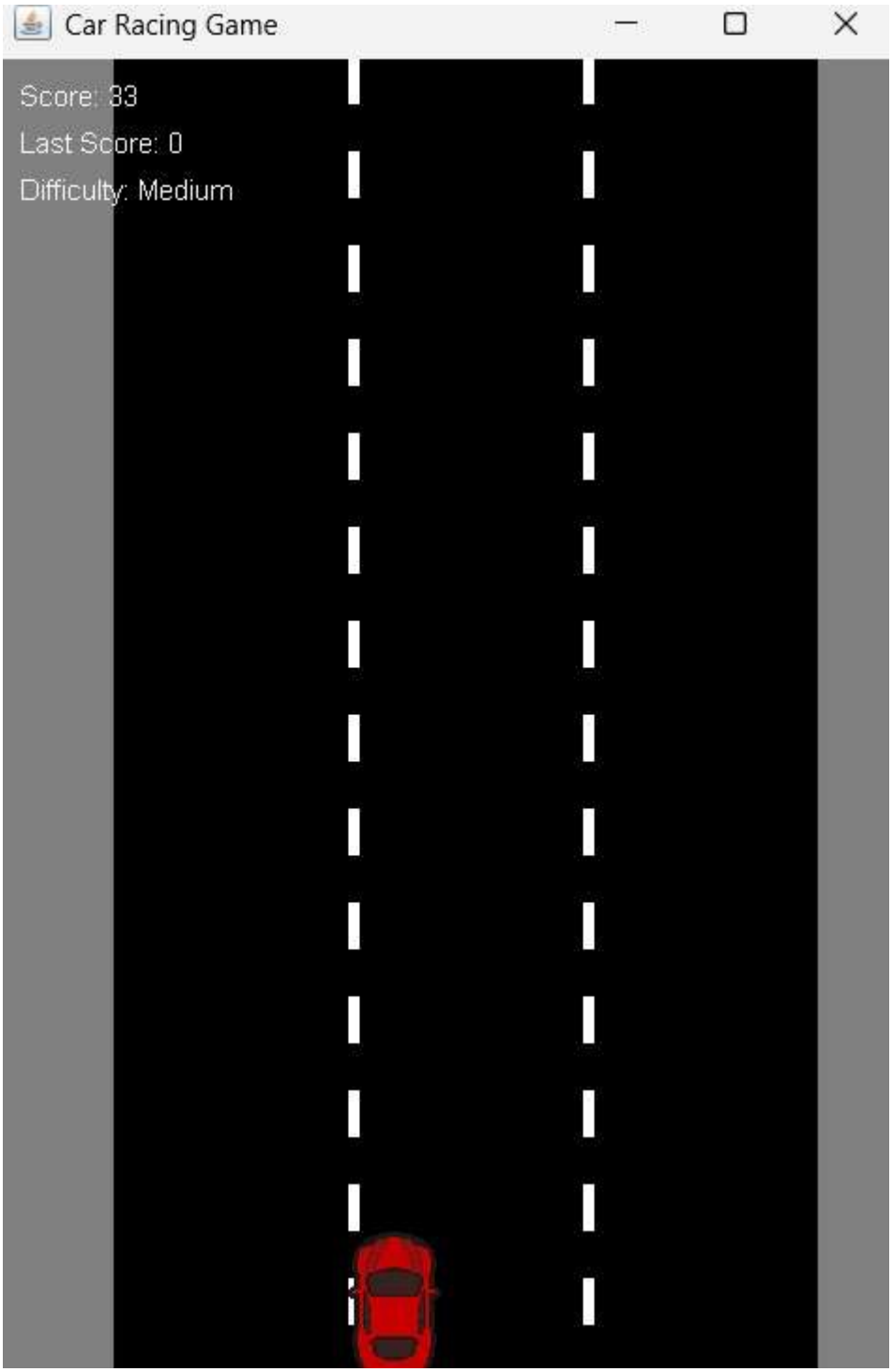
“GAME INTERFACE”

1ST MODULE :

2ND MODULE :


Car Racing Game
Score: 104
Last Score: 0
Game Over
Game Over! Restart?
Yes
No

3RD MODULE :



Car Racing Game Interface Design:

- **Start Game Module:**

1. First Page Layout:

  - A visually engaging background related to racing.
  - Prominent "Start Game" button for easy access.
  - Display of the current score prominently on the screen.

- **2. Second Car Running:**
  - The second car should be animated to run smoothly across the screen.
  - The score display should update in real-time to reflect the player's performance.

- **3. Game Over Screen:**
  - Clear indication of "Game Over" with a visually distinct message.
  - Two buttons for restarting the game:

✓ Yes: To restart the game.

✓ No: To exit or return to the main menu.

REFERENCES

maidpro.com
thecleaningauthority.com
handy.com
tidy.com
https://safaiwale.in
https://www.safsafaiwala.com/
http://www.dialhome.in/
http://cleaningboss.in/
https://book.heygoldie.com/Graduatesafaiwala