Software Requirements Specification (SRS)

Project Title: Traffic Sign Recognition Demo
Version: 1.0
Date: 25-08-2025
Status: Final Submission

Authors & Contributors

Role in Scrum Team	Name	SRN	Agile Contribution Area
Product Owner / Mentor	Dr. Swetha P	–	Acts as Product Owner guiding vision, clarifying requirements, and ensuring alignment with learning objectives.
Scrum Master	Adishree Gupta	PES1UG23CS024	Facilitates sprints, removes blockers, ensures Agile ceremonies (planning, review, retrospective), manages documentation.
Developer	Akshat	PES1UG23CS048	Focuses on model training, backend APIs, and sprint-wise integration of ML modules.
Developer	Monica M	PES1UG24CS813	Designs and iterates UI/UX, integrates recognition results into user stories, and provides visual feedback to the team.
Developer	Aditya Sharma	PES1UG23CS035	Handles dataset curation, preprocessing tasks, and sprint-based testing of ML pipelines.

Revision History

Version	Date	Author	Change Summary	Approval
1.0	25-08-2025	Dr. Swetha P	SRS with diagrams embedded	–

Approvals

Role	Name	Signature / Email	Date
Course Coordinator	Prof .Rajesh Banginwar		**25/08/2025**

1. Introduction

1.1 Purpose

This document specifies the requirements for the **Traffic Sign Recognition System (TSRS)**, developed as part of the Software Engineering course.

Our project demonstrates how **Software Engineering principles**—including requirements engineering, SDLC models, Agile methodology, testing, and security—can be applied to a **gamified, real-world challenge**.

According to the **World Health Organization**, over **1.3 million people die each year in road accidents**, many due to **ignored or misunderstood traffic signs**.

The **TSRS** combines **AI-based traffic sign recognition** with a **fun, interactive racing game**, illustrating how software engineering can contribute to:

- **Road safety** by reinforcing traffic sign awareness.
- **Driver training** through engaging gameplay that teaches sign recognition.
- **Smart city transport systems** by demonstrating AI-assisted traffic monitoring.

1.2 Scope

Users:

- Play the racing game and identify traffic signs encountered on the track.
- Interact via multiple-choice buttons to recognize signs within a limited time.

System:

- Automatically move the car on a continuous track with obstacles.
- Display random traffic signs and evaluate player responses.
- Use a **stubbed AI/ML model** to predict traffic signs and show confidence scores.
- Trigger **game consequences** for correct or incorrect recognition (e.g., speed boost, slowdown, "Caught by Police" pop-up).

Admins:

- Monitor gameplay logs and track player performance.
- Update or retrain the AI/ML model.

Practical Applications:

- **Driving schools:** Make learning traffic signs fun and interactive.
- **Gamified driver training:** Improve reflexes and recognition skills.
- **Smart city initiatives:** Demonstrate AI-assisted traffic awareness tools.

1.3 Audience

- **Faculty & Evaluators:** Assess application of Software Engineering and gamification principles.
- **Developers & Researchers:** Extend the project for AI-based traffic training or autonomous driving applications.
- **Students & Learners:** Experience an engaging, educational tool integrating **AI, gaming, and Software Engineering practices**.

1.4 Motivation & Vision

Just as **CrowdStrike** applied structured Software Engineering to cybersecurity, **our project applies Agile methodology principles to road safety education through gaming**.

Vision:

- Deliver a **working AI-powered racing game** that reinforces traffic sign recognition skills.
- Use **Agile iterative cycles** to improve AI predictions, gameplay mechanics, and UI design.
- Integrate **security and data validation** into gameplay and AI features.
- Inspire peers to see **Software Engineering as a practical, interactive framework** for solving real-world challenges.

2. Overall Description

The Traffic Sign Recognition System (TSRS) is a web-based application designed to demonstrate the integration of artificial intelligence and software engineering principles for road safety and smart transportation. The system enables users to upload images of traffic signs, which are then preprocessed, classified using a trained machine learning model, and displayed with corresponding confidence scores. By simulating a real-world use case of intelligent transportation, TSRS highlights how machine learning, secure design practices, and agile methodology can work together to build reliable software solutions.

From a product perspective, TSRS consists of three main components: a frontend user interface, a backend machine learning engine, and a database for storing logs and reports. The frontend provides a simple and accessible dashboard where users can upload images, view recognition results, and download reports. The backend handles preprocessing of the uploaded images, invokes the ML classification model, and communicates results back to the frontend. The database securely stores logs of recognition attempts, user details, and system performance reports, which can be accessed and monitored by administrators.

The major functions of the system include user authentication, secure image upload, preprocessing of images, classification of signs into more than 50 categories, and presentation of results with confidence scores. In addition, the system generates downloadable reports for users and provides administrators with access to performance monitoring tools and the ability to retrain the machine learning model using updated datasets. These features make the system valuable not only as a demonstration of traffic sign recognition but also as a foundation for real-world applications such as autonomous driving, smart traffic enforcement, and driver education.

The system is intended for two primary user roles: end-users (such as learners or researchers) and administrators. End-users interact with the system primarily to upload images and view classification results, while administrators are responsible for system monitoring, log review, and retraining the ML model to maintain accuracy. Both types of users interact with the system through secure sessions, with role-based access ensuring that sensitive functions are only accessible to authorized administrators.

TSRS is designed to operate in a web environment, with users accessing the system via modern browsers such as Google Chrome and Microsoft Edge. The backend services are implemented in Python using Flask or FastAPI, with the machine learning engine built on TensorFlow or PyTorch. A MySQL database supports the logging and reporting functions. To ensure robust and secure operation, the system is constrained by several requirements, including a maximum image upload size of 5MB, an accuracy target of at least 90%, and encrypted communication via HTTPS.

Overall, TSRS demonstrates the application of software engineering methodologies to an AI-driven project. It emphasizes clarity, systematic design, and testing while integrating security at every stage through the Secure Software Development Life Cycle (SecSDLC). The system not only provides a working proof-of-concept for traffic sign recognition but also showcases the adaptability and reliability expected from modern intelligent transport systems.

3. External Interface Requirements

3.1 User Interfaces

- **User Dashboard:**
    - Simple web-based interface for image upload (PNG/JPG).
    - Displays classification results with confidence scores.
    - Option to download a PDF report.
- **Admin Dashboard:**
    - Displays logs of recognition attempts.
    - Provides controls to retrain ML model with new datasets.
    - Usage statistics and system health monitoring.
- **Accessibility Features:**
    - Responsive design for desktop and mobile browsers.

3.2 Hardware Interfaces

- **Client Device:**
    - User requires a desktop/laptop/mobile device with a web browser.
- **Server Hardware:**
    - Runs ML model (TensorFlow/PyTorch) and backend services (Flask/FastAPI).

- **Database Server:**
    - MySQL database to store logs, reports, and user data.

3.3 Software Interfaces

- **Frontend → Backend:**
    - REST API endpoints for uploading images and retrieving classification results.
- **Backend → ML Engine:**
    - Python interface (TensorFlow/PyTorch model API).
- **Backend → Database:**
    - SQL queries for logs, authentication, and report generation.
- **File Handling:**
    - Validation libraries to check image type and size.

3.4 Communication Interfaces

- **Protocol:** HTTPS enforced for all data exchanges.
- **Data Format:** JSON for API communication.
- **Upload Constraints:** Max image size 5MB per upload.

4. System Features

4.1 User Interface (UI)

- **Clean and intuitive layout for desktop and mobile devices.**
- **Automatic car movement with endless scrolling track.**
- **Smooth animations for car, traffic signs, and obstacles.**
- **Pop-up cards and animations for correct/incorrect recognition.**
- **Dark/Light mode toggle.**
- **Responsive design for different screen sizes.**

4.2. Traffic Sign Recognition

- **Preloaded set of traffic sign images appearing randomly on track.**
- **Stubbed ML model for predictions (can be replaced by a real model later).**
- **Top-1 or top-3 predictions displayed.**
- **Multiple-choice or button-based user input for sign recognition.**
- **Timer for recognition: player must respond within a limited time.**

4.3. Gameplay Mechanics

- **Automatic car movement with increasing speed as distance progresses.**
- **Random obstacles and other cars on the track.**
- **Player must identify traffic signs to continue safely.**
- **Correct identification → car continues at normal speed or gets speed boost.**

- **Incorrect or missed sign → pop-up card showing consequences (e.g., “Caught by Police”, collision, fines, or slow down).**
- **Streaks and bonuses for consecutive correct identifications.**

4.4. Scoring and Progress

- **Points awarded for correct recognition.**
- **Bonus points for speed and streaks.**
- **Penalties for wrong or missed signs (deduct points or slow car).**
- **Distance traveled tracked as part of scoring.**
- **Leaderboard for high scores and longest distances.**

4.5. Result Visualization

- **Pop-up cards for consequences on wrong recognition.**
- **Visual feedback for correct recognition: green checkmark, confetti, speed boost.**
- **Confidence bar/indicator showing AI prediction confidence (optional).**
- **Real-time score and streak display.**

4.6. Admin / Demo Features (Optional)

- **View logs of signs used in demo.**
- **Track player performance statistics (correct vs incorrect guesses).**
- **Option to replace or retrain ML model in future.**

4.7. Security Features

- **HTTPS enforced for web app communication.**
- **Validation of uploaded files if using real user uploads.**
- **Sanitize inputs to prevent malicious activity.**
- **Session timeout after inactivity (optional).**

4.8. Educational / Fun Features

- **Sign info panel: meaning, rules, and fines for each traffic sign.**
- **Road safety tips triggered by recognized signs.**
- **Mini-quiz mode before or after race for learning reinforcement.**
- **Fun consequences for wrong recognition: humorous pop-ups like “Caught by Police!”**
- **Animations and sound effects for immersive gameplay.**

5. Requirements Engineering

5.1 Elicitation & Analysis

Sources: brainstorming, labs, case studies, personas.

5.2 Functional Requirements

Functional Requirements (FRs)

Req ID	Requirement Definition	Rationale
FR1	The system shall allow the player's car to move automatically on the track.	Core gameplay mechanic ensures continuous game flow.
FR2	The system shall display traffic signs at random intervals on the track.	Adds challenge and tests recognition skills.
FR3	The system shall allow the player to identify traffic signs using multiple-choice buttons.	Enables interaction and gameplay.
FR4	The system shall use a stubbed ML model to predict traffic signs.	Demonstrates AI integration.
FR5	The system shall provide a confidence score for each prediction.	Shows AI reliability and future ML model replacement.
FR6	The system shall trigger consequences for incorrect recognition.	Adds challenge and fun penalties.
FR7	The system shall provide visual feedback for correct recognition.	Motivates players and enhances UX.
FR8	The system shall track player score during gameplay.	Core to scoring and leaderboard.
FR9	The system shall track player streaks of correct recognitions.	Encourages continuous engagement.
FR10	The system shall track the distance travelled by the car.	Part of performance evaluation.
FR11	The system shall display pop-up cards for penalties.	Makes incorrect actions immersive.
FR12	The system shall allow the player to pause and resume the game.	Provides control over gameplay.
FR13	The system shall allow restarting the game at any time.	Improves usability and replayability.
FR14	The system shall display obstacles and other cars on the track.	Adds challenge and makes gameplay dynamic.
FR15	The system shall provide a leaderboard showing top scores.	Encourages competition and replayability.
FR16	The system shall allow players to view sign information and rules.	Educational purpose.
FR17	The system shall play sound effects for actions and consequences.	Enhances immersion.
FR18	The system shall animate speed boosts for consecutive correct guesses.	Provides visual reward and excitement.

FR19	The system shall animate slowdowns or collisions for incorrect guesses.	Improves clarity of consequences.
FR20	The system shall maintain session-based recognition history.	Tracks performance over a session.
FR21	The system shall gradually increase difficulty over time.	Keeps players challenged.
FR22	The system shall display top-3 AI predictions for each sign.	Shows AI uncertainty and engages player decisions.
FR23	The system shall allow sound toggling (on/off).	Improves accessibility.
FR24	The system shall allow visual toggling (dark/light mode).	Enhances user experience.
FR25	The system shall allow optional preloaded traffic sign images for demo mode.	Simplifies gameplay for demo purposes.

5.3 Non-Functional Requirements

Non-Functional Requirements (NFRs)

Req ID	Requirement Definition	Rationale
NFR1	The system shall be responsive on desktop and mobile devices.	Ensures wide usability.
NFR2	The system shall enforce HTTPS for all communications.	Ensures secure data transmission.
NFR3	The system shall validate user inputs to prevent malicious activity.	Enhances security.
NFR4	The system shall limit image uploads to 5 MB.	Maintains performance.
NFR5	The system shall maintain a frame rate of at least 30 FPS.	Ensures smooth gameplay.
NFR6	The system shall respond to AI predictions within 2 seconds.	Keeps gameplay fast.
NFR7	The system shall preload images and assets for smooth rendering.	Reduces lag and improves UX.
NFR8	The system shall allow multiple concurrent sessions (future multiplayer).	Ensures scalability.
NFR9	The system shall store leaderboard data persistently.	Maintains continuity.
NFR10	The system shall provide visually appealing animations.	Enhances engagement.
NFR11	The system shall be intuitive and easy to use.	Improves accessibility.
NFR12	The system shall maintain session data until the user exits.	Tracks performance correctly.
NFR13	The system shall scale to larger sets of traffic sign images.	Prepares for future expansion.
NFR14	The system shall load new traffic signs within 1 second.	Ensures smooth gameplay.

NFR15	The system shall display pop-ups within 0.5 seconds of incorrect action.	Provides immediate feedback.
NFR16	The system shall maintain minimal latency for button interactions.	Improves responsiveness.
NFR17	The system shall provide clear visual feedback for all actions.	Enhances UX.
NFR18	The system shall have backup and recovery for leaderboard data.	Prevents data loss.
NFR19	The system shall allow toggling of sound effects for accessibility.	Improves inclusivity.
NFR20	The system shall support dark and light themes.	Enhances usability.
NFR21	The system shall have minimal CPU and memory footprint.	Ensures compatibility with low-end devices.
NFR22	The system shall allow easy integration with a real ML model in the future.	Ensures maintainability.
NFR23	The system shall handle at least 1000 concurrent users (future scalability).	Supports expansion.
NFR24	The system shall provide quick restart of gameplay after game over (<1 sec).	Maintains engagement.
NFR25	The system shall be compatible with modern web browsers (Chrome, Firefox, Edge, Safari).	Ensures accessibility.

5.6 Personas & Scenarios

Persona 1: Learner Driver

- **Age:** 18
- **Occupation:** Student, learning to drive
- **Tech Comfort:** Moderate; comfortable using mobile apps and games
- **Goals:**
  - Learn traffic signs effectively
  - Improve reflexes and recognition speed
  - Enjoy interactive learning instead of reading manuals
- **Pain Points:**
  - Traditional driving manuals are boring and hard to remember
  - Nervous about recognizing signs during real driving

Persona 2: Driving School Instructor

- **Age:** 35
- **Occupation:** Driving school teacher
- **Tech Comfort:** Moderate; uses web apps for teaching resources
- **Goals:**
  - Provide interactive learning tools to students
  - Track student progress in traffic sign recognition
  - Reduce accidents due to ignorance of road signs
- **Pain Points:**
  - Hard to make students practice traffic signs frequently
  - Limited tools to evaluate recognition skills in a fun way

Persona 3: Student Researcher

- **Age:** 21
- **Occupation:** Computer Science student
- **Tech Comfort:** High; familiar with AI/ML and web technologies
- **Goals:**
  - Study AI applications in real-world scenarios
  - Understand integration of gaming, ML, and software engineering
  - Extend the system to autonomous driving or analytics
- **Pain Points:**
  - Difficulty in finding projects that combine AI, gaming, and SE principles
  - Limited access to interactive educational datasets

Scenarios

Scenario 1: Learner Driving Game Session

She opens the game on his laptop, selects a difficulty level, and starts the race. Traffic signs appear randomly on the track. He identifies each sign using multiple-choice buttons. Correct answers give speed boosts, while wrong answers trigger a “Caught by Police” pop-up and slowdown. At the end of the session, he checks his score, streak, and leaderboard ranking.

Scenario 2: Driving School Instructor Monitoring

He logs into the admin panel to review her students’ gameplay. She checks each student’s recognition accuracy, streaks, and common mistakes. She uses this data to provide feedback and plan targeted practice sessions.

6. Security

- The app will use HTTPS to ensure all data exchanged between the player’s device and server remains secure.

- User inputs and selections will be validated to prevent injection attacks and maintain a safe, stable gaming environment.

- Uploaded images will be checked for format and size, rejecting any files that might harm the system.

- Session management will include automatic timeouts to protect users who leave the game inactive for extended periods.

- Sensitive game data, including leaderboard scores, will be securely stored and protected against unauthorized access.

- Pop-up cards and notifications will be sanitized to avoid code injection and malicious content display.

- Game assets and scripts will be loaded from trusted sources only to prevent tampering or malware attacks.

- Player actions and game events will be logged securely to detect suspicious behavior or unauthorized modifications.

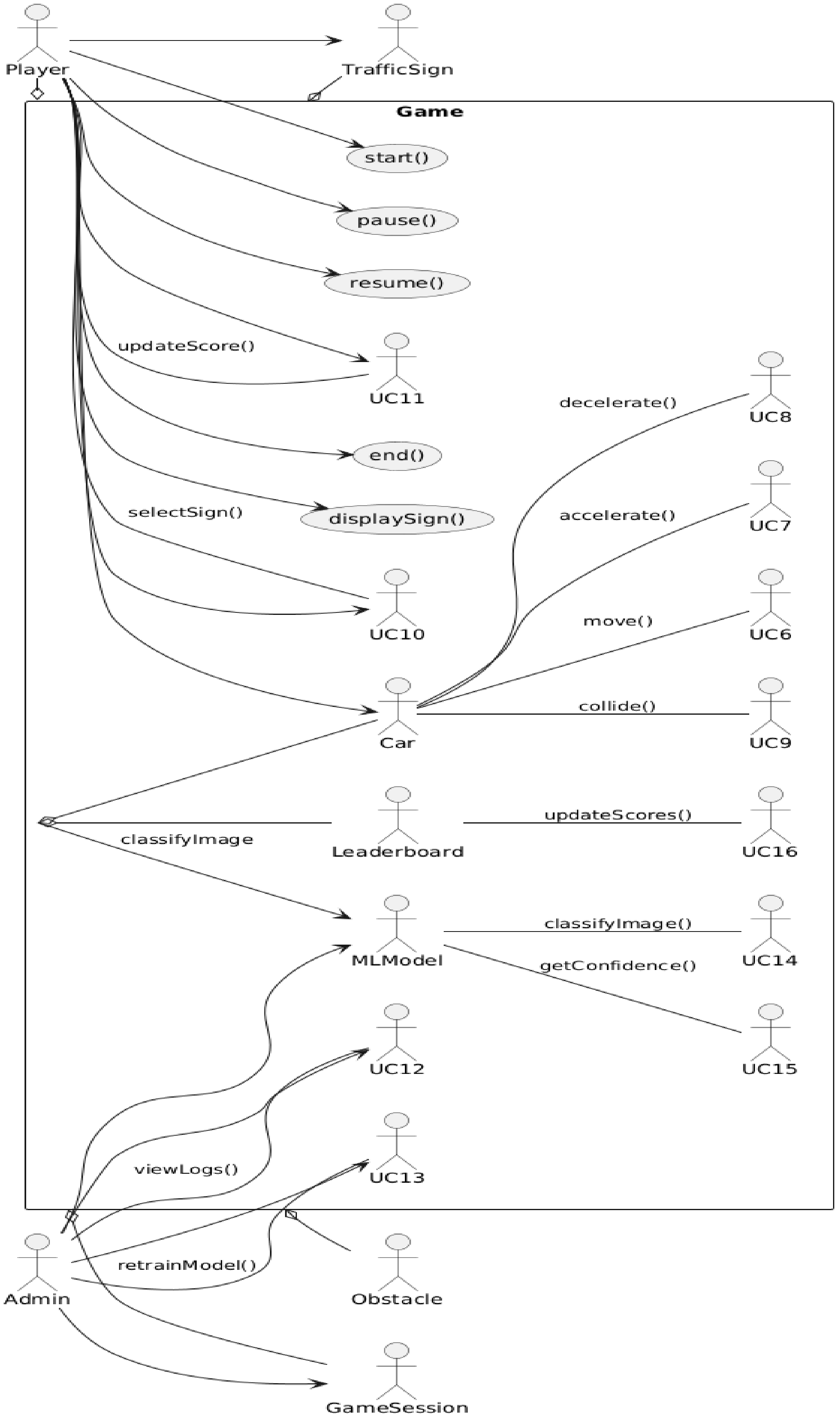
- All communication between AI prediction system and game engine will be encrypted to prevent data leaks.

- Regular updates will be applied to dependencies and libraries to protect against known security vulnerabilities.

7. Quality & Testing

1. **Performance & Speed** – The game should run smoothly without lag; tested by playing continuously and checking FPS.
2. **Reliability** – The game should not crash during normal play; tested by playing for a long time and doing repeated actions.
3. **Risk Management** – Potential problems like wrong AI predictions or invalid inputs are handled safely; tested by giving wrong inputs on purpose.
4. **Security & Privacy** – Data and player information is safe; tested using simple penetration tests and checking file upload validations.
5. **Code Coverage** – All parts of the game logic should be tested; verified by checking if every code path runs at least once.
6. **Ethics & AI Usage** – AI predictions are fair and safe; tested by checking that feedback is correct and not biased.
7. **Open Source & Licenses** – Libraries used are free and legal; checked by verifying licenses and giving proper credit.
8. **System Testing** – Game features, AI, and controls are tested together; verified by playing the game and checking all functions work.
9. **Security Testing** – Check for vulnerabilities using simple tests like entering wrong inputs or random files; ensures safe gameplay.
10. **Maintainability** – Game code is easy to update or improve; tested by changing AI or adding new signs without breaking the game.

8. UML Diagram



9. Requirements Traceability Matrix (RTM)

Req ID	Requirement Definition	Source / Reference	Acceptance Test / Verification	Quality / Project Aspect	Priority
FR1	Car moves automatically on the track	Game Mechanics	Play game and check smooth movement	Usability, Performance	High
FR2	Traffic signs appear randomly	Game Mechanics	Observe multiple sessions to verify randomness	Usability, Game Design	High
FR3	Player identifies signs via buttons	UI Requirement	Click buttons; correct label recorded	Usability, Design	High
FR4	Stubbed ML model predicts signs	AI Module	Upload demo image; prediction shown	Architecture, AI Integration	High
FR5	Display confidence score for AI predictions	AI Module	Verify confidence value appears	Usability, Technical Feedback	Medium
FR6	Wrong recognition triggers consequences	Game Mechanics	Test incorrect guesses result in pop-ups/slowdown	Error Handling, Usability	High
FR7	Visual feedback for correct recognition	UI/UX	Correct guesses trigger animations	Usability, Design	High
FR8	Track score, streaks, distance	Game Mechanics	Score updates correctly during gameplay	Metrics, Monitoring	High
FR9	Obstacles and other cars appear	Game Mechanics	Check collisions and obstacle appearance	Design, Risk Management	High
FR10	Leaderboard displays top scores	Project Design	Enter multiple scores; verify leaderboard	Project Management, Agile PM	Medium
FR11	Pause, resume, and restart game	UI Requirement	Buttons function correctly; state preserved	Usability, Design Flow	Medium

FR12	Educational panel explains signs	Educational Feature	Panel shows correct info	Software Quality, Useability Engineering	Low
FR13	Play sound effects for actions	UI/UX	Sound triggers correctly	Design, Feedback Mechanism	Medium
FR14	Animate speed boost for correct streaks	Game Mechanics	Speed boost animation appears	Modularity, Design Pattern	Medium
FR15	Animate slowdown for wrong guesses	Game Mechanics	Slowdown animation occurs	Error Handling, Feedback	High
FR16	Session-based recognition history	Game Mechanics	Session logs all guesses	Forward/Backward Compatibility	Medium
FR17	Gradual increase in difficulty	Game Mechanics	Verify signs appear faster	Technical Debt, Risk Management	Medium
FR18	Display top-3 AI predictions	AI Module	Panel shows multiple predictions	Software Architecture, Agile Architecture	Low
FR19	Toggle sound on/off	UI Requirement	Verify audio toggles correctly	Useability Engineering	Low
FR20	Toggle dark/light mode	UI Requirement	Theme changes correctly	Design Flow, Modularity	Low
FR21	Optional preloaded sign images for demo	Demo Feature	Demo mode works with preloaded images	Project Monitoring, Standup Review	Low
NFR1	HTTPS and input validation	Security Architecture	Test file upload and secure communication	Security & Privacy	High
NFR2	Frame rate ≥ 30 FPS	Tools Setup, Agile PM	Measure FPS with performance tool	Performance Metric, Test Measurement	High
NFR3	AI prediction < 2 seconds	Architecture & Design	Timer measurement for AI response	Technical Debt, Test Driven Development	Medium
NFR4	Version control with Git	Project Management	Check commits, branches, merges	Version Control, Teamwork	High
NFR5	Project standup meetings	PM Role	Document meeting minutes	Teamwork, Risk Management	Medium

NFR6	System testing (functional, integration, regression)	Testing Practice	Execute test scripts	SW Quality, Test Driven Development	High
NFR7	Security testing (Penetration, Fuzzing)	Security Architecture	Conduct test scenarios	Security & Privacy	High
NFR8	Modular design with API	Software Architecture	Test module replacement	Design Modularity, Forward/Backward Compatibility	Medium
NFR9	Error handling for wrong inputs	Design Techniques	Test invalid inputs	Error Handling & Management	High
NFR10	Documentation of design, code, and architecture	Project Management	Verify design docs exist	Project Lifecycle, Agile PM	Medium