**Introduction:**

The "SkyTrace" project is an interactive geospatial path simulation tool designed to track the movement of an object (like a plane) on a digital map. The primary aim is to simulate and visualize paths in real time, enhancing the experience with danger zone detection and warnings.

This project not only demonstrates the capabilities of Tkinter and TkinterMapView in creating dynamic and interactive applications but also integrates various modules like Shapely for geometric calculations, pygame for audio alerts, and PIL for image processing.

**Key Features:**

Real-time Movement:

The application allows users to define a path on the map, along which a simulated object (represented as a plane) moves. This movement is depicted dynamically, showing progress in real time, which makes the application more interactive and engaging.

Danger Zone Detection:

The system is equipped to detect predefined danger zones, which are areas on the map where the simulated object should be cautious. When the plane enters or approaches a danger zone, the system provides real-time alerts through both visual indicators and audio notifications, ensuring that the user is aware of potential hazards.

User Interaction:

The application provides multiple methods for user interaction, including:

Map Clicks: Users can add points to the path directly by right-clicking on the map, making it easy to define and modify the path.

Clipboard Data: Users can load latitude and longitude coordinates directly from the clipboard, facilitating quick and accurate path setup.

Control Buttons: The interface includes buttons to start, pause, stop, clear, and undo paths, giving users full control over the simulation process.

**Purpose:**

* **Simulation of Geospatial Movements:**
  + SkyTrace is designed to offer users a dynamic tool to simulate the movement of objects (such as airplanes, vehicles, or other entities) along a predefined path on a digital map. This simulation provides an interactive visual representation of how the object moves across various terrains, highlighting changes in position in real-time.
* **Risk Management and Safety Awareness:**
  + A critical aspect of SkyTrace is its ability to detect proximity to predefined danger zones. These zones represent areas of risk, such as restricted airspaces, hazardous terrains, or regions with adverse conditions. The application alerts users when their simulated path intersects or comes close to these zones, emphasizing the importance of route safety.
* **Educational and Practical Utility:**
  + The tool serves both educational and practical purposes. In educational settings, it can be used to teach concepts related to geospatial data, navigation, and risk management. Practically, it can aid in planning safe routes for various scenarios, such as flight path planning, maritime navigation, or emergency evacuation routes.

**Significance:**

* **Educational Applications:**
  + SkyTrace can be utilized in academic environments to demonstrate the principles of geospatial data, pathfinding algorithms, and the importance of safety in navigation. It offers a hands-on experience for students and professionals, enhancing their understanding of how geospatial technologies can be applied in real-world scenarios.
* **Planning and Decision-Making:**
  + For professionals involved in route planning, logistics, or emergency management, SkyTrace provides a valuable tool for simulating and analyzing different paths. By highlighting potential risks associated with certain routes, the application can help users make informed decisions to avoid danger zones, ultimately ensuring safer journeys.
* **Demonstrating Geospatial Data Applications:**
  + The project showcases the integration of various technologies and libraries to manipulate and visualize geospatial data. It serves as a practical example of how Python can be used to develop applications that require complex spatial calculations, real-time updates, and user interaction.

**Scope:**

* **Dynamic Simulation with Speed Variations:**
  + SkyTrace allows users to simulate the movement of objects at varying speeds, providing flexibility to model different scenarios. The speed can be adjusted based on the user’s requirements, making the simulation more realistic and adaptable to various contexts, such as slow-moving vehicles or high-speed aircraft.
* **Proximity Detection and Alerts:**
  + The application is equipped with functionality to detect when the simulated object is near or within danger zones. Upon detection, it provides immediate alerts through audio and visual cues, ensuring the user is promptly informed of potential risks. This feature is crucial for scenarios where safety is a top priority.
* **Interactive Map Interface:**
  + Users interact with a digital map where they can easily add, modify, or remove points on the path through intuitive right-click actions. The interface is designed to be user-friendly, allowing real-time updates and seamless integration of additional features, such as loading coordinates from the clipboard or undoing actions.
* **Customization and Flexibility:**
  + The tool is designed to be flexible, allowing users to load different sets of danger zones, customize path points, and adjust simulation parameters like speed. This adaptability makes SkyTrace suitable for various use cases, from academic demonstrations to practical route planning.

**Requirements**

### ****Hardware Requirements:****

* **Standard PC or Laptop:**
  + A regular desktop computer or laptop is sufficient to run the SkyTrace application.
  + **Processor:** A dual-core processor or better is recommended for smooth performance, particularly during real-time simulation.
  + **RAM:** At least 4 GB of RAM is advisable to ensure that the application runs without lags, especially when handling multiple modules simultaneously.
  + **Storage:** Minimal storage requirements, primarily for installing Python and necessary libraries, and storing simulation data. An SSD is preferable for faster data access.
  + **Display:** A screen resolution of 1280x720 or higher is recommended for clear visualization of the map and user interface elements.
  + **Mouse and Keyboard:** Essential for interacting with the GUI, adding path points, and controlling the simulation. A mouse is particularly important for precise map interaction.

### ****Software Requirements:****

* **Python 3.x:**
  + The core programming language used to develop the SkyTrace application. Python's flexibility and extensive libraries make it ideal for handling the various functionalities required in the project.
  + Python 3.7 or later is recommended to ensure compatibility with all libraries used.
* **Tkinter:**
  + The standard Python library for creating graphical user interfaces (GUIs). Tkinter is used for building the entire interface of the SkyTrace application, including buttons, input fields, and other interactive components.
* **TkinterMapView:**
  + A specific module within Tkinter that allows the integration of maps into the GUI. It supports dynamic map rendering, enabling users to interactively add path points and visualize the movement of the simulated object (plane).
  + **Map Interaction:** Essential for displaying maps, adding waypoints, and rendering the simulated path in real-time.
* **Shapely:**
  + A Python library used for geometric operations. In the SkyTrace application, Shapely is utilized to define and manipulate the shapes of danger zones on the map, enabling accurate proximity detection.
  + **Geospatial Calculations:** Shapely allows for the calculation of distances and the determination of whether the plane is inside or near a danger zone.
* **Pygame:**
  + A library for creating games and multimedia applications in Python. In SkyTrace, Pygame is used for audio playback, providing sound alerts when the plane enters or is near a danger zone.
  + **Audio Alerts:** Pygame handles the playback of custom sound files to alert users of potential risks in real-time.
* **Pandas:**
  + A powerful data manipulation library for Python. Pandas is employed in SkyTrace to read and process Excel files containing geospatial data for danger zones.
  + **Data Handling:** Pandas ensures efficient reading, filtering, and manipulation of Excel data, making it possible to dynamically load and update danger zones during the simulation.
* **PIL (Pillow):**
  + A Python Imaging Library used for opening, manipulating, and saving various image file formats. Pillow is incorporated into SkyTrace to handle any image processing requirements, such as displaying custom icons or map overlays.
  + **Image Processing:** PIL ensures that any image-related tasks, like resizing or transforming icons and overlays, are handled efficiently.

### ****Communication Requirements:****

* **None Required (Offline Tool):**
  + **Offline Operation:** SkyTrace is designed as a standalone application that does not require any network connectivity to function. All functionalities, including map rendering, path simulation, and danger zone detection, are performed locally on the user's machine.
  + **Data Independence:** The application uses locally stored files (e.g., Excel for danger zones) and does not rely on external servers or internet access, making it highly portable and secure for various use cases, including educational environments or field simulations where internet access might be limited.

### Non-Functional Requirements (Detailed)

**Performance:**

* The application must ensure smooth and real-time updates of the map, especially during path simulation and danger zone detection.
* Responsiveness is critical as users interact with the map by adding points or adjusting the path.
* The audio alerts must be played without delay when entering or approaching a danger zone to provide timely feedback.

**Usability:**

* The user interface is designed to be user-friendly with clear labels, buttons, and instructions.
* Functions like starting, pausing, stopping, and clearing the path are accessible via intuitive buttons, making it easy for users of all experience levels to operate.
* Map interactions, such as adding points by clicking, are straightforward, and error messages guide users in case of incorrect input formats.

**Reliability:**

* The application must consistently detect danger zones as the simulated object moves along the path.
* The system is expected to handle varying speeds and different path complexities without missing any danger zones.
* Audio alerts and pop-up messages should be reliably triggered based on the object's position relative to the danger zones.

**Security:**

* The application runs locally on the user's device, ensuring that no sensitive data is transmitted over the internet.
* Since it is an offline tool, it provides a secure environment where user data and interactions are kept private.
* The use of local resources (like clipboard data for path coordinates) ensures that no external threats can interfere with the operation.

Gantt Chart

#### ****1. Initial Planning (Week 1)****

* **Task 1.1: Define Project Scope and Objectives (2 Days)**
  + Establish the primary goals and features of the SkyTrace project, including real-time path simulation and danger zone detection.
* **Task 1.2: Research and Requirements Gathering (3 Days)**
  + Gather information on the necessary libraries, tools, and modules required for the project.
  + Define both functional and non-functional requirements.
* **Task 1.3: Design Initial System Architecture (2 Days)**
  + Develop a preliminary system architecture, outlining the interactions between modules.

#### ****2. Development Phase (Weeks 2-4)****

* **Task 2.1: Develop Input Module (4 Days)**
  + Implement functionality for receiving user inputs, including map clicks and clipboard data.
  + Ensure proper validation and storage of path points.
* **Task 2.2: Implement Danger Zone Detection Module (6 Days)**
  + Integrate geospatial data and develop the logic for detecting proximity to danger zones.
  + Implement real-time audio and visual alerts for detected danger zones.
* **Task 2.3: Build Simulation Module (5 Days)**
  + Create the logic for moving the simulated object along the path.
  + Implement real-time updates to the map based on object movement.
* **Task 2.4: Develop User Interface Module (5 Days)**
  + Design and implement the user interface, including buttons for controlling the simulation and displaying alerts.
  + Integrate the interface with the underlying modules.

#### ****3. Integration & Testing (Week 5)****

* **Task 3.1: Module Integration (3 Days)**
  + Integrate all modules (Input, Danger Zone Detection, Simulation, UI) into a cohesive application.
* **Task 3.2: Unit Testing (3 Days)**
  + Test individual modules to ensure they function correctly in isolation.
* **Task 3.3: System Testing (4 Days)**
  + Perform comprehensive testing of the entire system to identify and fix any issues.
  + Validate that the application meets all functional and non-functional requirements.

#### ****4. Final Presentation Preparation (Week 6)****

* **Task 4.1: Documentation (3 Days)**
  + Prepare detailed documentation covering the system architecture, module descriptions, and user manual.
* **Task 4.2: Slide Deck Preparation (2 Days)**
  + Create a presentation that includes an overview of the project, system architecture, use case diagram, class diagram, Gantt chart, and testing results.
* **Task 4.3: Rehearsal and Refinement (2 Days)**
  + Rehearse the final presentation and refine the content based on feedback.

#### ****5. Final Presentation (End of Week 6)****

* **Task 5.1: Present the SkyTrace Project (1 Day)**
  + Deliver the final presentation to stakeholders, demonstrating the application's features and functionality.

SRS

# SRS

### 1. ****Introduction****

#### 1.1 Purpose

The purpose section should provide a clear understanding of why the SkyTrace project exists and what it aims to achieve. This includes defining the stakeholders, such as developers, end-users, and any other parties involved.

**Expanded Example:** The SkyTrace project is designed to develop an advanced geospatial simulation tool that facilitates interactive path simulation on a map interface, with an emphasis on real-time danger zone detection. The primary purpose of this project is to enhance geospatial analysis capabilities by allowing users to visualize and monitor paths for potential hazards during simulations. This SRS document serves as a formal agreement between the development team and stakeholders, including project sponsors, educators, and end-users. It outlines the detailed software requirements, ensuring that the final product meets the intended functionality, usability, and performance criteria.

#### 1.2 Scope

The scope section details what the project will cover, outlining the main objectives and boundaries. It is important to clarify what is within and outside the scope of the project.

**Expanded Example:** SkyTrace is a standalone desktop application designed to simulate and analyze geospatial paths in real-time. The primary objectives include:

* **Interactive Path Simulation:** Users can dynamically create and simulate paths on a map interface.
* **Real-time Danger Zone Detection:** The application will alert users when a simulated path is near or within predefined danger zones.
* **User Customization:** Users can set simulation speeds, modify paths, and customize alert settings.

**Out of Scope:**

* SkyTrace will not support live data feeds or real-time GPS tracking.
* The application will not include multi-user collaboration features.
* Cloud-based data storage or processing will not be implemented; all operations will be local to the user's machine.

#### 1.3 Definitions, Acronyms, and Abbreviations

This section defines any specific terms that will be used throughout the document, helping to avoid ambiguity.

**Expanded Example:**

* **GIS:** Geographic Information System - a system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data.
* **Waypoint:** A specific geographic location defined by latitude and longitude coordinates used to create a path.
* **Danger Zone:** A predefined geographic area where hazards exist, which could pose a risk to the simulated path.

#### 1.4 References

This section includes all documents and resources that provide background information or guidelines relevant to the SkyTrace project.

**Expanded Example:**

* IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.
* Python 3.8 Documentation, available at [https://docs.python.org/3.8/](https://docs.python.org/3.8/" \t "_new)
* TkinterMapView Documentation, available at [https://tkintermapview.readthedocs.io/](https://tkintermapview.readthedocs.io/" \t "_new)
* Shapely Documentation, available at [https://shapely.readthedocs.io/en/stable/](https://shapely.readthedocs.io/en/stable/" \t "_new)
* Pygame Documentation, available at https://www.pygame.org/docs/

#### 1.5 Overview

The overview briefly summarizes the organization of the document and what readers can expect to find in each section.

**Expanded Example:** This document is structured to provide a comprehensive understanding of the SkyTrace project, beginning with an introduction that outlines the project's purpose, scope, and key terms. It then describes the overall system, including its context, functionality, user characteristics, and constraints. Specific requirements are detailed, covering functional and non-functional aspects of the system. Finally, appendices provide supplementary information, such as diagrams and algorithm descriptions.

### 2. ****Overall Description****

#### 2.1 Product Perspective

The product perspective section places SkyTrace in a larger context, describing how it fits into existing systems and highlighting any dependencies.

**Expanded Example:** SkyTrace is designed as an independent application but fits within the broader context of educational and research tools used for geospatial analysis. It does not rely on external servers or cloud services, ensuring all processing is performed locally on the user’s machine. The application leverages existing Python libraries such as Tkinter for GUI management, TkinterMapView for map visualization, Shapely for geometric calculations, and Pygame for audio notifications. While SkyTrace can operate independently, it could be extended or integrated with other geospatial tools in the future, such as GIS platforms for more advanced data analysis.

#### 2.2 Product Functions

This section lists and explains the primary functions of SkyTrace. Each function should be described in terms of its inputs, processing, outputs, and any specific constraints or requirements.

**Expanded Example:**

**Function 1: Dynamic Path Creation**

* **Inputs:** User-defined waypoints (latitude and longitude) entered by right-clicking on the map interface.
* **Processing:** The system calculates the path between waypoints and continuously updates the path as new points are added.
* **Outputs:** A dynamically drawn path displayed on the map, updated in real-time as the simulation progresses.
* **Constraints:** The system must handle up to 100 waypoints without significant performance degradation.

**Function 2: Real-time Danger Zone Detection**

* **Inputs:** Predefined danger zones stored in the system's database or loaded from external files.
* **Processing:** The system checks the proximity of the simulated path to the danger zones using geometric calculations (e.g., distance, intersection).
* **Outputs:** Alerts displayed on the screen and played through audio when the path approaches or enters a danger zone. The system specifies whether the danger zone is on the right or left of the path.
* **Constraints:** The system must detect and alert for danger zones within 100 milliseconds of approaching.

#### 2.3 User Characteristics

This section provides a detailed profile of the intended users, their needs, and how they are expected to interact with the system.

**Expanded Example:** SkyTrace is designed for a diverse group of users, including:

* **Educators:** Use the application to demonstrate concepts related to geospatial analysis, pathfinding, and hazard detection. They need an easy-to-use interface with clear instructions for classroom use.
* **Students:** Engage with the application to learn about geospatial analysis, using the simulation features to explore how different paths interact with danger zones.
* **Researchers:** Use the application for experimental simulations, analyzing path data and danger zone interactions. Researchers may require more advanced customization options for in-depth analysis.

#### 2.4 Constraints

List any design, technical, or regulatory constraints that the project must adhere to. This could include platform limitations, performance benchmarks, or legal requirements.

**Expanded Example:**

* **Platform Limitations:** SkyTrace must be compatible with Windows, macOS, and Linux operating systems, and it must run on Python 3.8 or higher.
* **Performance Benchmarks:** The system must process user inputs and update the map in real-time, with a maximum delay of 100 milliseconds for danger zone detection.
* **Legal and Regulatory Constraints:** The application must comply with local data privacy regulations, ensuring that no user data is transmitted or stored externally.

#### 2.5 Assumptions and Dependencies

Identify any assumptions made during the project planning and any dependencies that could impact the development or operation of the system.

**Expanded Example:**

**Assumptions:**

* + Users will have basic proficiency in using desktop applications and understanding geographical concepts.
  + The user's machine has adequate processing power and memory to handle real-time simulations.
  + The necessary Python libraries (Tkinter, TkinterMapView, Shapely, Pygame) are available and compatible with the user's system.

**Dependencies:**

* + The project depends on third-party libraries such as Tkinter, TkinterMapView, Shapely, and Pygame. Any changes or updates to these libraries could affect the system’s functionality.
  + The availability of map data and danger zone definitions, which could be provided by external files or a pre-loaded database.

### 3. ****Specific Requirements****

#### 3.1 External Interface Requirements

This section details how SkyTrace will interact with external entities, including users, hardware, and other software systems.

**Expanded Example:**

**User Interface (UI):**

* The UI will be designed using Tkinter, featuring a main window with buttons for simulation control (Start, Stop, Clear Path, Undo Path, Load Lat/Lon).
* A map area will be integrated using TkinterMapView, where users can right-click to add waypoints and visualize the path.
* Pop-up windows or dialog boxes will be used to display alerts, such as when a danger zone is detected.

**Hardware Interface:**

* SkyTrace is expected to run on standard desktop hardware with no special requirements.
* The application should function properly with typical input devices, such as a mouse and keyboard.

**Software Interface:**

* SkyTrace will integrate with the TkinterMapView module to provide map visualization capabilities.
* Shapely will be used for geometric operations, such as detecting when a path is near or intersects with a danger zone.
* Pygame will handle the playback of audio alerts.

**Communication Interface:**

* The application will not require any external communication or internet connectivity, ensuring that all data is processed locally.

#### 3.2 System Features

This section provides detailed descriptions of each feature, breaking down the functional requirements into specific, actionable items.

**Expanded Example:**

**Feature 1: Dynamic Path Drawing**

* **Description:** The system allows users to create a path by clicking on the map interface. As the simulation progresses, the path is dynamically drawn in green, updating in real-time.
* **Functional Requirements:**
  + The map interface must register right-clicks as waypoints.
  + The system should automatically draw lines between waypoints, forming a continuous path.
  + The path color should change to green during simulation to indicate active tracking.

**Feature 2: Danger Zone Detection**

* **Description:** The application monitors the path for proximity to predefined danger zones, alerting the user when a zone is approached.
* **Functional Requirements:**
  + The system must load and store danger zones, either from a local database or external files.
  + Geometric calculations should determine if a path is within a defined distance from a danger zone.
  + Visual (on-screen alerts) and auditory (sound alerts using Pygame) warnings should trigger when the path is near a danger zone.

**Feature 3: Simulation Control**

* **Description:** Users can control the simulation with options to start, stop, clear, and undo the path.
* **Functional Requirements:**
  + Buttons for 'Start Simulation,' 'Stop Simulation,' 'Clear Path,' and 'Undo Path' must be implemented.
  + The system must handle the starting and stopping of the simulation without errors.
  + Clearing the path should remove all visual traces of the current path, resetting the simulation.

### 4. ****Non-Functional Requirements****

#### 4.1 Performance Requirements

Detail the expected performance standards for SkyTrace, such as speed, responsiveness, and resource utilization.

**Expanded Example:**

* The application must update the map display in real-time, with no more than a 100-millisecond delay between user input and map updates.
* Danger zone detection calculations should be performed within 50 milliseconds to ensure timely alerts.
* The application must handle up to 100 waypoints and 50 danger zones without significant performance degradation.

#### 4.2 Usability Requirements

Describe how user-friendly and accessible the system must be, considering the target audience.

**Expanded Example:**

* The user interface should be intuitive and easy to navigate, with clear labels and tooltips for all controls.
* A help menu or user guide should be available within the application, providing instructions on how to use each feature.
* The application should support keyboard shortcuts for common actions, such as starting/stopping the simulation and clearing the path.

#### 4.3 Reliability Requirements

Specify the system's reliability, including uptime, error rates, and recovery procedures.

**Expanded Example:**

* The application must maintain a 99% uptime during normal operation, with minimal crashes or unresponsive behavior.
* Error messages should be clear and provide guidance on how to resolve the issue.
* The system must automatically save the current path and danger zones, allowing users to recover their session after an unexpected shutdown.

#### 4.4 Security Requirements

Detail any security considerations, including data privacy and protection against unauthorized access.

**Expanded Example:**

* SkyTrace will not transmit or store user data externally, ensuring all data remains on the local machine.
* The application must include basic data protection mechanisms, such as encryption for any saved path or danger zone data.
* User access to saved sessions should be restricted by local operating system permissions, ensuring that only authorized users can modify or delete data.

### 5. ****Appendices****

#### 5.1 Glossary

Provide a glossary of terms used in the SRS, ensuring that readers unfamiliar with specific terminology can understand the document.

**Expanded Example:**

* **Waypoint:** A point of reference in a path, defined by its geographical coordinates (latitude and longitude).
* **Danger Zone:** A specific area on the map that poses a potential hazard, which the system monitors during simulations.

#### 5.2 Diagrams

Include any relevant diagrams, such as use case diagrams, sequence diagrams, and system architecture diagrams.

**Expanded Example:**

* **Use Case Diagram:** Illustrates the interactions between users and the system, including creating paths, starting simulations, and detecting danger zones.
* **System Architecture Diagram:** Provides an overview of the application's structure, showing how the different modules (UI, danger zone detection, path drawing) interact with each other.