S.D.M.E Society's

SDM COLLEGE OF ENGINEERING AND TECHNOLOGY DHAVALAGIRI, DHARWAD-580002

(AFFILIATED TO VISHVESVARAYA TECHNOLOGICAL UNIVERSITY)

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

SRS On

WILDLIFE WATCH

Under the guidance of

Prof. VARSHA S J

SUBMITTED BY,

PRADYUMNA P	**2SD21IS033**
VISHAL K SAKLATHI	**2SD21IS061**
PAVAN KALBURGI	**2SD22IS402**

HOD-ISE	**Project Co-Ordinator**	**Project Guide**
Dr Jagadeesh Pujari	**Prof. Varsha S J**	**Prof. Varsha S J**

6th Semester B.E **Academic Year 2023-24**

Software Requirements Specification Document (SRS)

1. Introduction:

“WildLife Watch” is a groundbreaking web platform and mobile app that empowers communities to actively engage in wildlife conservation while promoting responsible interaction with natural habitats. Through real-time wildlife sighting reporting, educational resources, and threat alerts, users can contribute to monitoring wildlife populations and protecting endangered species. The purpose of this document is to provide a comprehensive outline of the software requirements for the development of "WildLife Watch," a web platform and mobile application dedicated to wildlife conservation and responsible interaction with natural habitats. This Software Requirements Specification (SRS) defines the functional and non-functional requirements essential for the successful implementation of the Wildlife Watch system.

1.1 Purpose

The purpose of the Software Requirements Specifications (SRS) for the "WildLife Watch" web app platform is to provide a comprehensive blueprint for the development, elucidating the essential functionalities and requirements that the platform must fulfil. This SRS serves as a guiding document, outlining the project's scope, objectives, and constraints related to Wildlife management within the web app. The intended audience comprises wildlife watchers, rangers, officials. This document outlines the functional and non-functional requirements of the Wildlife Watch platform, including its core features, system constraints, user interfaces, performance specifications, and development milestones. It serves as a guide for stakeholders, developers, and testers involved in the design, development, and implementation of the Wildlife Watch system.

1.2 Scope

The scope of "WildLife Watch" encompasses the development of a web platform and mobile app aimed at engaging communities in wildlife conservation and responsible interaction with natural habitats. The platform will enable users to report real-time wildlife sightings with multimedia content, access educational resources on conservation topics, and receive threat alerts related to habitat degradation or species endangerment. Additionally, the platform will foster community engagement through social features and collaborative projects, facilitating data collection for scientific research and conservation efforts.

The primary objectives include providing a user-friendly interface for wildlife reporting and education, promoting awareness of local conservation issues, and supporting community-driven conservation initiatives. The system will prioritize scalability and usability to accommodate a diverse user base, from casual wildlife enthusiasts to conservation professionals.

1.3 Definitions, Acronyms, and Abbreviations

1. WildLife Watch: Refers to the a web platform and mobile application dedicated to wildlife conservation and responsible interaction with natural habitats.
2. SRS: Software Requirements Specifications, the comprehensive document outlining the functional and non-functional requirements of the WildLife Watch web app.

3. API: Application Programming Interface, a set of rules that allows different software applications to communicate with each other, facilitating data exchange for the WildLife Watch web app.

4. UI: User Interface, the visual elements and design components of the WildLife Watch web app that users interact with.

5. UX: User Experience, encompassing the overall experience and satisfaction of users interacting with the WildLife Watch web app.

6. DBMS: Database Management System, the software used to manage and organize the database that stores information for the WildLife Watch web app.

7. CMS: Content Management System, if applicable, for handling and updating content within the WildLife Watch web app.

8. OCR: Optical Character Recognition, if relevant, for extracting text information from images or documents within the WildLife Watch web app.

9. SSL: Secure Sockets Layer, a security protocol ensuring encrypted communication between users and the WildLife Watch web app.

1.4 References

[1] Soledad Luna, Margaret Gold, Alexandra Albert, Luigi Ceccaroni, “Developing Mobile Applications for Environmental and Biodiversity Citizen Science: Considerations and Recommendations”, (June 2018) Multimedia Tools and Applications for Environmental & Biodiversity Informatics.

[2] Wright, M. D., Turner, W. C., & others, "A review of wildlife monitoring technologies”, (2016) Journal of Wildlife Management

[3] Eweoya, I. O., Ajayi, O. J., & others, "Design and Implementation of Web-based GIS for Wildlife Management System", (2017) Journal of Geographic Information System

[4] Shanahan, D. J., Fuller, R. A., & others, “The role of citizen science in wildlife monitoring and conservation”, (2015) Trends in Ecology & Evolution, 30(8), 462-470.

[5] Bhatia, N., & Saini, S.,"Real-time web applications: Technologies and challenges", (2012) International Journal of Computer Applications

1.5 Overview

Overview of the WildLife Watch web app:

"WildLife Watch" is a web platform and mobile application designed to empower communities in wildlife conservation and responsible habitat interaction. This Software Requirements Specification (SRS) outlines the essential functionalities and characteristics of the system, which will enable users to report real-time wildlife sightings, access educational resources, receive threat alerts, and engage in community-driven conservation efforts. The platform aims to promote biodiversity conservation by leveraging user participation, data collection, and collaboration with conservation organizations.

The primary objectives of this project are:

1.Real-time Wildlife Sighting Reporting:

- Users should be able to report sightings of wildlife species they encounter in their local habitats.
- The platform will support the submission of multimedia data (photos, videos) to enhance sighting reports.

2.Threat Alerts and Monitoring:

- The platform will integrate a system for alerting users about potential threats to wildlife, such as habitat destruction, poaching, or invasive species.
- Users can receive timely notifications and updates on conservation efforts and initiatives.

3.Educational Resources:

- The application will provide educational content related to wildlife conservation, including species profiles, conservation tips, and habitat preservation guidelines.
- Users can access curated information to increase awareness and understanding of local wildlife.

4.Community Engagement:

- Wildlife Watch aims to facilitate community participation in conservation activities through social features, discussions, and collaborative projects.
- Users can connect with like-minded individuals, conservation experts, and local organizations to coordinate conservation efforts.

5.Data Collection and Analysis:

- The platform will aggregate wildlife sighting data to contribute to broader scientific research and conservation initiatives.
- Data analytics tools will support the assessment of wildlife population trends and distribution patterns over time.

2.The Overall Description

Overall Description of the WildLife Watch web app:

"WildLife Watch" is a web platform and mobile application designed to empower communities in wildlife conservation and responsible habitat interaction. This section provides a comprehensive overview of the factors influencing the development of the web app, offering insights into its purpose, functionality, and potential impact.

2.1 Product Perspective

2.1.1 System Interfaces

- The system will interface with the *MongoDB/Firebase* database to store and retrieve wildlife sighting reports, educational content, and threat alerts.

- The web interface (built with *ReactJS*) and mobile app (built with *Flutter*) will provide user-friendly interactions for reporting wildlife sightings and accessing educational resources.

2.1.2 Interfaces

- The platform will have interfaces for user registration, wildlife sighting reporting, educational content browsing, and receiving threat alerts.

- Admin interfaces will allow moderators(watchers) to manage reported sightings and content.

2.1.3 Hardware Interfaces

- The mobile app will require standard smartphone hardware (camera, microphone) for wildlife sighting reporting and user engagement.

- The Website requires a browser for visiting the page and monitoring the information.

2.1.4 Software Interfaces

- The platform will integrate with third-party services for email notifications, authentication (e.g., OAuth), and possibly content delivery networks (CDNs) for hosting multimedia educational content.

2.1.5 Communications Interfaces

- The platform will communicate over HTTP(S) protocols for client-server interactions between the app/web frontend and backend services.

- Email notifications will be sent via SMTP or a third-party email service.

2.1.6 Memory Constraints

- The mobile app will need to manage local storage efficiently for caching data and images related to wildlife sightings.

- The web platform should optimize image and video loading to accommodate varying network speeds.

2.1.7 Operations

- Regular backups and data synchronization processes will be implemented to ensure data integrity and availability.

- The platform should handle concurrent user interactions smoothly, especially during peak usage times.

2.1.8 Site Adaptation Requirements

- The platform should be adaptable to different geographical locations and ecosystems, allowing for customization based on local wildlife species and conservation needs.

2.2 Product Functions

User:

- Allow users to report wildlife sightings with time, location descriptions (if available), and media uploads like photos, name of animal, condition.

- Provide educational resources such as articles, videos, and quizzes about local wildlife and conservation practices.

- Send threat alerts and conservation updates to users based on reported sightings and ongoing conservation efforts.

Watcher:

-Allow watcher to login to the system with different ID (than users) and operate further.

-Watcher should be able to add, update, delete information of animals and also review the sightings of users.

2.3 User Characteristics

- Users may vary in technical proficiency, from casual wildlife enthusiasts to conservation experts.

- Users should be motivated by a shared interest in wildlife conservation and responsible interaction with natural habitats.

-Watcher and Officials are also part of user’s scope

2.4 Constraints

- Limited or no access to GPS technology may impact the accuracy of wildlife sighting reports.

- Availability of internet connectivity in remote areas may affect the real-time nature of data reporting and access.

2.5 Assumptions and Dependencies

- Assumption: Users will have access to basic smartphone devices capable of running the Flutter app.

- Initially focus on a specific forest or area for deployment to validate the platform's effectiveness and gather user feedback.

- Dependency: Successful integration with MongoDB for data storage and retrieval.

2.6 Apportioning of Requirements

- Initially focus on a specific forest or area for deployment to validate the platform's effectiveness and gather user feedback.

- Plan for scalability and expansion to other regions based on initial deployment success.

Overall, "WildLife Watch" aims to leverage modern web and mobile technologies to promote community engagement in wildlife conservation efforts, emphasizing real-time reporting and education as key components of the platform. The success of the platform will depend on user adoption, reliable data management, and continuous improvement based on user feedback and conservation outcomes.

3.Specific Requirements

3.1 External Interfaces

User Interface (Web & Mobile App):

-The system shall provide a user-friendly interface for both web and mobile platforms to accommodate users accessing the platform via different devices.

-The user interface shall allow users to log in, report wildlife sightings, access educational resources, and receive alerts.

Database Interface:

-The system shall integrate with a database system to store user information, wildlife sightings, and watcher data.

-The database interface should support CRUD (Create, Read, Update, Delete) operations for managing wildlife data.

3.2 Functions

User Login:

-Users shall be able to log in securely using their credentials (ID and password).

-Invalid login attempts should be handled to prevent unauthorized access.

Watcher Operations:

-Watchers can add, delete, and update wildlife information in the database based on real-time observations.
-The system shall provide CRUD functionalities specifically tailored for watcher roles.

Wildlife Sighting Reporting:

-Users can report wildlife sightings by providing details such as animal name, photo, and location (place).

-Reported sightings should be stored in the database for monitoring and analysis purposes.

3.3 Performance Requirements

Real-time Data Processing:

-The system should handle real-time data processing for wildlife sightings and updates from watchers.

-Response times for user interactions (login, reporting sightings) should be optimized for a seamless user experience.

Scalability:

-The system architecture should be scalable to accommodate increasing numbers of users and wildlife data without compromising performance.

3.4 Logical Database Requirements

Data Entities

The logical database design should define the entities (or tables) that will store relevant data for the system:

- 1.User Profile Entity:
  - Attributes: `UserID` (Primary Key), `Name`, `Contact`, `Address`, `Password`
  - Description: Stores information about registered users of the platform.
- 2.Watcher Entity:
  - Attributes: `WatcherID` (Primary Key), `Name`, `Age`, `Gender`, `Contact`, `Address`, `Role`, `Password`, `PlaceID` (Foreign Key)
  - Description: Contains details of forest watchers who monitor wildlife and input data into the system.
- 3.Wildlife Sighting Entity:
  - Attributes: `SightingID` (Primary Key), `AnimalName`, `Photo`, `Place`, `DateTime`
  - Description: Stores reported wildlife sightings along with associated details.

Relationships

Define the relationships between entities to maintain data integrity and enforce business rules:

1.User-Watcher Relationship:

- Many-to-One Relationship:

- A watcher can be connected with Users.

- Foreign Key: `PlaceID` in the Watcher entity references `PlaceID` in the Place entity (if applicable).

2.User -Wildlife Sighting Relationship:

- One-to-Many Relationship:

- Each user can report multiple wildlife sightings.

- Foreign Key: `UserID` in the Wildlife Sighting entity links sightings to the corresponding watcher.

Data Integrity Constraints

Implement constraints to ensure data integrity and enforce business rules:

- Primary Keys:
  - Each entity should have a primary key (e.g., `UserID`, `WatcherID`, `SightingID`) to uniquely identify records.
- Foreign Keys:

  - Use foreign keys to establish relationships between entities and enforce referential integrity.
  - For example, `PlaceID` in the Watcher entity references a `PlaceID` in a separate `Place` entity that contains location information.

Indexing and Performance Optimization

Implement indexing strategies for efficient data retrieval and performance optimization:

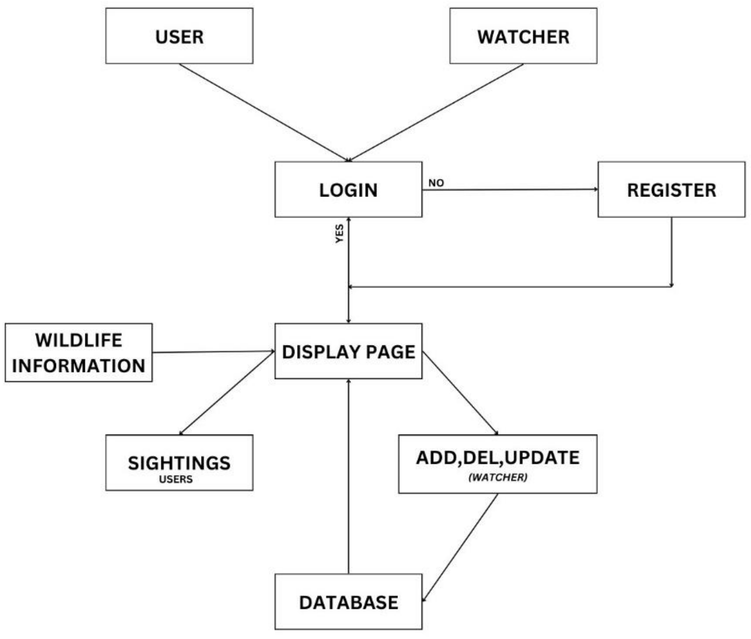
- Indexing:
  - Create indexes on frequently queried attributes (e.g., `UserID`, `WatcherID`) to speed up data retrieval operations.
- Query Optimization:
  - Design queries that leverage indexes and avoid full table scans to optimize database performance.

Normalization

Apply normalization techniques to reduce redundancy and improve data consistency:

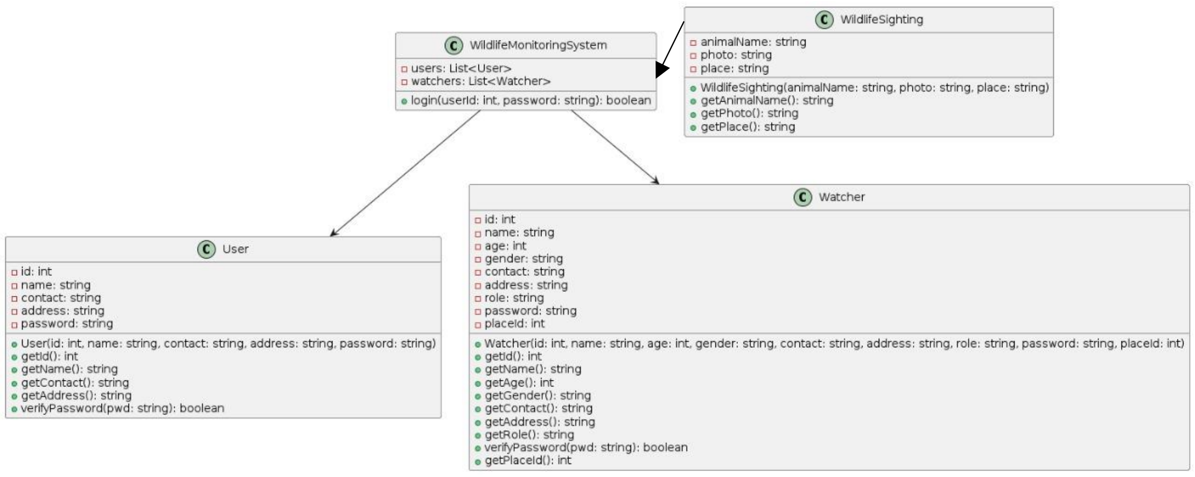
- Normalization:
  - Break down data into smaller, manageable entities to minimize data redundancy and avoid update anomalies.
  - Ensure that each piece of data is stored in only one place to maintain consistency and facilitate data updates.

Data Flow Diagram



This Data Flow Diagram demonstrates how data flows through different modules of application.

Class Diagram



3.5 Design Constraints

3.5.1 Standards Compliance

1. **Data Protection Regulations**

   The database design must comply with data protection laws or any other relevant local regulations regarding the protection of user data and wildlife information.

   Database Security Standards

   (1) Implementation of encryption methods for sensitive data at rest and during transmission.

   (2) Access controls and user authentication mechanisms to safeguard user and watcher information.

3.**Storage Capacity**

   (1) The database design should be scalable to accommodate increasing wildlife data without compromising performance.
   (2) Constraints on the maximum size of the database due to hardware limitations should be considered.

3.6 Software System Attributes

3.6.1 Reliability

Data Integrity:

- The system should maintain data integrity through error handling, validation checks, and data backups.
- Redundancy and failover mechanisms should be in place to mitigate data loss.

3.6.2 Availability

High Availability:

- The platform should be highly available to users, minimizing downtime and ensuring continuous service availability.
- Load balancing and fault tolerance strategies should be employed to achieve high availability.

3.6.3 Security

Authentication and Authorization:

- Robust authentication mechanisms should be implemented to verify user and watcher identities.
- Role-based access control (RBAC) should be enforced to restrict unauthorized access to sensitive functionalities.

3.6.4 Maintainability

Modularity and Documentation:

- The system should be designed with modular components to facilitate maintenance and updates.
- Code should be well-documented, following best practices and coding standards for maintainability.

3.6.5 Portability

Cross-Platform Compatibility:

- The system should be compatible with different platforms (web browsers, mobile devices) to reach a broader audience.
- Platform-specific optimizations may be applied to ensure consistent performance across devices.

3.7 Organizing the Specific Requirements

3.7.1 System Mode

Operational Modes:

1. **User Mode**:
   - Users interact with the system to report wildlife sightings, access educational resources, and receive alerts.
   - Features accessible to users include wildlife sighting reporting and educational content browsing.
2. **Watcher Mode**:
   - Watchers (forest monitors) operate in this mode to add, delete, or update wildlife information in the database.
   - Watchers have specialized functionalities for managing wildlife data and contributing real-time observations.

3.7.2 User Class

Categories:

1. **Users**:
   - Represents individuals registered on the platform.
   - Attributes include UserID, Name, Contact, Address, and Password.
2. **Watcher**:
   - Represents individuals registered on the platform and have some unique role like entering information of Wildlife.
   - Attributes include \`WatcherID\` (Primary Key), \`Name\`, \`Age\`, \`Gender\`, \`Contact\`, \`Address\`, \`Role\`, \`Password\`, \`PlaceID\` (Foreign Key)

3.7.3 Objects

Database Objects and Associated Services:

Key Objects:

User Object:

An instance of a registered user with specific attributes (e.g., name, contact details).

Watcher Object:

An instance of a forest watcher with attributes (e.g., name, age, gender) and specialized roles for wildlife monitoring.

3.7.4 Feature

Main Features:

User Login:

-Allows users and watchers to authenticate securely into the system.

Wildlife Sighting Reporting:

-Enables users to report wildlife sightings by providing details such as animal name, photo, and location.

Watcher Operations:

-Provides watchers with functionalities to add, delete, and update wildlife information based on real-time observations.

3.7.5 Stimulus

User Interaction:

- Login attempt by a user or watcher.

- Submission of a wildlife sighting report.

3.7.6 Response

System Response:

- Successful authentication and access granted to the appropriate mode (user or watcher).
- Wildlife sighting data stored in the database and made available for analysis.

3.7.7 Functional Hierarchy

User Login Process:

- Stimulus: User provides login credentials.
- Response: System verifies credentials and grants access based on user role (user or watcher).

Wildlife Sighting Reporting:

- Stimulus: User submits wildlife sighting report.
- Response: System stores the report in the database and may trigger alerts or notifications based on the reported sighting.

3.8 Additional Comments

The system should be designed with extensibility in mind, allowing for future enhancements such as advanced analytics, community engagement features, and integration with conservation initiatives.
By addressing logical database requirements, the Wildlife Monitoring System can effectively manage and utilize data to support wildlife conservation efforts, user interactions, and system operations. The database design should be scalable, efficient, and secure to handle increasing data volumes and user interactions while ensuring data integrity and reliability. Regular monitoring and optimization of the database will be essential to maintain system performance and responsiveness over time.

Organizing the specific requirements involves structuring the system's functionality, user roles, key objects, features, and interactions into a coherent framework. By defining operation modes, user classes, objects, features, stimuli, responses, and functional hierarchies, the Wildlife Monitoring System can effectively address user needs and support wildlife conservation efforts. This organization ensures clarity, efficiency, and usability in implementing the system's functionalities and user interactions. Ongoing refinement and adaptation of these organizational aspects will be essential to meet evolving user requirements and system enhancements.

4. Change Management Process

The Change Management Process within the Software Requirements Specification (SRS) for the Wildlife Watch System is crucial for adapting to evolving needs and ensuring system success. It involves identifying, evaluating, approving, and implementing proposed changes to system requirements efficiently. Stakeholders, including users, domain experts, and technical teams, participate in proposing and evaluating changes based on impact analysis, feasibility, and alignment with project goals. A team oversees the approval process, ensuring transparent decision-making. Implementing approved changes involves coordination among development, testing, and deployment teams, with thorough documentation and communication to stakeholders. The process includes monitoring the impact of changes on system performance and user satisfaction. By embracing change management, the system gains adaptability, risk mitigation, stakeholder engagement, and opportunities for continuous improvement, aligning with wildlife conservation objectives and community empowerment efforts. This systematic approach fosters resilience and responsiveness in addressing evolving requirements and technological advancements.

*******************************Thank You *********************************