**Parrot Analyzer Documentation**

**1.Introduction**

**App Overview**

Parrot Analyzer is an advanced employee geofencing and shift management application designed to provide real-time tracking, expense reporting, and team management capabilities. It enables businesses to monitor employee locations, manage shifts efficiently, and optimize workforce productivity. The app is built to support various organizational roles with tailored functionalities for each.

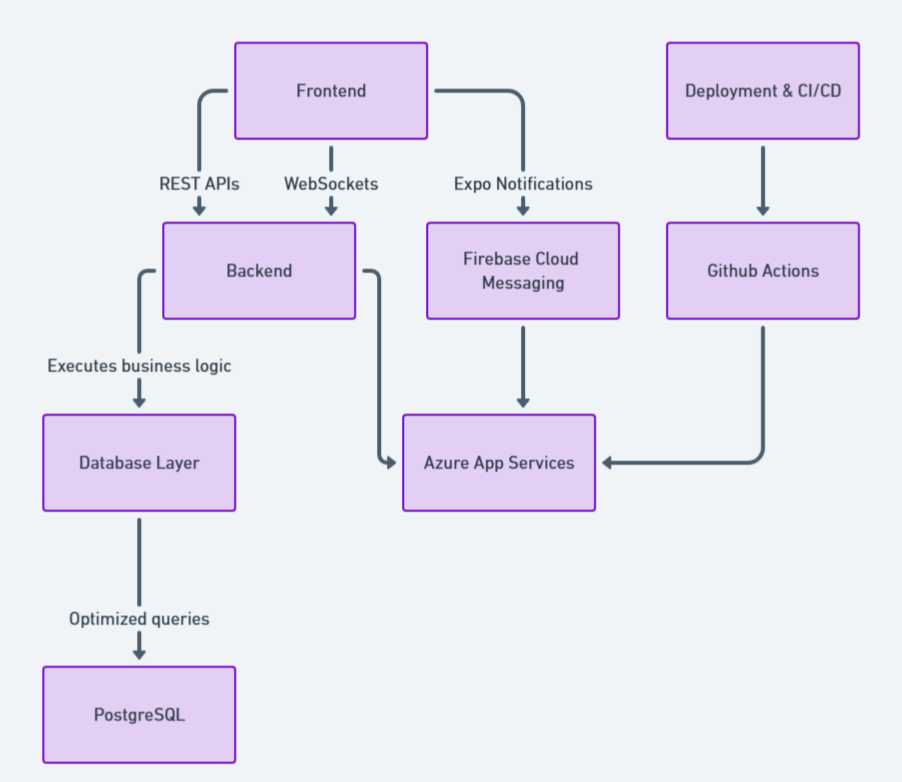
**Key Features**

* **Real-time Employee Tracking:** Uses geofencing technology to track employee movements during shifts.
* **Shift Management:** Allows Group Admins and Management Personnel to create, assign, and monitor shifts.
* **Expense Reporting:** Employees can submit expenses for approval, and admins can manage reimbursements.
* **User Role Management:** Hierarchical structure with Super Admin, Management Personnel, Group Admins, and Employees.
* **Company Management:** Super Admin can add, manage, and monitor multiple companies.
* **Bulk User Upload:** CSV import functionality for adding multiple employees or admins at once.
* **Subscription-Based Services:** Companies can manage subscriptions, track usage limits, and make payments.
* **WebSocket-Powered Live Updates:** Ensures real-time status updates on employee locations and shift changes.
* **Notifications System:** Provides alerts for shift assignments, expense approvals, and critical updates.
* **Permissions Control:** Role-based access for different features to ensure security and compliance.
* **Report Management:** Generate and download detailed reports for shifts, expenses, and workforce activities.

**Target Users**

* **Businesses & Enterprises:** Companies requiring employee tracking and shift management solutions.
* **Management Personnel:** Oversee company-wide operations, user permissions, and subscriptions.
* **Group Admins:** Manage employees, track attendance, and approve expenses.
* **Employees:** Log shifts, track time, submit expenses, and receive notifications.
* **Super Admins:** Manage multiple companies, enforce user limits, and control subscriptions.

**2.** **System Architecture  
  
High Level System Architectural Diagram:**

****

**Figure 1. High Level System Architecture**

### **Flow Explanation:**

1. **Frontend (React Native with Expo)** – Users interact with the app, which communicates with the backend via REST APIs and WebSockets for real-time tracking.
2. **Backend (Node.js with Express)** – Handles API requests, business logic, authentication (JWT), and real-time tracking.
3. **Database (PostgreSQL)** – Stores structured data like users, shifts, expenses, and permissions with optimized queries.
4. **Cloud & Deployment (Azure & Firebase)** – Backend is hosted on Azure, CI/CD is managed via GitHub Actions, and Firebase handles push notifications.
5. **Real-Time & Security** – Web
6. Sockets enable live tracking, JWT secures authentication, and RBAC controls user access.

**3. Tech Stack**

#### **Frontend (React Native with Expo)**

* **React Native** – Cross-platform framework for mobile app development.
* **Expo** – Simplifies development, testing, and deployment.
* **React Navigation** – Manages in-app navigation.
* **Redux/Context API** – Handles global state management.
* **WebSocket (Socket.io-client)** – Enables real-time tracking.
* **Firebase Cloud Messaging (FCM)** – Manages push notifications via Expo.

#### **Backend (Node.js with Express)**

* **Node.js** – JavaScript runtime for backend services.
* **Express.js** – Lightweight framework for handling API requests.
* **WebSocket (Socket.io)** – Supports real-time updates.
* **JSON Web Tokens (JWT)** – Provides authentication.
* **Firebase Storage** – Manages file uploads and storage.

#### **Database**

* **PostgreSQL** – Manages relational data (users, companies, shifts, expenses).

#### **Cloud & Deployment**

* **Azure App Services** – Hosts backend API.
* **Firebase** – Provides FCM push notifications and file storage.
* **GitHub Actions** – Automates CI/CD pipeline.

#### **Security Measures**

* **JWT Authentication** – Secure user authentication.
* **Role-Based Access Control (RBAC)** – Manages user permissions.
* **Data Encryption** – Secures sensitive database information(bcrypt).

### **4. Azure Deployment Details**

#### **4.1 Hosting Services**

* **Azure App Services** – Hosts the backend Node.js API, ensuring scalability and high availability.
* **PostgreSQL on Neon Serverless** – Provides a **scalable, managed PostgreSQL database service** with secure and reliable storage.
* **Firebase Cloud Messaging (FCM)** – Integrated for push notifications via Expo.

#### **4.2 CI/CD Pipeline Setup**

* **GitHub Actions** – Automates deployment of the backend to Azure.
* **Continuous Integration (CI)** – Runs automated tests and builds before deployment.
* **Continuous Deployment (CD)** – Deploys backend updates to Azure App Services on successful build.

#### **4.3 Security Measures**

* **JWT Authentication** – Ensures secure access for users.
* **Role-Based Access Control (RBAC)** – Restricts access based on user roles (Super Admin, Management Personnel, Group Admin, Employee).
* **Database Security** – Implements **encryption and secure connection protocols** for PostgreSQL.
* **bcrypt** – Hashes and secures user passwords to prevent unauthorized access

**5. User Roles & Permissions**

**5.1. Super Admin (Company)**

**Responsibilities:**

* **Oversee Company Management:**
  + Create, configure, and deactivate client companies.
  + Set user limits and manage company access.
* **User Account Management:**
  + Create, update, delete, and assign roles for all user accounts.
* **Global System Configuration:**
  + Configure settings such as geofencing boundaries, default shift timings, and notification parameters.
* **Monitoring & Reporting:**
  + Monitor system performance using comprehensive dashboards.
  + Generate detailed reports on attendance, leave, and expenses.
* **Security & Audit:**
  + Maintain audit logs for all administrative actions.
  + Enforce robust role-based access controls.

**Access Permissions:**

* **Full Access:**
  + Access to all system modules and data.
* **Override Authority:**
  + Ability to override actions performed by lower-level roles.
* **Data Export & Audit:**
  + Rights to export data and generate audit reports for compliance.

**Front-End Integration:**

* **Dedicated Dashboard:**
  + Interfaces for managing companies, user accounts, system settings, and viewing reports.
  + Built using React Native with Expo and styled with NativeWind for a responsive, intuitive UI.

**Back-End Integration:**

* **Secure API Endpoints:**
  + Endpoints such as /admin/companies, /admin/users, and /admin/settings are secured with role-based middleware.
* **Database Storage:**
  + All relevant data (company details, user roles, system configurations, and audit logs) is stored in PostgreSQL.
* **Security Measures:**
  + Ensures secure data transmission and comprehensive data protection protocols.

**Workflow & Processes:**

* **Login & Dashboard Access:**
  + Super Admin logs in to a unified dashboard with full access to all system configurations.
* **Management Activities:**
  + Perform company and user management, adjust global settings, and monitor overall system performance.
* **Audit & Compliance:**
  + Every administrative action is logged for transparency and compliance, ensuring complete control over the platform.

**5.2. Management Personnel**

**Management Personnel (Admin Level)**

**Responsibilities:**

* **Leave Policy Configuration:**
  + Create, update, and manage various leave types (e.g., Earned, Sick, Casual, Maternity, etc.).
  + Set entitlements, carry-forward rules, eligibility criteria, and maximum consecutive leave limits.
* **Aggregated Analytics & Reporting:**
  + Access high-level dashboards showing key performance metrics such as team attendance, leave utilization, expense trends, and travel efficiency.
  + Generate exportable reports (PDF/Excel) for strategic decision-making and compliance tracking.
* **Escalation Handling:**
  + Review and process escalated leave requests forwarded by Group Admins.
  + Approve or reject exceptional cases and policy exceptions.
* **Group Admin Oversight:**
  + Monitor and manage Group Admin accounts and the performance of their assigned teams.
  + Create and manage Group Admin accounts, either individually or via bulk CSV import.

**Access Permissions:**

* **Aggregated Data Access:**
  + View organization-wide leave and attendance reports, aggregated across all groups.
* **Configuration Privileges:**
  + Full access to interfaces for setting leave policies and managing escalated requests.
* **Administrative Controls:**
  + Rights to override or adjust leave requests, and perform high-level analytics.

**Front-End Integration:**

* **Dedicated Management Dashboard:**
  + A comprehensive dashboard displaying quick action panels (e.g., Shift Tracker, Leave Insights).
  + Interactive forms and tools for configuring leave policies and managing escalated requests.
  + Visual components (charts, summary cards) to monitor team performance and attendance trends.
  + Developed using React Native with Expo, styled with NativeWind for a modern, responsive interface.

**Back-End Integration:**

* **Secure API Endpoints:**
  + Endpoints such as /management/leave-policies, /management/reports, and /management/escalations support configuration and data retrieval.
  + Role-based middleware ensures only Management Personnel can access these endpoints.
* **Database Storage:**
  + PostgreSQL is used to store leave policy configurations, aggregated leave data, and analytics.
  + Data is organized to support efficient querying and report generation.

**Workflow & Processes:**

* **Policy Setup & Monitoring:**
  + Management Personnel log in to a dedicated dashboard to configure leave policies and set entitlements.
  + They continuously monitor aggregated leave trends and overall team performance.
* **Escalation Processing:**
  + Escalated leave requests from Group Admins are reviewed and processed through the management interface.
  + Decisions (approval, rejection) are logged and communicated back to Group Admins.
* **Analytics & Reporting:**
  + Regular generation of reports and export of data to support HR and compliance audits.
  + Ongoing analysis of key metrics to inform strategic adjustments in leave policies and overall workforce management.

**5.3. Group Admin**

**Group Admin (Admin Level)**

**Responsibilities:**

* **Employee Management:**
  + Add individual employees or import bulk employee data via CSV.
  + Search, view, and delete employee records.
  + Access detailed employee profiles for performance and attendance monitoring.
* **Expense Management:**
  + Review employee expense submissions, including travel details and supporting documentation.
  + Approve or reject expense requests, with the ability to provide feedback.
  + Monitor overall expense trends within the group.
* **Task Management:**
  + Create and assign tasks to employees, capturing details such as task title, description, priority, and due date.
  + View and filter task statuses (e.g., In Progress, Pending, Completed) and track progress via a task progress bar.
* **Attendance Management:**
  + Monitor attendance records using a calendar view and filter options for individual or group-wide tracking.
  + Review metrics like total shifts, average hours per day, and overall attendance performance.
* **Live Tracking & Shift Tracking:**
  + Use real-time live tracking to monitor employee locations via an interactive map.
  + Oversee shift tracking and ensure accurate attendance logging through integrated GPS data.
* **Leave Management:**
  + Manage leave requests by reviewing, approving, or rejecting submissions from employees.
  + Access a "Leave Insights" section to track leave balances and monitor leave trends within the group.
* **Reporting & Analytics:**
  + Generate and export detailed reports (e.g., Expense, Attendance, Task, Travel, Performance, and Leave Reports) with filtering options.
  + Visualize data using charts (line, bar, pie) to aid decision-making.
* **Profile & Dashboard:**
  + Access a personal dashboard displaying key statistics such as recent activities, recent task updates, and overall group performance.
  + Edit personal profile details, change password, and toggle theme settings.

**Access Permissions:**

* Full access to employee data within their group.
* Ability to manage leave, expense, task, and attendance records specific to their group.
* Rights to generate group-specific reports and export data as needed.
* Authority to process leave and expense requests, with escalation options to Management Personnel when necessary.

**Front-End Integration:**

* A dedicated Group Admin dashboard with multiple sections (Employee Management, Expense Management, Task Management, Attendance, Live Tracking, and Leave Management).
* User interfaces built with React Native using Expo, styled with NativeWind for a modern, responsive design.
* Components include CSV import functionality, interactive calendars, dynamic maps, action buttons (approve/reject/escalate), and summary cards for quick insights.

**Back-End Integration:**

* RESTful API endpoints such as /group/employees, /group/leave, /group/expenses, /group/tasks, and /group/attendance for all data operations.
* Real-time communication via Socket.io for live tracking updates.
* Data stored in PostgreSQL, with tables linking employee records, leave requests, tasks, expenses, and attendance logs to specific groups.
* Secure role-based middleware ensures that only Group Admins can access and modify group-specific data.

**Workflow & Processes:**

* **Login & Dashboard Access:** Group Admin logs in to a tailored dashboard displaying a summary of pending leave and expense requests, employee performance metrics, and live tracking data.
* **Employee & Request Management:** From the dashboard, Group Admins can search and manage employee records, view detailed profiles, and process incoming leave and expense requests.
* **Approval & Escalation:** Leave and expense requests are processed via action buttons, with the option to escalate issues to Management Personnel if required.
* **Reporting:** Group Admins can generate detailed reports with filtering options and export them in PDF format for further analysis.
* **Ongoing Monitoring:** Real-time updates and notifications (when implemented) ensure that Group Admins are continuously informed about employee activities and pending requests.

**5.3 Employee**

**Employee (User Level)**

**Responsibilities:**

* **Schedule & Self-Tracking:**
  + View and add personal schedules using an editable calendar.
  + Maintain a detailed schedule plan with date-specific entries to track work shifts and activities.
* **Expense Management:**
  + Submit expense reports with comprehensive details:
    - Company information, form code, and employee details (name, employee number, department, designation, location, date).
    - Travel information such as vehicle type, vehicle number, total distance, start and end dates/times, total travel time, average speed, and route details.
    - Expense breakdown including lodging, daily allowances/food, diesel, toll charges, and other miscellaneous expenses.
    - Financial summary highlighting total expenses, any advance taken, and amount payable.
  + Upload supporting documents via camera or file uploads.
  + Access a dedicated "My Expenses" section with filtering options and an overview of total claimed versus approved amounts.
* **Leave Insights:**
  + View active leave policies with detailed dropdowns (default entitlement, carry-forward rules, eligibility, maximum consecutive days, special conditions).
  + Submit leave requests and track the status (pending, approved, rejected, or escalated) along with leave balance and history.
* **Task Management:**
  + View and manage "My Tasks" on the dashboard, including status updates (in progress, pending, completed).
  + Use a task progress bar to monitor the completion of assigned tasks.
* **Profile & Personal Dashboard:**
  + Access a personal profile section displaying a profile photo, personal details (name, email, phone, role), and reporting hierarchy (reports to: Group Admin).
  + View key performance metrics such as total hours worked, expenses submitted, attendance, and tasks.
  + Monitor monthly progress through a visual progress bar covering working hours, expense claims, attendance rate, and task completion.
  + Review recent activities and have easy access to editing profile details, changing passwords, toggling theme settings, and accessing support (help center, contact support, terms, and privacy).
  + Contact Support includes a live chat feature (powered by Gemini AI) for 24-hour assistance and a contact form that submits inquiries via email (using nodemailer).

**Access Permissions:**

* Full access to personal data related to attendance, expenses, leave, and tasks.
* Ability to submit and edit their own expense reports, leave requests, and schedules.
* Restricted to view only their own task progress and profile information.

**Front-End Integration:**

* Developed using React Native with Expo, featuring an intuitive, mobile-optimized interface styled with NativeWind.
* Components include interactive calendars, form fields for expense submission and leave applications, dynamic dashboards, and progress indicators.
* The UI provides clear navigation for switching between schedule, expense, leave, task, and profile sections.

**Back-End Integration:**

* RESTful API endpoints manage data operations for personal scheduling, expense submissions, leave management, and task tracking.
* PostgreSQL stores all employee-related data (attendance records, expense reports, leave requests, task details, and personal profiles).
* Role-based middleware ensures that employees access only their own data and functions.

**Workflow & Processes:**

* **Daily Use:** The employee logs into the app to view and update their schedule, submit expenses as needed, apply for leave, and manage tasks assigned by Group Admins.
* **Expense & Leave Requests:** Submitted requests are processed by Group Admins, with status updates reflected in the employee's dashboard.
* **Task Progress:** Employees update task statuses to indicate progress, and the dashboard aggregates task completion data in real time.
* **Profile Updates:** Employees can update their personal information and view key statistics about their work performance over time.
* **Support Interaction:** For any issues or queries, employees access live chatbot support via Gemini AI or use the contact form to get direct assistance.

# ****5. Installation & Setup Guide****

## ****5.1 System Requirements****

### **Minimum Device Requirements**

* **Android:** Android 9 (Pie) or later, minimum 4GB RAM
* **iOS:** iOS 13 or later, compatible with iPhone & iPad
* **Web:** Latest versions of Chrome, Firefox, Edge, or Safari

### **Server Requirements**

* **Operating System:** Ubuntu 20.04+ / Windows Server 2019+
* **Processor:** Minimum Quad-core CPU
* **Memory:** Minimum 8GB RAM
* **Storage:** Minimum 50GB SSD
* **Database:** PostgreSQL (latest stable version)
* **Node.js Version:** Latest LTS version
* **Expo CLI:** Latest stable version

## ****5.2 Installation Steps****

### **Clone the Repository**

To install and set up **Parrot Analyzer**, first clone the repository:

git clone [repository-url]

cd ParrotAnalyzer

### **Frontend Setup (React Native + Expo)**

1. Navigate to the frontend directory:
2. cd ParrotAnalyzer
3. Install dependencies:
4. npm install
5. # or
6. yarn install
7. Start the Expo development server:
8. npx expo start
9. Choose a platform to run the app:
   * Press **'a'** for Android
   * Press **'i'** for iOS (MacOS required)
   * Press **'w'** for Web

### **Backend Setup (Node.js + Express.js + PostgreSQL)**

1. Navigate to the backend directory:
2. cd backend
3. Install dependencies:
4. npm install
5. # or
6. yarn install
7. Configure the **.env** file (Environment Variables)  
   Create a **.env** file in the backend directory and configure the following variables:
8. DATABASE\_URL=your\_postgresql\_url
9. PORT=3000
10. GOOGLE\_GEMINI\_API\_KEY=your\_gemini\_api\_key
11. EMAIL\_USER=your\_email
12. EMAIL\_PASS=your\_email\_password
13. Run the backend server:
14. npm run dev
15. # or
16. yarn dev

## ****5.3 Azure Deployment Guide****

### **Setting Up Azure Resources**

1. **Create an Azure Web App**
   * Navigate to **Azure Portal** → **App Services**
   * Click **Create App Service**
   * Select **Node.js environment**
   * Deploy using GitHub Actions or Azure DevOps
2. **Create an Azure PostgreSQL Database**
   * Go to **Azure Database for PostgreSQL**
   * Click **Create a New Database**
   * Configure storage, networking, and security settings
   * Retrieve the **database URL** and update the .env file
3. **Environment Variables & API Keys**
   * Store secrets in **Azure Key Vault** or **Application Settings**
   * Ensure that API keys and database URLs are securely configured
4. **CI/CD Deployment using Azure DevOps or GitHub Actions**
   * Configure **GitHub Actions** to automatically deploy new updates
   * Use **Azure DevOps Pipelines** for staging and production environments

## ****5.4 Environment Configuration****

### **Frontend (.env) Configuration**

Create a .env file in the **root frontend directory**:

EXPO\_PROJECT\_ID=your\_expo\_project\_id

EXPO\_PUBLIC\_API\_URL=your\_backend\_api\_url

### **Backend (.env) Configuration**

Ensure that all backend secrets are properly configured:

DATABASE\_URL=your\_postgresql\_url

PORT=3000

EMAIL\_USER=your\_email

EMAIL\_PASS=your\_email\_password

GOOGLE\_GEMINI\_API\_KEY=your\_gemini\_api\_key

## ****5.5 Running the Application****

### **Running Locally**

1. Start the **backend server**
2. cd backend
3. npm run dev
4. Start the **frontend application**
5. npx expo start
6. Access the app on an emulator or real device via Expo Go

## ****5.6 Troubleshooting & Common Issues****

| **Issue** | **Cause** | **Solution** |
| --- | --- | --- |
| **Port 3000 already in use** | Another process is running on the same port | Run npx kill-port 3000 or change PORT in .env |
| **Expo server not starting** | Expo CLI is outdated | Run npm update -g expo-cli |
| **Database connection error** | Incorrect PostgreSQL URL | Verify DATABASE\_URL in .env |
| **CORS errors** | API requests blocked | Configure CORS in Express backend |

**Glossary of Key Terms for Parrot Analyzer**

Below is a glossary of key terms used throughout the Parrot Analyzer documentation. It is recommended to include this section immediately after the Table of Contents to help new developers and stakeholders quickly understand the technical language used in the project.

1. **RBAC (Role-Based Access Control):**  
   A security model that restricts system access to authorized users based on their roles, ensuring each user only accesses the features and data necessary for their responsibilities.
2. **JWT (JSON Web Token):**  
   A compact, URL-safe means of representing claims to be transferred between two parties. It is used for secure authentication and authorization throughout the system.
3. **Geofencing:**  
   A location-based service that creates virtual boundaries using GPS. In Parrot Analyzer, geofencing is used to filter out indoor movements, ensuring that only outdoor travel is accurately tracked.
4. **Live Tracking:**  
   The real-time monitoring of employee locations via GPS, which is displayed on interactive maps. This feature enables supervisors to monitor movements and calculate travel metrics accurately.
5. **Attendance Management:**  
   A module that allows employees to log their shifts by starting and ending attendance records. It automatically records timestamps and GPS data to compute work hours.
6. **Expense Management:**  
   The process by which employees submit expense reports for work-related costs, including detailed travel and expense breakdowns. These reports are reviewed and approved by administrators.
7. **Leave Management:**  
   A system that enables employees to apply for leave, track leave balances, and review leave history, with multi-level approval workflows involving Group Admins and Management Personnel.
8. **Socket.io:**  
   A JavaScript library that enables real-time, bi-directional communication between clients and servers. It is used in Parrot Analyzer to power live tracking and instant notifications.
9. **React Native with Expo:**  
   A cross-platform mobile application framework that allows developers to build native apps using JavaScript. Expo provides tools and services that simplify development, testing, and deployment.
10. **NativeWind:**  
    A utility-first CSS framework for styling React Native applications, inspired by TailwindCSS, which enables the creation of modern and responsive user interfaces.
11. **PostgreSQL:**  
    An open-source relational database management system used to store and manage structured data (such as users, shifts, expenses, and leave records) in Parrot Analyzer.
12. **Node.js & Express.js:**  
    Node.js is a JavaScript runtime that allows for building scalable backend services, while Express.js is a lightweight framework used to create RESTful APIs.
13. **Gemini AI:**  
    An AI-powered live chatbot support tool integrated into Parrot Analyzer to assist employees with real-time queries and provide interactive guidance.
14. **Azure:**  
    Microsoft’s cloud platform used for hosting and deploying backend services, ensuring scalability, high availability, and performance.
15. **Firebase Cloud Messaging (FCM):**  
    A cross-platform messaging solution that enables the delivery of push notifications to users, ensuring real-time alerts and updates.
16. **CI/CD (Continuous Integration/Continuous Deployment):**  
    Practices and tools used to automate the building, testing, and deployment of the application, ensuring smooth and reliable updates.