Project Outline

GWIIT

General Workplace Inventory & Inspection Tool

Contents

[Purpose 3](#_Toc175745695)

[Regulatory Context 4](#_Toc175745696)

[User Personas 5](#_Toc175745697)

[Site Safety Manager 6](#_Toc175745698)

[Site Safety Specialist 6](#_Toc175745699)

[Seasonal / Intern Safety Coordinator 6](#_Toc175745700)

[Facilities Manager 7](#_Toc175745701)

[Facilities Technician 7](#_Toc175745702)

[Feature List 8](#_Toc175745703)

[Core Functionality Features 9](#_Toc175745704)

[Inspection Management 9](#_Toc175745705)

[Task and Scheduling Management 11](#_Toc175745706)

[Ticketing System 12](#_Toc175745707)

[Data Management 14](#_Toc175745708)

[Reporting 15](#_Toc175745709)

[User Experience and Accessibility Features 16](#_Toc175745710)

[Interface Design 16](#_Toc175745711)

[User Interaction 17](#_Toc175745712)

[Accessibility and Inclusivity 18](#_Toc175745713)

[Administrative and Reporting Features 19](#_Toc175745714)

[User and Role Management 19](#_Toc175745715)

[Data and Reporting 20](#_Toc175745716)

[Monitoring and Alerts 21](#_Toc175745717)

[Offline Capabilities and System Integration Features 22](#_Toc175745718)

[Offline Functionality 22](#_Toc175745719)

[Data Synchronization 23](#_Toc175745720)

[System Integration 24](#_Toc175745721)

[Security and Continuous Improvement Features 25](#_Toc175745722)

[Security Management 25](#_Toc175745723)

[Continuous Improvement 26](#_Toc175745724)

[Stack 27](#_Toc175745725)

[Frontend (Client-Side) 27](#_Toc175745726)

[React.js: 27](#_Toc175745727)

[React-Image-Annotate (or similar): 27](#_Toc175745728)

[Moment.js (or date-fns): 27](#_Toc175745729)

[Bootstrap: 27](#_Toc175745730)

[Chart.js: 27](#_Toc175745731)

[Axios: 28](#_Toc175745732)

[Backend (Server-Side) 29](#_Toc175745733)

[Django: 29](#_Toc175745734)

[Django REST Framework (DRF): 29](#_Toc175745735)

[Django-ImageKit (or similar): 29](#_Toc175745736)

[Django-Taggit: 29](#_Toc175745737)

[Django-Allauth: 29](#_Toc175745738)

[Django Guardian: 30](#_Toc175745739)

[Database 31](#_Toc175745740)

[PostgreSQL: 31](#_Toc175745741)

[Data Management and Reporting 32](#_Toc175745742)

[Django Admin: 32](#_Toc175745743)

[Authentication and Security 33](#_Toc175745744)

[Django's Built-in Authentication System: 33](#_Toc175745745)

[Deployment 34](#_Toc175745746)

[Nginx: 34](#_Toc175745747)

[Gunicorn: 34](#_Toc175745748)

[DigitalOcean: 34](#_Toc175745749)

[Development Workflow Tools 35](#_Toc175745750)

[Git/GitHub: 35](#_Toc175745751)

# Purpose

The web application being developed addresses the critical need for streamlined workplace safety inspections and compliance management within small to medium-sized businesses. Currently, managing the inspection process involves manual tracking, which can lead to missed inspections, incomplete records, and potential non-compliance with Federal, State, and local standards. This web app aims to simplify and automate the entire process, ensuring that all workplace inspections are properly documented and maintained in accordance with best practices and regulatory standards.

By integrating features such as QR code scanning, real-time notifications, photo documentation, and comprehensive reporting, the app not only enhances compliance but also reduces the administrative burden on personnel. It enables users to perform inspections efficiently, create and maintain a ticketing system to correct errors, provides administrators with robust tools for managing compliance, and ensures that all necessary records are securely stored and easily accessible for audits. In doing so, this web app improves inspections and record keeping allowing businesses to focus on their core operations with confidence in their compliance.

# Regulatory Context

Ensuring compliance with OSHA and NFPA standards is crucial for the effective management of safety in the workplace. The web application is designed to help businesses meet these regulatory requirements, which include specific guidelines for the placement, inspection, maintenance, and record-keeping of portable fire extinguishers, emergency exits, first aid kits, and much more.

# User Personas

This section outlines the key user personas who will interact with the web application. User personas are fictional representations of the different types of users that will benefit from the app. Understanding a potential user base helps to understand their needs, challenges, and behaviors. By defining these personas, we can ensure that the application is tailored to meet the specific requirements of each user group, enhancing overall usability and effectiveness. These personas will guide the development process, ensuring that the features and functionality are aligned with the real-world scenarios and objectives of a potential user.

## Site Safety Manager

**Name:** Kevin, age 50

**Technical Proficiency:** Has an advanced understanding of standard enterprise software. Ability to learn new technology with training.

**Goals:** To improve safety compliance across site. Decrease time for Specialists and Coordinators to do inspections, creating more time for preventative work.

**Challenges:** Limited time for data analysis, requires quick access to accurate information, manages multiple employees with varying levels of knowledge and ability. Responsible for overall Safety, Health, and Environmental compliance for location. Cell phone service and wi-fi signals at the site are spotty.

**Behaviors:** Regularly uses a desktop or laptop computer for work. Prefers straightforward and efficient software that is easy to navigate. Uses a mobile device while not in the office to perform, note, or document issues within the worksite.

## Site Safety Specialist

**Name:** Derek, age 35

**Technical Proficiency:** Has a beginner understanding of standard enterprise software. Ability to learn new technology with training, but may need job aides while first using new technology.

**Goals:** To improve safety compliance across site. Accurately and quickly perform inspections.

**Challenges:** Limited ability for data analysis, requires quick access to accurate information, must be able to quickly perform inspections, frequently interrupted while performing tasks and inspections. Cell phone service and wi-fi signals at the site are spotty.

**Behaviors:** Regularly uses a desktop or laptop computer. Needs to be able to use mobile device when performing inspections and audits.

## Seasonal / Intern Safety Coordinator

**Name:** Clint, age 22

**Technical Proficiency:** Has a rudimentary understanding of standard enterprise software. Ability to learn new technology with training, but may need job aides while first using new technology.

**Goals:** To perform inspections quickly and accurately.

**Challenges:** Must be able to quickly perform inspections. Has little to no Safety, Health, and Environmental training, received minimal job training, seasonal / intern employee with limited time to master enterprise specific technology. Cell phone service and wi-fi signals at the site are spotty.

**Behaviors:** Occasionally uses a laptop computer. Needs to be able to use mobile tablet device when performing inspections.

## Facilities Manager

**Name:** Lisa, age 45

**Technical Proficiency:** Intermediate understanding of enterprise software and maintenance management systems.

**Goals:** Ensure facility is maintained to a high standard. Quickly address and resolve issues identified during inspections. Maintain efficiency while ensuring compliance.

**Challenges:** Coordinating with multiple teams to address maintenance issues. Balancing budget constraints with need for quick outcomes. Ensuring all tasks are completed quickly, despite varying levels of urgency.

**Behaviors:** Frequently uses a desktop or laptop to manage work orders and schedules. Uses mobile devices to check updates on the go. Prefers tools that integrate with existing facility management systems and provide real-time updates.

## Facilities Technician

**Name:** Mike, age 30

**Technical Proficiency:** Basic understanding of enterprise software; proficient with mobile apps and handheld devices.

**Goals:** Respond quickly to maintenance requests and issues identified during inspections. Ensure that all repairs and maintenance tasks are completed efficiently and correctly.

**Challenges:** Managing a high volume of maintenance tasks. Navigating and updating work orders on mobile devices. Dealing with intermittent connectivity issues affecting mobile connectivity.

**Behaviors:** Primarily uses mobile devices and tablets to receive and update work orders. Prefers straightforward, easy-to-use interfaces that minimize time spent on administrative tasks. Frequently works in areas with poor connectivity, so offline capabilities are essential.

# Feature List

The web application is designed to streamline the management, inspection, and compliance tracking of fire extinguishers across various sites. Built with a focus on usability and efficiency, the app provides a comprehensive suite of features tailored to the needs of safety managers and specialists. By integrating core functionalities such as QR code scanning, real-time notifications, and detailed reporting, the app ensures that all fire safety equipment is regularly inspected and maintained according to regulatory standards. Additionally, the app offers robust user management and accessibility features, making it easy for teams of all sizes to ensure compliance. Each feature is developed with the end-user in mind, aiming to reduce the administrative burden and improve overall compliance across the organization.

## Core Functionality Features

### Inspection Management

#### Feature: Location Documentation Interface

**Purpose:** Provide a clear and organized interface where users can document locations where fire extinguishers are installed.

**Example:** Users can access a list view showing all documented fire extinguisher locations, making it easier to perform inspections.

#### Feature: Filter and Search

**Purpose:** Advanced filtering and search options, allows users to quickly find specific locations based on building, floor, or other location data.

**Example:** Users can access a filtered list showing all documented fire extinguisher meeting the search or filter criteria the user entered.

#### Feature: QR Code Generator

**Purpose:** Automatically generate and assign QR codes to individual fire extinguishers and locations.

**Example:** When setting up the app for a new site, the administrator generates and prints QR codes to be placed on each fire extinguisher.

#### Feature: Batch QR Code Generator

**Purpose:** Allow administrators to generate QR codes in batches for large installations, streamlining the setup process.

**Example:** When setting up the app for a new site, the administrator generates and prints QR codes in batches to ease installation and updates the QR Code data after/as they are placed.

#### Feature: Customizable QR Code

**Purpose:** Options to include additional/selectable information in the QR code (e.g., make, model, location, service history, etc.) that can be selected during initial setup from a list or tick boxes of options.

**Example:** When setting up the app for a new site, the administrator selects what information they want to be stored in the QR codes for their site.

#### Feature: QR Code Scanner

**Purpose:** Allow users to scan QR codes to quickly access and log inspection details.

**Example:** During an inspection, a safety specialist scans a fire extinguisher’s QR code to pull up its history and log the current inspection.

#### Feature: Offline QR Code Scanner

**Purpose:** QR code scanner works offline, with data syncing when connectivity is restored.

**Example:** When unable to connect with data service, the QR code scanning, inspecting, or recording of QR code data can be saved on device and synced when data connectivity is restored.

#### Feature: Error Handling QR Code Scanner

**Purpose:** Error handling for instances where a QR code is unreadable or the associated extinguisher data is missing.

**Example:** Allow users secondary or tertiary options to finish task when there is an issue with the QR code.

#### Feature: Photo Documentation

**Purpose:** Enable users to take photos of fire extinguishers and surroundings.

**Example:** An inspector takes a photo of a fire extinguisher or the area surrounding it.

#### Feature: Timestamp and/or Geotagging

**Purpose:** Automatically add timestamps and/or geotagging to photos to provide context for when and where the photo was taken.

**Example:** An inspector takes a photo of a fire extinguisher and marks a concern directly on the image, such as a blocked access path.

#### Feature: Photo Annotation Tools

**Purpose:** Various annotation tools such as drawing, text labels, and shapes to mark different types of concerns directly on the image.

**Example:** An inspector takes a photo of a fire extinguisher and marks a concern directly on the image, such as a blocked access path.

#### Feature: Interactive Map Integration

**Purpose:** Interactive map that allows administrator to drop pins for extinguisher locations and users to view them in relation to site layouts.

**Example:** Users can access a map view showing all documented fire extinguisher locations, making it easier to perform inspections.

### Task and Scheduling Management

#### Feature: Inspection Scheduling and Task Management

**Purpose:** Process for scheduling inspections and assigning tasks to specific users

**Example:** Administrator can schedule recurring inspections or tasks to specific users.

#### Feature: Recurring Scheduling and Tasks

**Purpose:** Set up recurring inspection schedules that auto-generate tasks at predefined intervals.

**Example:** Administrator can schedule recurring inspections or tasks to specific users.

#### Feature: Real-time Notifications and Alerts for tasks

**Purpose:** Notify users of upcoming inspections or missed tasks.

**Example:** The app sends an alert to the safety manager when an inspection is overdue, prompting immediate action.

#### Feature: Customizable Alerts for tasks

**Purpose:** Allow users to customize the types of alerts they receive, such as email, or in-app.

**Example:** Users can set up alerts to come to their email or decline email notifications.

#### Feature: Missed task escalations

**Purpose:** Escalation process where missed tasks trigger alerts to assigned person(s).

**Example:** If a user misses a task(s) a notification will be sent to the user and the manager they are assigned.

### Ticketing System

#### Feature: Ticket Creation

**Purpose:** Allow users to create a ticket when an issue is found during an inspection.

**Example:** During an inspection, an inspector finds a damaged fire extinguisher. They create a ticket detailing the issue, which is then assigned to the maintenance team for resolution.

#### Feature: Ticket Assignment

**Purpose:** Enable tickets to be assigned to specific users, teams, managers, or administrators.

**Example:** After creating a ticket for a damaged fire extinguisher, the inspector assigns it to the maintenance team for action. The ticket is also copied to the site manager for oversight.

#### Feature: Ticket Reassignment

**Purpose:** Allow administrators or users with appropriate permissions to reassign tickets to different users or teams as needed.

**Example:** If the maintenance team is unavailable, the site manager reassigns the ticket to another qualified team or individual to ensure the issue is resolved promptly.

#### Feature: Ticket Due Date Management

**Purpose:** Set and manage due dates for ticket completion, with options to modify deadlines as necessary.

**Example:** The site manager sets a due date of two days for resolving the damaged extinguisher ticket. If more time is needed, the due date can be extended, and all relevant parties are notified.

#### Feature: Ticket History and Audit Trail

**Purpose:** Maintain a detailed history of all changes made to a ticket, including who made the changes and when.

**Example:** The audit trail for the damaged extinguisher ticket shows that the due date was extended by the site manager and that the ticket was reassigned from the original maintenance team to another team.

#### Feature: Ticket Dashboard

**Purpose:** Provide a dashboard for users to monitor open, in-progress, and closed tickets, with options to filter by priority, due date, or assigned team.

**Example:** The site manager accesses the ticket dashboard to view all open tickets, including those approaching their due dates. They can also filter tickets by priority to address the most urgent issues first.

#### Feature: Compliance Monitoring

**Purpose:** Track the compliance of completing tickets within the assigned deadlines and escalate overdue tickets as needed.

**Example:** The system flags a ticket for a missed inspection that hasn't been resolved within the deadline. The ticket is escalated to higher management, and a compliance report is generated to document the delay.

#### Feature: Notification and Alerts for Tickets

**Purpose:** Notify users and administrators of ticket updates, including new assignments, due date changes, and completed tickets.

**Example:** The maintenance team receives an in-app notification when a new ticket is assigned to them, and the inspector receives an alert when the ticket is resolved.

#### Feature: Role-Based Access Control for Tickets

**Purpose:** Control access to ticketing functions based on user roles, ensuring that only authorized users can create, view, or modify tickets.

**Example:** Regular users can create tickets and view their status, while managers have the ability to reassign tickets, change due dates, and generate reports.

#### Feature: Reporting on Tickets

**Purpose:** Generate reports that detail ticket status, resolution times, compliance with deadlines, and other relevant metrics.

**Example:** At the end of the month, the system generates a report summarizing all tickets created, resolved, and any that missed their deadlines. The report can be used for internal reviews or audits.

### Data Management

#### Feature: Database Management System

**Purpose:** Manage and store data related to users, fire extinguishers, inspection locations, and completed tasks.

**Example:** Administrators can access the database to retrieve past inspection records, ensuring all data is securely stored onsite.

#### Feature: Database Backup and Recovery

**Purpose:** Automated backup and recovery processes to prevent data loss.

**Example:** Administrators can set up back up features for the database(s) on implementation. Options for back up based on time/date, new tasks/inspections, changes to admin options, etc.

#### Feature: Automated Backup and Recovery

**Purpose:** Regularly back up data and provide recovery options in case of system failure.

**Example:** In the event of a system crash, the app automatically restores data from the most recent backup, ensuring continuity.

#### Feature: External Database Integration

**Purpose:** Allow connection to external databases for user management and data synchronization.

**Example:** The app integrates with a company’s existing SQL database, allowing centralized user management across multiple sites.

### Reporting

#### Feature: Reporting

**Purpose:** Generate detailed reports that demonstrate compliance with OSHA and NFPA standards.

**Example:** System generates reports summarizing inspection activities, compliance status, and any outstanding issues. Admin can set reporting intervals, or generate immediate report.

#### Feature: Reporting Templates

**Purpose:** Pre-built report templates that align with industry best practices.

**Example:** Allow site admins and role-based users to produce different reports with varying levels of detail.

#### Feature: Export Options

**Purpose:** Reports or general data allowed for export in various formats, such as PDF, Excel, or CSV, for ease of sharing and record-keeping.

**Example:** Allow site admins and role-based users export data with varying levels of detail.

## User Experience and Accessibility Features

### Interface Design

#### Feature: Cross-Platform Compatibility

**Purpose:** Ensure the app operates on mobile devices, tablets, PCs, and MACs.

**Example:** A user can start an inspection on a tablet and later review the data on a desktop without any loss of functionality.

#### Feature: Adaptive Layouts for Varying Screen Sizes

**Purpose:** Ensure the app’s layout adapts not just to different devices but also to varying screen sizes and resolutions, providing an optimal viewing experience on all devices.

**Example:** On smaller screens, the app automatically switches to a single-column layout, while on larger screens, it displays additional columns and information.

#### Feature: Responsive Design

**Purpose:** Automatically adjust the app’s layout based on the user’s device screen size and orientation.

**Example:** The app’s layout adapts to both landscape and portrait orientations, providing an optimal user experience on any device.

#### Feature: Customizable User Interface (UI)

**Purpose:** Allow users to customize the interface according to their preferences, including font size, color schemes, and layout options.

**Example:** A user with visual impairments can increase the font size and adjust the color contrast to make the text easier to read.

#### Feature: Dark Mode

**Purpose:** Provide a dark mode option for users who prefer a darker interface, reducing eye strain in low-light environments.

**Example:** Users can toggle between light and dark modes in the app’s settings, with the interface colors automatically adjusting for better visibility.

#### Feature: Customizable Dashboard Widgets

**Purpose:** Allow users to customize their dashboard with widgets that display the most relevant information for their role.

**Example:** A safety specialist can add a widget to their dashboard that shows upcoming inspections, while a site manager might prioritize a widget showing overdue tasks.

#### Feature: Task Dashboard

**Purpose:** Dashboard where users and administrators can view upcoming tasks, deadlines, and priority levels. RBAC to limit users to only view users and admin to have more viewing capability.

**Example:** Users are able to see upcoming and completed task overview and due dates. Higher level users can see data for those users assigned to them.

### User Interaction

#### Feature: Keyboard Shortcuts

**Purpose:** Implement keyboard shortcuts for common actions, allowing power users to navigate the app more efficiently without relying on a mouse.

**Example:** Pressing "Ctrl + S" could save an inspection report, or "Ctrl + F" could open the search function.

#### Feature: Error Prevention and Correction Mechanisms

**Purpose:** Include features that help prevent errors, such as confirmation dialogs for critical actions, and provide clear instructions on how to correct mistakes.

**Example:** Before deleting an inspection record, the app asks for confirmation and warns the user of the consequences. If an error is made, the app provides step-by-step instructions to correct it.

#### Feature: Seamless Onboarding Process

**Purpose:** Provide a smooth onboarding process for new users, including account setup, role assignment, and introductory tutorials.

**Example:** New users are guided through setting up their profile, selecting their role, and completing an introductory tutorial that covers essential app features.

#### Feature: Contextual Help and Tooltips

**Purpose:** Provide contextual help and tooltips throughout the app, guiding users through more complex features or explaining new functionalities.

**Example:** When a user hovers over a new button or feature, a tooltip appears explaining what it does and how to use it.

#### Feature: Feedback Collection

**Purpose:** Provide a simple, in-app method for users to give feedback or report issues, ensuring continuous improvement based on real user experiences.

**Example:** A feedback button in the navigation bar allows users to quickly rate their experience, suggest improvements, or report bugs.

### Accessibility and Inclusivity

#### Feature: Accessibility Features

**Purpose:** Implement screen reader compatibility, keyboard navigation, and adjustable text sizes.

**Example:** A user with visual impairments can navigate the app using a screen reader, and all content is accessible via keyboard shortcuts.

#### Feature: Multilingual Support

**Purpose:** Offer the app’s interface and content in multiple languages.

**Example:** A user selects their preferred language from the settings, allowing them to use the app in their native language.

#### Feature: Voice Command Integration

**Purpose:** Enable users to navigate and perform actions in the app using voice commands, providing an alternative input method for users with mobility impairments.

**Example:** A user can say "Start inspection" to initiate an inspection or "Open dashboard" to view the task dashboard.

#### Feature: Color Blindness Accessibility

**Purpose:** Provide options for color blindness accessibility, including color schemes that are friendly to users with different types of color vision deficiencies.

**Example:** Users can enable a color blindness mode in the settings, which adjusts the app’s color scheme to ensure that all visual indicators are distinguishable.

#### Feature: Multi-Factor Authentication (MFA)

**Purpose:** Enhance security while maintaining ease of use by offering multi-factor authentication (MFA) during login.

**Example:** Users are prompted to enter a verification code sent to their mobile device or email in addition to their password, adding an extra layer of security.

#### Feature: Session Timeout and Auto-Save

**Purpose:** Implement a session timeout feature to log out users after a period of inactivity, along with an auto-save feature to prevent data loss.

**Example:** If a user is inactive for 15 minutes, the session expires, and the user is logged out. However, any ongoing inspection or data entry is automatically saved.

## Administrative and Reporting Features

### User and Role Management

#### Feature: User Management Interface

**Purpose:** Allow administrators to add users, define roles, and assign tasks.

**Example:** The site safety manager assigns inspection tasks to specific employees based on their roles within the app.

#### Feature: Role-Based Dashboard Customization

**Purpose:** Allow administrators to customize dashboards based on user roles, ensuring that each user sees the most relevant information for their responsibilities.

**Example:** A safety manager’s dashboard might display compliance metrics and overdue inspections, while a technician’s dashboard focuses on upcoming tasks and recent tickets.

#### Feature: Customizable Role-Based Permissions

**Purpose:** Allow administrators to define and customize permissions for each role, providing granular control over what users can access and modify.

**Example:** A user with a "Viewer" role can view reports but cannot modify them, while a "Manager" role can both view and edit reports.

### Data and Reporting

#### Feature: Customizable Reporting

**Purpose:** Allow administrators to generate and customize reports based on specific criteria.

**Example:** The app generates a compliance report tailored to the specific needs of a regional audit.

#### Feature: Audit Trail and Reporting

**Purpose:** Track and review inspection activities through detailed logs and reports.

**Example:** Administrators can review a detailed audit trail of inspections, including timestamps and user actions.

#### Feature: Automated Report Scheduling

**Purpose:** Enable administrators to schedule automated reports that are generated and distributed at regular intervals (e.g., weekly, monthly) via email or within the app.

**Example:** A monthly compliance report is automatically generated and emailed to all site managers on the first of every month.

#### Feature: Data Export and Import Tools

**Purpose:** Provide tools for exporting and importing data in various formats, making it easier to integrate with other systems or transfer data between different environments.

**Example:** An administrator exports inspection data in CSV format for analysis in an external tool, or imports user data from another system during the initial setup.

#### Feature: Compliance Dashboard

**Purpose:** Offer a dedicated dashboard for monitoring compliance metrics, including inspection completion rates, overdue tasks, and audit readiness.

**Example:** A compliance officer uses the dashboard to track the percentage of inspections completed on time across all sites and identifies areas that need improvement.

#### Feature: Advanced Search and Filtering for Reports

**Purpose:** Provide advanced search and filtering options within the reporting module, allowing administrators to quickly locate specific reports or data points.

**Example:** An administrator filters reports by date range, site location, and inspection type to find a specific compliance report needed for an audit.

#### Feature: Version Control for Reports

**Purpose:** Implement version control for reports, allowing administrators to view and revert to previous versions of reports if needed.

**Example:** If a report is modified incorrectly, the administrator can revert to a previous version of the report that contains the correct data.

### Monitoring and Alerts

#### Feature: User Activity Monitoring

**Purpose:** Track and monitor certain user activity within the app to identify potential security risks or misuse.

**Example:** The system logs specific user activity, such as logins, and data modifications, allowing administrators to review these logs for suspicious behavior.

#### Feature: Custom Notification Rules

**Purpose:** Allow administrators to create custom notification rules that trigger alerts based on specific conditions, such as when certain thresholds are met or when critical tasks are overdue.

**Example:** The system sends an alert to the administrator when more than 10% of inspections at a particular site are overdue.

## Offline Capabilities and System Integration Features

### Offline Functionality

#### Feature: Offline Mode

**Purpose:** Allow the app to function without an internet connection, with data syncing when connectivity is restored.

**Example:** A safety coordinator performs inspections in a remote area with no connectivity; the data syncs automatically when they return online.

#### Feature: Data Encryption for Offline Storage

**Purpose:** Encrypt data stored locally on devices to ensure security.

**Example:** All offline data, including inspection records and user information, is encrypted to prevent unauthorized access.

#### Feature: Local Backup for Offline Data

**Purpose:** Provide a local backup option for data collected in offline mode, ensuring that data is not lost if the device experiences an issue before syncing.

**Example:** The app automatically creates a local backup of all inspection data collected offline, storing it on the device until it can be safely synced with the server.

### Data Synchronization

#### Feature: Seamless Data Synchronization

**Purpose:** Automatically sync data across devices and resolve conflicts upon reconnecting to the internet.

**Example:** After working offline, a user’s inspection data syncs with the server, resolving any data conflicts without user intervention.

#### Feature: Selective Data Sync

**Purpose:** Allow users to select specific data sets to sync when connectivity is restored, prioritizing critical information over less urgent data.

**Example:** After returning online, a user can choose to sync only the inspection records first while deferring the synchronization of other non-critical data.

#### Feature: Bandwidth Optimization for Syncing

**Purpose:** Optimize the data synchronization process to use minimal bandwidth, ensuring faster and more efficient syncing, especially in areas with slow internet connectivity.

**Example:** When syncing data in a low-bandwidth area, the app compresses the data before transmission, reducing the amount of data sent and speeding up the process.

#### Feature: Sync Status Dashboard

**Purpose:** Provide a dashboard or indicator that shows the status of data synchronization, including what data has been synced, what is pending, and any errors that need to be resolved.

**Example:** A user can view a dashboard that shows the progress of their data synchronization and receive notifications if any issues arise during the process.

#### Feature: Deferred Sync Option

**Purpose:** Allow users to defer data synchronization until a later time, giving them control over when large amounts of data are synced, particularly useful during periods of high network usage.

**Example:** A user defers syncing inspection data until after business hours when the network is less congested, ensuring a smoother and faster sync process.

#### Feature: Cross-Platform Data Consistency

**Purpose:** Ensure that data remains consistent across different platforms (e.g., mobile, tablet, desktop) during offline use and after synchronization.

**Example:** A user starts an inspection on a tablet in offline mode, and when they later access the data from a desktop, the app ensures that all changes made during offline use are accurately reflected.

#### Feature: Customizable Sync Schedules

**Purpose:** Allow administrators to set customizable synchronization schedules, defining when and how often data should be synced to the server.

**Example:** An administrator sets the app to sync data automatically every 30 minutes during business hours and manually during off-hours.

### System Integration

#### Feature: ERP/EHS System Integration

**Purpose:** Enable integration with existing Enterprise Resource Planning (ERP) or Environment, Health, and Safety (EHS) systems.

**Example:** Inspection data from the app is automatically synced with the company’s ERP system for comprehensive reporting and analysis.

#### Feature: API Integration for Third-Party Services

**Purpose:** Provide APIs to integrate with third-party services or applications, allowing for greater flexibility and customization in how the app interacts with other systems.

**Example:** A company uses the API to integrate the app with a third-party maintenance tracking system, allowing inspection data to be automatically forwarded for further processing.

## Security and Continuous Improvement Features

### Security Management

#### Feature: Role-Based Access Control (RBAC)

**Purpose:** Restrict access to specific features based on user roles.

**Example:** Administrators can set up RBAC to appropriately match the tasks and access for their users.

#### Feature: Multi-Factor Authentication (MFA)

**Purpose:** Enhance the security of user accounts by requiring multiple forms of verification during login.

**Example:** Users must enter a password and a verification code sent to their mobile device or email to access the app, providing an extra layer of security against unauthorized access.

#### Feature: Data Encryption in Transit and at Rest

**Purpose:** Ensure that all data is encrypted both when stored (at rest) and during transmission (in transit), protecting sensitive information from interception or unauthorized access.

**Example:** Inspection records and user data are encrypted before being stored in the database and are also encrypted when being transmitted between the app and the server.

#### Feature: Session Management and Timeout

**Purpose:** Implement session management features to automatically log out users after a period of inactivity, reducing the risk of unauthorized access.

**Example:** If a user is inactive for 15 minutes, they are automatically logged out of the app, with any unsaved data being securely stored.

#### Feature: Incident Response and Logging

**Purpose:** Provide mechanisms for logging security incidents and responding to potential threats or breaches, ensuring timely action can be taken.

**Example:** The app logs any unusual activity, such as multiple failed login attempts, and automatically notifies administrators to investigate and respond.

#### Feature: Automated Patch Management

**Purpose:** Implement automated patch management to ensure that the app is always up-to-date with the latest security fixes and improvements.

**Example:** The app automatically applies security patches and updates during off-hours, minimizing disruption to users while keeping the system secure.

#### Feature: Security Audits and Compliance Checks

**Purpose:** Regularly conduct security audits to ensure compliance with data protection regulations.

**Example:** The app undergoes quarterly security audits, verifying that all data handling practices meet industry standards.

### Continuous Improvement

#### Feature: Feedback Mechanism

**Purpose:** Allow users to report issues, suggest improvements, and provide feedback.

**Example:** Users can submit feedback directly through the app, which is then reviewed and prioritized for future updates.

#### Feature: Development Roadmap Integration

**Purpose:** Integrate user feedback into a development roadmap, showing planned updates and improvements.

**Example:** The development team updates the roadmap based on user feedback, which is then visible to users, fostering transparency and trust.

#### Feature: Change Log and Release Notes

**Purpose:** Maintain a detailed change log and release notes that document all updates, fixes, and new features, providing transparency to users.

**Example:** After each update, users can view the release notes within the app to see what changes have been made and how they might impact their workflow.

# Stack

## Frontend (Client-Side)

Alpine.js:

Purpose: Alpine.js will be used to enhance interactivity within the Django-generated forms and other elements. It allows for reactive UI components, form validation, dynamic content, and other interactive features without the overhead of a full JavaScript framework.

Relevant Features:

Real-Time Form Validation

Dynamic Form Fields (Adding/Removing)

Conditional Logic (Show/Hide Elements)

Multi-Step Forms and Form Wizards

Toggleable Content (e.g., modals, tooltips)

### Moment.js (or date-fns):

Purpose: Moment.js or date-fns will handle timestamping and date formatting, ensuring accurate tracking of when inspections and photo documentation occur.

#### Relevant Features:

Timestamp and/or Geotagging

### Bootstrap:

Purpose: Bootstrap will be utilized for responsive design, pre-built components, and utilities, making it easier to build a mobile-friendly UI.

#### Relevant Features:

Responsive Design

Adaptive Layouts for Varying Screen Sizes

### Chart.js:

Purpose: Chart.js will be used for creating interactive data visualizations and reports based on inspection data.

#### Relevant Features:

Reporting

Compliance Dashboard

### Axios:

Purpose: Axios will handle HTTP requests between the frontend and backend, simplifying the process of data fetching and submission.

#### Relevant Features:

Data Synchronization

API Integration for Third-Party Services

Annotorious (or Fabric.js):

Purpose: This library will enable photo annotation capabilities, allowing users to draw, label, and highlight areas of concern in photos directly within the browser.

Relevant Features:

Photo Annotation Tools

Leaflet.js (or Mapbox GL JS):

Purpose: Leaflet.js will be used to create interactive maps that allow administrators to drop pins for specific locations. Users can interact with these pins to view additional information or perform actions.

Relevant Features:

Interactive Map Integration

Location Documentation Interface

## Backend (Server-Side)

### Django:

Purpose: Django will serve as the primary backend framework, handling everything from user authentication to database management.

#### Relevant Features:

User Management Interface

Task and Scheduling Management

Ticketing System (including creation, assignment, and management)

### Django REST Framework (DRF):

Purpose: DRF will be used to create RESTful APIs, allowing seamless interaction between the frontend and backend.

#### Relevant Features:

API Integration for Third-Party Services

Data Export and Import Tools

### Django-ImageKit (or similar):

Purpose: This package will handle image processing and storage, including resizing, format conversion, and storing uploaded images.

#### Relevant Features:

Photo Documentation

### Django-Taggit:

Purpose: Django-Taggit can be used to manage tagging for photos and other content, making it easier to categorize and retrieve related data.

Relevant Features:

Photo Documentation with Annotation Tools

### Django Guardian:

Purpose: Django Guardian will be used to implement object-level permissions, allowing for granular control over what specific users can access or modify within the app. This will enable both model-level and object-level permissions, providing flexibility in how permissions are assigned and enforced.

Django-OTP

django-redis

Django-Axes: Tracks failed login attempts and enforces lockout after 3 consecutive failures.

Redis: Manages active session storage for fast retrieval and scalability.

Django-user-sessions: Tracks session history, logs session start and end times, and records activity.

Django Simple History: Logs any changes made to user data (e.g., profile, roles, preferences).

## Database

### PostgreSQL:

Purpose: PostgreSQL will be the database of choice due to its robustness, scalability, and full compatibility with Django.

#### Relevant Features:

Database Management System

Automated Backup and Recovery

External Database Integration

## Data Management and Reporting

### Django Admin:

Purpose: The Django Admin interface will be used for managing data and performing administrative tasks.

#### Relevant Features:

Customizable Reporting

Audit Trail and Reporting

## Authentication and Security

### Django's Built-in Authentication System:

Purpose: Django’s built-in authentication system will manage user login, logout, and password management.

#### Relevant Features:

Role-Based Access Control (RBAC)

Security Audits and Compliance Checks

## Deployment

### Nginx:

Purpose: Nginx will be used as the web server, efficiently serving static files and acting as a reverse proxy for the Django application.

#### Relevant Features:

Session Management and Timeout

Data Encryption in Transit and at Rest

### Gunicorn:

Purpose: Gunicorn will be used as the application server for running Django in a production environment.

#### Relevant Features:

Security Audits and Compliance Checks

Incident Response and Logging

### DigitalOcean:

Purpose: DigitalOcean provides a cost-effective platform for deploying Django applications, including managed PostgreSQL databases.

#### Relevant Features:

Automated Backup and Recovery

Development Roadmap Integration

## Development Workflow Tools

### Git/GitHub:

Purpose: Git will be used for version control, and GitHub for code hosting and collaboration.

#### Relevant Features:

Feedback Mechanism

Development Roadmap Integration