# **CIVIGUARD – Detailed Product Document**

## **Introduction**

CIVIGUARD is an innovative AI-powered civic assistant designed to facilitate the reporting of public issues such as potholes, garbage dumps, and water leaks. By allowing citizens to submit complaints through images and text, CIVIGUARD streamlines the grievance redressal process. The platform intelligently classifies complaints, predicts urgency, and auto-notifies local authorities, ultimately enhancing transparency and responsiveness in public governance.

## **Product Scope**

CIVIGUARD's scope encompasses the development of a Minimum Viable Product (MVP) that serves as a foundation for a full-scale civic engagement tool. The MVP will focus on key functionalities that allow for complaint submission, classification, urgency prediction, and notification, ensuring that citizens have a seamless experience while reporting issues.

## **Key Features**

* **Complaint Submission**: Users can report issues via image and text input, including location data.
* **AI Classification**: The system detects the type of issue based on submitted content using machine learning algorithms.
* **Urgency Prediction**: Issues are categorized based on urgency levels (Critical, Medium, Low) by analyzing sentiment and visual content.
* **Map Dashboard**: Visual representation of reported issues on a city map, with filtering options by category and urgency.
* **Auto Notification**: Automatic alerts sent to local authorities regarding reported issues.
* **Admin View**: A backend dashboard for authorities to manage and analyze incoming reports by various filters.

## **Modules or Components**

* **Frontend (React)**: User interface for complaint submission and viewing the map dashboard.
* **Backend (Node.js/Django)**: API for handling complaint submissions and interactions with the database.
* **Machine Learning (FastAPI)**: Model server for classifying images and text and predicting urgency.
* **Database**: Storage of complaints and user data (Firebase, MongoDB, or PostgreSQL).
* **Notification Service**: Integration for sending alerts via email or SMS.

## **User Roles or Flows**

* **Citizen**:

- Submits a complaint with an image, text description, and location. - Receives confirmation of submission and status updates.

* **Admin**:

- Accesses the admin dashboard to view, sort, and respond to complaints. - Uses filters to analyze data by type, urgency, and geographical area.

## **Technical Specifications**

* **Frontend**: Built with React and styled using Tailwind CSS.
* **Backend**: Node.js or Django for RESTful APIs.
* **Database**: Firebase, MongoDB, or PostgreSQL for data management.
* **Machine Learning**: FastAPI for serving models, using image classifiers like MobileNet or CNN and text classifiers such as TF-IDF + SVM or BERT.
* **Mapping**: Leaflet.js or Google Maps for integrating the map dashboard.

## **Functional Requirements**

* **Complaint Submission**: Users must be able to submit complaints with images and descriptions.
* **Classification**: The system should classify complaints based on input data.
* **Urgency Prediction**: The system must assess and classify the urgency of complaints.
* **Map Visualization**: Users should see a map with plotted complaints and filters.
* **Notification System**: The system should automatically notify authorities about new complaints.

## **Non-Functional Requirements**

* **Performance**: The system should handle at least 100 concurrent users without significant lag.
* **Usability**: The UI must be intuitive and accessible for users of all tech-savviness.
* **Security**: User data should be encrypted and secure, with proper authentication mechanisms in place.
* **Scalability**: The system must be designed to accommodate future growth and additional features.

## **Customizations**

* Allow for localized language support based on user location.
* Custom branding options for different municipalities or organizations.
* Adjustable settings for notification preferences by users and authorities.

## **Additional Features**

* **Heatmap Visualization**: A visual representation of issue density across different city areas.
* **WhatsApp Bot**: An auto-reply feature for complaints submitted via WhatsApp.
* **Historical Trend Graphs**: Analytics showing trends in complaints over time, segmented by area.

## **Appendices**

* **A. User Interface Mockups**: Initial design sketches for frontend components.
* **B. API Endpoints Documentation**: A list of all endpoints with their respective methods and expected inputs/outputs.
* **C. Machine Learning Model Details**: Information on the datasets used, model selection, and evaluation metrics.
* **D. Deployment Plan**: Steps for deploying the application to a live environment, including server setup and configuration.

This detailed product document serves as a comprehensive guide for the development and implementation of CIVIGUARD, ensuring clarity in vision, scope, and execution.