DIGITAL FRAUD AWARENESS & DETECTION PLATFORM

FRAUD SHIELD

v3.0 // COMPLETE PROJECT DOCUMENTATION

A production-grade web application for detecting digital fraud in SMS and WhatsApp messages using a 9-engine detection system.

9	4	35+	5
DETECTION ENGINES	LANGUAGES	FRAUD PATTERNS	APP ROUTES

PROJECT: FraudShield — Digital Fraud Awareness & Detection Platform
VERSION: v3.0 (Production Ready)
STACK: Python · Flask · scikit-learn · Tesseract · SQLite · ReportLab
DATE: 22 February 2026
USE: Educational & Awareness Purposes Only

01 // PROJECT OVERVIEW

INTRODUCTION & GOALS

01

Project Overview

FraudShield is a production-ready web application for detecting digital fraud in SMS and WhatsApp messages. It combines rule-based pattern matching with an AI/NLP classifier to deliver real-time threat assessments. The platform supports English, Hindi, Tamil, and Telugu, and offers voice input, screenshot OCR, URL inspection, explainable AI, PDF reporting, and a crowdsourced community scam feed.

Project Goals

GOAL	DESCRIPTION
`Detect fraud in real time`	Analyze SMS/WhatsApp messages using dual-engine detection and return a scored risk verdict.
`Educate users`	Provide scam awareness content covering six fraud types and six golden safety rules.
`Multilingual coverage`	Support Hindi, Tamil, and Telugu in addition to English — covering most Indian mobile users.
`Explainable AI`	Show which words drove the AI score using TF-IDF feature weight visualization.
`Production quality`	Generate PDF forensic reports, log all scans to SQLite, and serve a community threat feed.

■ FraudShield is for educational and awareness purposes only. It is not a substitute for professional cybersecurity tools or law enforcement. Report actual fraud to cybercrime.gov.in or call helpline 1930.

02 // SYSTEM ARCHITECTURE COMPONENTS & DATA FLOW

02

System Architecture

Project Folder Structure

```
fraudshield/
■■■ app.py Flask server, routing, result aggregation
■■■ rule_engine.py Regex/keyword patterns, phrase highlighter
■■■ nlp_model.py TF-IDF + Logistic Regression classifier
■■■ database.py SQLite logging, stats, recent records
■■■ ocr_scanner.py Tesseract OCR with image preprocessing
■■■ pdf_report.py ReportLab forensic PDF report generator
■■■ url_inspector.py Heuristic URL analysis, typosquatting detection
■■■ multilingual.py Hindi, Tamil, Telugu fraud pattern matching
■■■ explainability.py TF-IDF feature importance, AI explanation
■■■ community_feed.py Anonymized scam feed, crowdsourced intelligence
■■■ requirements.txt Python dependencies
■■■ static/
■ ■■■ style.css Forensic terminal UI, amber palette
■■■ templates/
■■■ index.html Main analyzer — 3 input modes
■■■ dashboard.html Live threat dashboard with Chart.js
■■■ logs.html Analysis history table
■■■ community.html Crowdsourced threat feed
```

Detection Pipeline

Every submitted message passes through a structured 9-stage pipeline:

STAGE	MODULE	METHOD	OUTPUT
1. Input	app.py	Text / Voice / OCR	Raw message string
2. Rule Engine	rule_engine.py	35+ regex, weighted scoring	rule_score + flags
3. Multilingual	multilingual.py	Hindi / Tamil / Telugu patterns	multi_score + regional flags
4. AI Classifier	nlp_model.py	TF-IDF + Logistic Regression	ai_score (0–100)
5. URL Inspector	url_inspector.py	10 heuristic checks per URL	URL risk report
6. Score Fusion	app.py	0.6 x rule + 0.4 x AI	final_score, risk_level
7. Explainability	explainability.py	Feature weight extraction	Top contributing words
8. Storage	database.py	SQLite INSERT	Persisted log record
9. Output	index.html	Jinja2 render + PDF option	Full result UI + download

Scoring Formula

```
combined_rule = min(100, rule_score + multilingual_bonus)

final_score = round(0.6 x combined_rule + 0.4 x ai_score)

Risk levels:
0 to 30 → LOW (message appears safe)
31 to 70 → MEDIUM (exercise caution)
71 to 100 → HIGH (likely fraudulent)
```

03 // FEATURE DOCUMENTATION ALL 9 DETECTION & UI FEATURES

03

Feature Documentation

01 Voice Input

Web Speech API (browser-native)

Users speak a suspicious message aloud. The browser transcribes it in real time using the Web Speech API, then submits the transcript for analysis. No server processing needed.

- Browser-native — no server processing or API keys required
- Locale: en-IN for Indian English accent optimization
- Continuous recording with live interim results displayed
- Transcript submitted via hidden form POST on confirm click

02 Screenshot OCR

ocr_scanner.py + Tesseract 4.x

Upload a WhatsApp/SMS screenshot. The image is preprocessed (contrast, sharpness, upscaling) then scanned by Tesseract OCR. Supports drag-and-drop with live preview.

- Preprocessing: upscale small images, boost contrast and sharpness
- Multi-language OCR: English + Hindi + Tamil + Telugu
- Multiple PSM configs tried — best result auto-selected
- Drag-and-drop zone with image preview before submit

03 Live Threat Dashboard

dashboard.html + Chart.js 4.4

Analytics page with a risk distribution donut chart, top fraud signals bar chart, five KPI stat cards, and a live activity feed. Refreshes via /api/stats every 15s.

- Donut chart: HIGH / MEDIUM / LOW risk distribution
- Horizontal bar chart: top detected fraud keywords by frequency
- Five KPI cards: total scans, risk counts, average score
- Live activity feed with timestamps and risk badges

04 Multilingual Detection

`multilingual.py`

Detects fraud patterns in Hindi (Devanagari), Tamil, Telugu, and Hinglish. Adds a bonus score (capped at 50) to the combined rule score.

- Hindi: 19 native Devanagari script patterns
- Tamil: 15 native Tamil script patterns
- Telugu: 15 native Telugu script patterns
- Hinglish: 14 romanized transliteration patterns

05 URL Deep Inspector

`url_inspector.py`

Extracts and analyzes all URLs in a message using 10 heuristic checks. No external API needed. Results show domain, risk score, and a plain-language finding list.

- Whitelist of 30+ verified Indian domains (SBI, HDFC, IRCTC, UIDAI, etc.)
- Suspicious TLD detection: .xyz, .tk, .ml, .pw and 15+ others
- Brand impersonation and typosquatting via hyphen analysis
- Raw IP, URL shortener, and suspicious path keyword detection

06 Data Storage & Logging

`database.py + SQLite`

Every scan is persisted to a local SQLite database. Records include all scores, risk level, detected flags, and timestamp. Auto-initialised on first app start.

- Zero-config: table created automatically on startup
- Stores: message, rule_score, ai_score, final_score, risk_level, flags, timestamp
- get_stats() returns aggregate totals and average score
- get_recent_logs(n) fetches the last N records for dashboard

07 PDF Forensic Report

`pdf_report.py + ReportLab`

After analysis, users download a professionally formatted PDF with a unique report ID, all scores, detected signals, the original message, and safety recommendations.

- Five sections: Verdict, Score Breakdown, Signals, Transcript, Safety Tips
- Unique report ID generated from message hash and date
- Custom dark cover with amber accents
- Downloaded via POST to /download-report route

08 Explainable AI Panel

`explainability.py`

Shows which words drove the AI score by extracting TF-IDF feature weights from the Logistic Regression model. Each token is rated fraud-increasing or safe with a bar.

- Extracts non-zero features from TF-IDF sparse matrix per message
- Score = TF-IDF value x LR coefficient, normalised to 0–100
- Color-coded: red = fraud-increasing, green = safe tokens
- Auto-generated summary sentence from top fraud and safe words

09 Community Scam Feed

`community_feed.py`

HIGH and MEDIUM risk messages are anonymized and shared as public threat intelligence on /community. All PII is automatically redacted before display.

- Redacts: phone numbers, emails, Aadhaar, account numbers, full URLs
- URL anonymized to [LINK: domain.com] preserving domain for awareness
- Only HIGH and MEDIUM risk messages appear in the feed
- Trending fraud signals sidebar with detection frequency counts

04 // BACKEND API REFERENCE FLASK ROUTES & MODULE FUNCTIONS

04

Backend API Reference

Flask Routes

ROUTE	METHOD	DESCRIPTION	TEMPLATE
/	GET + POST	Main analyzer — text, voice, and OCR input	index.html
/logs	GET	Analysis history table with aggregate stats	logs.html
/dashboard	GET	Live threat dashboard with Chart.js charts	dashboard.html
/community	GET	Anonymized community scam feed	community.html
/download-report	POST	Generate and stream PDF forensic report	— binary
/api/stats	GET	JSON: stats + top_flags + recent logs	— JSON

Key Module Functions

rule_engine.py

FUNCTION	SIGNATURE	DESCRIPTION
analyze_message(msg)	str → (int, list)	Returns rule_score and detected phrase list
highlight_message(msg, phrases)	str, list → str	Wraps phrases in HTML mark tags

nlp_model.py

FUNCTION	SIGNATURE	DESCRIPTION
get_ai_score(message)	str → int	Returns fraud probability 0–100
clean_text(text)	str → str	Lowercase, URL replace, strip non-alpha

database.py

FUNCTION	SIGNATURE	DESCRIPTION
init_db()	→ None	Creates table on first run
log_analysis(msg, r, a, f, risk, flags)	→ None	Inserts one analysis record
get_recent_logs(limit)	int → list	Last N records by ID descending
get_stats()	→ dict	Total, HIGH/MED/LOW counts, average score

ocr_scanner.py

FUNCTION	SIGNATURE	DESCRIPTION

`extract_text_from_image(bytes)`	`bytes → (str, bool, str)`	Multi-config Tesseract OCR
`preprocess_image(bytes)`	`bytes → Image`	Upscale, contrast, sharpen

url_inspector.py

FUNCTION	SIGNATURE	DESCRIPTION
`inspect_urls_in_message(msg)`	`str → list[dict]`	Extract + analyze all URLs
`analyze_url(url)`	`str → dict`	10 heuristic checks, risk_score + findings

multilingual.py

FUNCTION	SIGNATURE	DESCRIPTION
`analyze_multilingual(msg)`	`str → (int, list)`	HI/TA/TE/Hinglish pattern check

explainability.py

FUNCTION	SIGNATURE	DESCRIPTION
`get_top_features(msg, pipeline)`	`str, Pipeline → list`	TF-IDF weighted feature extraction
`get_ai_explanation(msg, pipeline)`	`str, Pipeline → dict`	Features + summary sentence

community_feed.py

FUNCTION	SIGNATURE	DESCRIPTION
`get_community_feed(limit)`	`int → list[dict]`	Anonymized HIGH/MEDIUM feed records
`anonymize_message(msg)`	`str → str`	Redact phones, emails, IDs, full URLs
`get_top_flags(limit)`	`int → list[tuple]`	Most frequent flags with counts

05 // SETUP & DEPLOYMENT PREREQUISITES · LOCAL RUN · RENDER DEPLOY

05

Setup & Deployment

System Prerequisites

COMPONENT	VERSION	NOTES
Python	3.8+	Required for all backend modules
pip	Latest	For installing Python packages
Tesseract OCR	4.x	Required for Feature 2 (Screenshot OCR)
tesseract-ocr-hin	4.x	Hindi language pack
tesseract-ocr-tam	4.x	Tamil language pack
tesseract-ocr-tel	4.x	Telugu language pack
Modern browser	Chrome / Edge / Safari	Required for Voice Input (Web Speech API)

Python Dependencies (requirements.txt)

```
Flask # Web framework and routing
scikit-learn # TF-IDF vectorizer + Logistic Regression classifier
numpy # Numerical operations for the ML pipeline
gunicorn # Production WSGI server for cloud deployment
pytesseract # Python wrapper for Tesseract OCR engine
Pillow # Image preprocessing for OCR (PIL)
reportlab # PDF generation for forensic reports
```

Local Installation Steps

```
# Step 1 — Extract the project ZIP
cd fraudshield

# Step 2 — Create and activate virtual environment
python -m venv venv
source venv/bin/activate # macOS / Linux
venv\Scripts\activate # Windows

# Step 3 — Install Python packages
pip install -r requirements.txt

# Step 4 — Install Tesseract OCR (Ubuntu / Debian)
sudo apt install tesseract-ocr tesseract-ocr-hin tesseract-ocr-tam tesseract-ocr-tel

# Step 4 — Install Tesseract OCR (macOS Homebrew)
brew install tesseract tesseract-lang

# Step 5 — Start the application
python app.py

# Open in browser: http://127.0.0.1:5000
```

Render Cloud Deployment

1. Push the project folder to a GitHub repository.
2. Log in to render.com and click New Web Service.
3. Connect your GitHub repository and select the fraudshield branch.
4. Set Environment to Python 3.
5. Set Build Command: pip install -r requirements.txt
6. Set Start Command: gunicorn app:app
7. Add pre-install for Tesseract in the shell or a render.yaml build step.
8. Click Deploy — Render builds and hosts automatically.

> The SQLite database (logs.db) is stored on the server filesystem. On Render free tier, the filesystem resets on each deploy. For persistent logs in production, migrate to PostgreSQL using SQLAlchemy and Render's managed database service.

06 // ML MODEL DETAILS NLP CLASSIFIER & TRAINING DATA

06

ML Model Details

Model Architecture

The NLP classifier uses a scikit-learn Pipeline. It is trained at application startup on 46 labeled messages — no external model files required.

COMPONENT	CONFIGURATION	PURPOSE
TfidfVectorizer	ngram_range=(1,2), max_features=600	Convert text to weighted numerical features
LogisticRegression	max_iter=1000, C=1.5	Binary classifier: fraud (1) vs genuine (0)
Pipeline	Sequential: TF-IDF then LR	Single fit/predict interface for both steps
Training set	46 messages (24 fraud, 22 genuine)	Built-in dataset — no external files needed
Prediction	predict_proba()[:,1] x 100	Fraud probability as integer 0 to 100

Training Dataset Samples

LABEL	SAMPLE MESSAGE
FRAUD	Your bank account has been blocked. Update KYC immediately: http://scam.link
FRAUD	Congratulations! You won a lottery prize of Rs 50,000. Claim now.
FRAUD	URGENT: Your Aadhaar account will be suspended. Verify OTP immediately.
FRAUD	Win a free iPhone! Enter your PAN and Aadhaar at: http://win.fake.in
GENUINE	Your OTP for Zomato login is 847291. Valid for 5 minutes. Do not share.
GENUINE	Your salary of Rs 45,000 has been credited to your account.
GENUINE	Reminder: Your appointment with Dr. Sharma is at 11am on Monday.
GENUINE	Your Jio recharge of Rs 239 is successful. Validity: 28 days.

Rule Engine Pattern Weights

Patterns are matched with regex and scored using the table below. The combined score is capped at 100.

PATTERN	WT	PATTERN	WT
suspicious link	15	OTP	10
KYC	10	Aadhaar	10
PAN	10	password	10
credit card	10	debit card	10

PIN	10	account number	10
click here	8	update account	8
unfreeze	7	suspend	7
deactivate	7	act now	7
blocked	7	lottery	10
urgent	6	immediately	6
prize	8	won	8
verify	5	claim	5

07 // UI DESIGN SYSTEM TYPOGRAPHY · COLOURS · VISUAL EFFECTS

07

UI Design System

The interface follows a forensic terminal aesthetic — classified intelligence dashboard meets cyberpunk noir. Near-black base, single amber accent, scan-line texture, noise overlay, and crosshair cursor throughout.

Typography

FONT FAMILY	USAGE	WEIGHTS
Bebas Neue	Display headings, score numbers, page titles, risk levels	Regular (400)
IBM Plex Mono	Data values, scores, labels, code snippets, tags	300 / 400 / 500
Barlow	Body text, descriptions, paragraph content	300 / 400 / 500

Colour Palette

CSS VARIABLE	HEX	ROLE	USAGE
--bg	#0a0a08	Primary background	Page, message display, code blocks
--amber	#f5a623	Primary accent	Buttons, borders, highlights, marks
--high	#ff3d5a	HIGH risk	HIGH badges, signal chips, borders
--medium	#ff8c00	MEDIUM risk	MEDIUM badges and indicators
--low	#00c896	LOW / safe	LOW badges, status dots, safe signals
--text	#f0ede6	Primary text	Headings and high-emphasis content
--text-muted	#555550	Secondary text	Labels, captions, metadata

Visual Effects

EFFECT	DESCRIPTION
Scan-line overlay	Repeating 4px CSS gradient simulating CRT monitor scan-lines. Applied as a fixed div.
Noise texture	SVG feTurbulence filter as base64 background-image for analog film-grain effect.
Ticker bar	Infinite CSS translateX animation scrolling system status text across the top.
Corner brackets	Pure CSS absolute-positioned elements on the textarea providing a targeting reticle look.
Score bar animation	JS sets bar widths to 0% on load, then transitions to target values after 300ms.
Signal chip stagger	Each chip gets animation-delay = index x 60ms for a sequential reveal effect.
Crosshair cursor	cursor: crosshair applied globally reinforcing the forensic / targeting aesthetic.

08 // SCAM AWARENESS GUIDE FRAUD TYPES & SAFETY RULES

08

Scam Awareness Guide

FraudShield includes a dedicated educational section covering the six most common digital fraud types targeting Indian mobile users, with defence advice for each.

OTP / KYC Scams — HIGH

Fraudsters impersonate banks claiming your account will be blocked unless you share your OTP or update KYC immediately. Most common digital fraud in India.

Defence: Banks NEVER ask for OTPs via SMS. Hang up and call the official bank number.

Lottery & Prize Scams — HIGH

Messages claim you won a prize. To claim, you must pay a fee or share personal details. There is no prize — the fee is the scam.

Defence: You cannot win a lottery you never entered. Fees to claim prizes = always a scam.

Phishing Links — HIGH

URLs mimicking official sites (sbi-secure.in vs sbi.co.in) lead to fake login pages designed to steal credentials. FraudShield's URL Inspector catches these.

Defence: Check every character. Official banks use .co.in or .com — never random hyphens or TLDs.

Card / Account Freeze — MEDIUM

Scammers claim your card is frozen and request card number, PIN, or CVV to unfreeze. Sharing these gives full access to your account.

Defence: No institution will ever ask for your PIN or CVV. These must never be shared.

Aadhaar / PAN Scams — HIGH

Messages claim your Aadhaar or PAN is expired or linked to illegal activity, urging you to share ID details immediately.

Defence: UIDAI and IT Dept never request ID via SMS. Visit their official portals directly.

Vernacular Language Scams HIGH

Scammers increasingly operate in Hindi, Tamil, and Telugu to target regional users less familiar with digital fraud tactics in their native language.

Defence: Fraud tactics are identical across all languages. Urgency + personal data = danger.

Golden Safety Rules

NO.	SAFETY RULE
01	Never share OTP, PIN, password, or CVV — not even with bank employees.
02	Call back on official numbers from the bank website, not from the SMS.
03	Urgency is a weapon. Pause, verify, and never act under pressure.
04	Check URLs character by character. Fakes use subtle typos like "sbi-secure.in".
05	Report fraud to cybercrime.gov.in or call national helpline 1930 immediately.
06	If scammed, call your bank immediately and request an account freeze.

09 // REQUIREMENTS CHECKLIST FULL COMPLIANCE VERIFICATION

09

Requirements Checklist

All requirements from the original project brief and all nine recommended add-ons are fully implemented and tested in FraudShield v3.0.

Original Project Requirements

REQUIREMENT	STATUS	IMPLEMENTATION
UI for message input and result display	DONE	`index.html — 3 input modes: text, voice, OCR`
Rule-based fraud detection engine	DONE	`rule_engine.py — 35+ regex patterns, weighted scoring`
Workflow manager combining rule + AI outputs	DONE	`app.py — full_analysis() — 0.6 x rule + 0.4 x AI`
NLP-based AI scam probability model	DONE	`nlp_model.py — TF-IDF + Logistic Regression`
Phrase-highlighting mechanism	DONE	`rule_engine.highlight_message() — mark tags`
Data storage / message logging	DONE	`database.py — SQLite with auto-init schema`
Educational scam awareness content	DONE	`index.html awareness section + community.html`

Nine Recommended Add-ons

FEATURE	STATUS	MODULE
01 — Voice Input (Web Speech API)	DONE	`index.html JS`
02 — Screenshot OCR Scanner	DONE	`ocr_scanner.py`
03 — Live Threat Dashboard + Chart.js	DONE	`dashboard.html`
04 — Multilingual Detection (HI / TA / TE)	DONE	`multilingual.py`
05 — URL Deep Inspector	DONE	`url_inspector.py`
06 — SQLite Data Storage and Logging	DONE	`database.py`
07 — PDF Forensic Report Generator	DONE	`pdf_report.py`
08 — Explainable AI Feature Panel	DONE	`explainability.py`
09 — Community Scam Feed (anonymized)	DONE	`community_feed.py`

✓ All 16 requirements — 7 original and 9 add-ons — are fully implemented, tested end-to-end, and packaged in fraudshield_v3_FINAL.zip.

Quick-Start Commands

```
# Install all dependencies
pip install -r requirements.txt

# Install Tesseract OCR language packs (Ubuntu)
sudo apt install tesseract-ocr tesseract-ocr-hin tesseract-ocr-tam tesseract-ocr-tel

# Launch the application
python app.py

# Open in browser
# http://127.0.0.1:5000 Main Analyzer
# http://127.0.0.1:5000/dashboard Live Dashboard
# http://127.0.0.1:5000/community Community Feed
# http://127.0.0.1:5000/logs Analysis History
# http://127.0.0.1:5000/api/stats JSON Stats API

# National Cybercrime Helpline: 1930
# Report online: cybercrime.gov.in
```