Symptom-Driven Disease Prediction and Advisory Platform

Submitted By
Mr. Adnan Hashmi (250810125008)
Mr. Daksh Khosla (250810125006)
Mr. Priyanshu Singh (250810125010

Guided By
Prof. Jaibheem Gaikwad

Faculty, CDAC Delhi

How to Cite this Article:

Hashmi, A., Khosla, D. & Singh, P. (2026). Symptom-Driven Disease Prediction and Advisory Platform. International Journal of Creative and Open Research in Engineering and Management, 02(6). https://doi.org/10.55041/ijcope.v2i6.145

https://doi.org/10.55041/ijcope.v2i6.145

1. Abstract

This project presents a compact, maintainable, and explainable system that maps self-reported symptoms to likely disease labels and returns curated, non-prescriptive guidance to the user. The system uses a deterministic symptom indexing approach to convert free-text symptom inputs into fixed-length binary feature vectors.

A multi-class Support Vector Classifier (SVC) with a linear kernel is trained on a curated dataset of symptom-disease mappings and serialized for fast inference. Auxiliary content (disease descriptions, precautions, educational medication pointers, diet and activity advice) is stored in editable CSVs to allow healthcare experts to update guidance without modifying code.

The backend is implemented in Flask; the frontend is responsive and optimized for clarity and safety messaging. Model evaluation shows high aggregate performance on the available test split, but the report emphasizes responsible usage, limitations, and recommended next steps for clinical validation.

2. Introduction

2.1 Problem Statement

Millions of individuals around the world rely on symptom information to make decisions about self-care and seeking medical attention. Symptom checkers and decision-support tools can provide initial triage guidance but vary widely in accuracy, transparency and regional suitability. The core problem this project addresses is:

How to build an auditable, low-footprint system that translates user-reported symptoms into likely disease classes and returns clear, educational guidance while emphasizing safety and non-diagnostic usage.

Key challenges:

- Noisy natural language inputs (misspellings, synonyms).
- Large multi-class label space with possible class imbalance.
- Need for explainability and clinician-safe messaging.
- Lightweight deployment target for limited-resource environments.

2.2 Objectives

Primary Objectives:

- Implement a deterministic symptom mapping to a fixed feature vector dimension.
- Train a multi-class classifier (SVC baseline) and evaluate using standard classification metrics (accuracy, precision, recall, F1).
- Provide human-readable guidance for predicted classes stored in editable CSVs.
- Deploy a prototype Flask service with a responsive UI for demonstration.

Secondary Objectives:

- Include explainability features (top contributing symptoms) for increased transparency.
- Provide a maintainable content management approach (CSV + admin upload functionality).
- Provide full developer documentation and a reproducible training pipeline.

2.3 Scope and Limitations

In Scope	Out of Scope
• Dataset curation and preprocessing • Model training and evaluation • Flask backend and content CSVs	• Clinical validation as a regulated medical device • Large-scale production deployment
• Basic admin functionality • Responsive frontend, documentation and testing	• HIPAA/GDPR compliance beyond baseline anonymization • Complex multi-modal inputs (images, lab values) • Automated prescriptions

2.4 Stakeholders and Use Cases

Primary Stakeholders:

- End users seeking quick triage guidance.
- Clinicians who may use the tool for early triage suggestions (with caution).
- Project evaluators and educators at the institute.

Representative Use Cases:

- Lay user with non-urgent symptoms wishes to get initial guidance and decide whether to see a clinician.
- Clinic operator wants a lightweight triage assistant for patient intake in low-resource settings (as a pilot).
- Educator wants to demonstrate practical ML & deployment for student learning.

3. System Analysis

3.1 Background and Literature Review

3.1.1 Symptom Checkers & Digital Triage — State of the Art

Symptom checkers have existed in various forms: rule-based systems, decision trees, Bayesian networks, and more recently, machine learning models using text-based symptom inputs. Rule-based systems are interpretable but brittle; machine learning offers improved flexibility but requires careful training data curation and interpretability safeguards. Recent research emphasizes hybrid approaches: combining clinical rules with ML predictions and incorporating confidence thresholds and clinician oversight for safety-critical decisions.

3.1.2 Machine Learning for Multi-Class Clinical Classification

Common approaches include:

- Linear models (Logistic Regression, Linear SVC) — fast, interpretable via coefficients, effective for high-dimensional sparse features.
- Tree-based ensembles (Random Forest, XGBoost) — strong performance and partial interpretability (feature importances).
- Neural approaches (deep learning) — useful for large datasets and complex feature relationships but need more data and are less interpretable.

Given the problem's nature (sparse binary symptom vectors, moderate dataset size, need for interpretability), a linear SVC was chosen as the primary model.

3.1.3 Explainability and Safety in Medical AI

Explainability is essential for user trust and regulatory scrutiny. For linear models, coefficients can be used to identify symptoms positively associated with a predicted class; for more complex models, methods like SHAP provide per-prediction explanations. Safety strategies include: conservative confidence thresholds, educational wording for medication advice, and explicit disclaimers encouraging clinical consultation.

3.2 System Overview & Requirements

3.2.1 Functional Requirements

1. Symptom Input Methods — Free-text comma separated input with normalization; optional checkbox list of common symptoms.
2. Prediction Endpoint — POST /predict accepts JSON: {"symptoms":"fever,cough"} or form data. Returns JSON including predictions with label, confidence, description, precautions, medications, diet, and activity.
3. Content Management — Admin route to upload CSVs and validate schema; versioning metadata displayed for each CSV.
4. Model Metadata — Endpoint /model-info returning training date, version, training metrics, and class list.
5. User Interface — Responsive input page, results page with prominent safety notice, and about page with methodology summary.
6. Logging and Audit — Anonymized logs capturing non-identifiable input features and inference times for debugging and model improvement.

3.2.2 Non-Functional Requirements

Requirement	Description
Performance	95% of single inference requests must complete under 500 ms on a demo VM.
Scalability	Architecture must support containerization and horizontal scaling for production.
Reliability	Server must fail gracefully; unknown symptoms produce informative messages.
Security	Input sanitization, validation of uploaded CSVs, HTTPS required in deployment.
Maintainability	Clear modular code, unit tests, CI/CD pipelines recommended.

3.3 Dataset and Preprocessing

3.3.1 Data Sources

The dataset used is a curated symptom-disease mapping dataset prepared for the project. Disease labels and associated symptom lists were consolidated from multiple domain resources and manually reviewed for consistency. Data used for training, validation, and testing follows standard ML practices; exact dataset provenance and anonymization steps are documented in internal project logs.

3.3.2 Data Cleaning and Normalization

Key cleaning steps applied:

- Lowercasing — all text standardized to lowercase for consistent comparisons.
- Removing punctuation and excessive whitespace.
- Synonym mapping — maintaining a small synonym dictionary (e.g., "high temp" → "fever").
- Spell correction heuristics — for common misspellings, a whitelist correction is applied.
- Unknown symptom handling — symptoms not in the master symptom_index are logged and ignored; the UI informs the user which inputs could not be recognized.

3.3.3 Feature Engineering and Symptom Indexing

The central feature engineering approach uses a deterministic **symptom index** mapping: a list of N known symptoms (N = 132 in our training runs). Each symptom is assigned a stable index; inputs set a binary 0/1 flag for each symptom's presence. This representation is compact, interpretable, and suitable for linear models.

3.3.4 Train/Test Split and Class Imbalance Handling

- Stratified split preserving per-class proportions (70/30 train/test) for representative evaluation.
- SVC implemented with class_weight='balanced' to penalize errors for minority classes.
- SMOTE oversampling explored but not used in the baseline to avoid introducing synthetic artifacts.

3.3.5 Model Selection

Models evaluated:

- Logistic Regression (one-vs-rest) — interpretable and fast.
- Linear SVC (one-vs-rest) — chosen for robust linear separation and stable coefficients for explainability.
- Random Forest / Gradient Boosting — strong non-linear models; useful as future ensemble members.
- Multilayer Perceptron — considered but requires more data and hyperparameter tuning.

3.3.6 Support Vector Classifier — Configuration

Parameter	Value / Description
Classifier	sklearn.svm.LinearSVC (one-vs-rest)
Regularization (C)	Tuned via grid search: 0.01, 0.1, 1.0, 10
Class Weight	balanced (mitigate class imbalance)
Tolerance	1e-4
Max Iterations	10000 (ensure convergence)
Standardization	Not required (binary features)

3.3.7 Training Pipeline

7. Load cleaned dataset and master symptom index.
8. Vectorize symptom lists into binary feature arrays of shape (n_samples, n_symptoms).
9. Apply stratified train/test split.
10. Grid search (cross-validation) for C and other hyperparameters, scoring on weighted F1.
11. Final model trained on full training set with chosen hyperparameters.
12. Evaluate on the held-out test set to produce final metrics.

4. System Design and Architecture

4.1 High-Level Architecture and Workflow

System Components:

- Frontend (Client) — HTML/CSS/JS UI (Bootstrap/Jinja2 templates) for input and results.

- Backend (Flask) — loads model and CSV content at startup; exposes endpoints: /, /predict, /model-info, /admin/upload.
- Data Storage — data/ folder with CSVs for disease descriptions and content.
- Model Artifact — model/svc.pkl with model_metadata.json.
- Logging — rotating logs for errors and anonymized inference data.
- Optional Reverse Proxy — nginx for static assets and HTTPS termination.

Workflow: User submits symptoms → Flask backend receives request → SymptomMapper normalizes and vectorizes input → ModelService predicts top labels and confidences → ContentStore assembles human-readable guidance → Response returned and rendered.

4.2 Component Descriptions

Component	Description
SymptomMapper	Text normalization, synonym mapping, and vectorization of user input into binary feature vectors.
ModelService	Model loading, prediction, and metadata management. Handles inference and confidence scoring.
ContentStore	CSV parsing and caching, retrieval by disease label, schema validation.
API Layer (Flask)	Input validation, request orchestration, optional rate limiting, and response formatting.
Admin Module	Simple CSV upload with preview and rollback capability for content management.
Monitoring	Response times and error rates for demo; Prometheus/Grafana recommended for production.

4.3 API Contracts and Data Flow

POST /predict

Request: {"symptoms": "fever,cough,headache"} (JSON or form-encoded)

Response JSON includes: predictions (disease + confidence), description, precautions, medications (educational), diet, activity, and explanation (top contributing symptoms).

GET /model-info

Response: metadata object with model version, training date, metrics, and class list.

POST /admin/upload

Upload CSV file; server validates headers and sample rows. If valid, replaces CSV and records update metadata. Rollback capability is provided.

4.4 Security and Privacy Measures

- No PII — by design, the service does not request or store personally identifiable information; logs are anonymized and truncated.
- Input Validation — all inputs sanitized to prevent injection attacks; CSV admin uploads validated for expected headers and content schema.
- HTTPS — enforced in deployment for all production environments.
- Access Control — admin routes protected by authentication (basic token or OAuth for production).

- Data Retention Policy — logs retained only for the minimum required period; mechanism provided to clear logs.

4.5 User Interface & User Experience

Frontend Design Principles

- Clarity — large, readable fonts; simple language; avoidance of medical jargon where possible.
- Safety First — persistent disclaimer: "This tool provides educational information and is not a clinical diagnosis."
- Accessibility — semantic HTML, ARIA attributes for screen readers, contrast ratio meeting WCAG AA standards.
- Responsiveness — page layouts adapt to mobile and desktop to accommodate a wide user base.

4.6 Evaluation Metrics

Metric	Value	Notes
Accuracy	0.9627	*96.27%*
Weighted F1 Score	0.9545	—
Training Samples	3,444	—
Testing Samples	1,476	—
Number of Features	132	*Symptom dimensions*
Number of Classes	41	*Disease labels*

4.7 Model Performance Charts

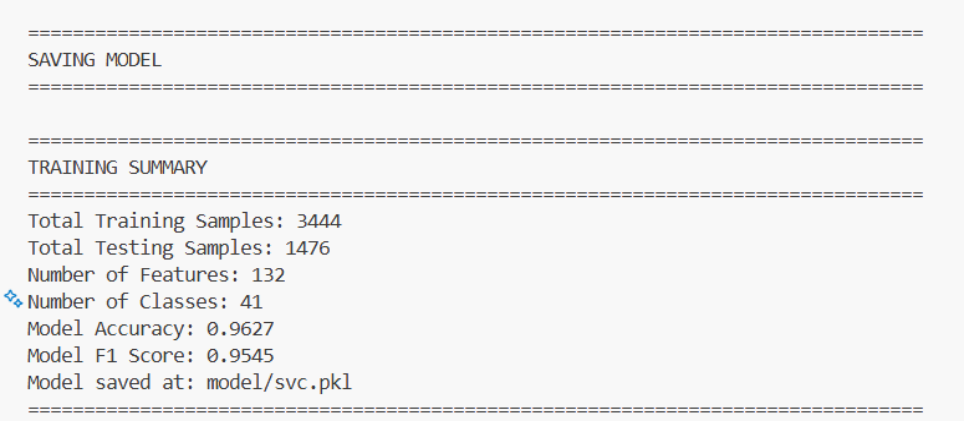
```
================================================================================
TRAINING SVC MODEL
================================================================================
Training SVC with linear kernel...
[OK] Model training completed!

Generating predictions on test set...
[OK] Predictions completed!

================================================================================
MODEL EVALUATION METRICS
================================================================================

Accuracy: 0.9627 (96.27%)
F1 Score (Weighted): 0.9545
```

Figure 4.1 — Model Performance Overview



```
================================================================================
SAVING MODEL
================================================================================

================================================================================
TRAINING SUMMARY
================================================================================
Total Training Samples: 3444
Total Testing Samples: 1476
Number of Features: 132
Number of Classes: 41
Model Accuracy: 0.9627
Model F1 Score: 0.9545
Model saved at: model/svc.pkl
================================================================================
```

Figure 4.2 — Evaluation Metrics Distribution

4.8 Application Screenshots

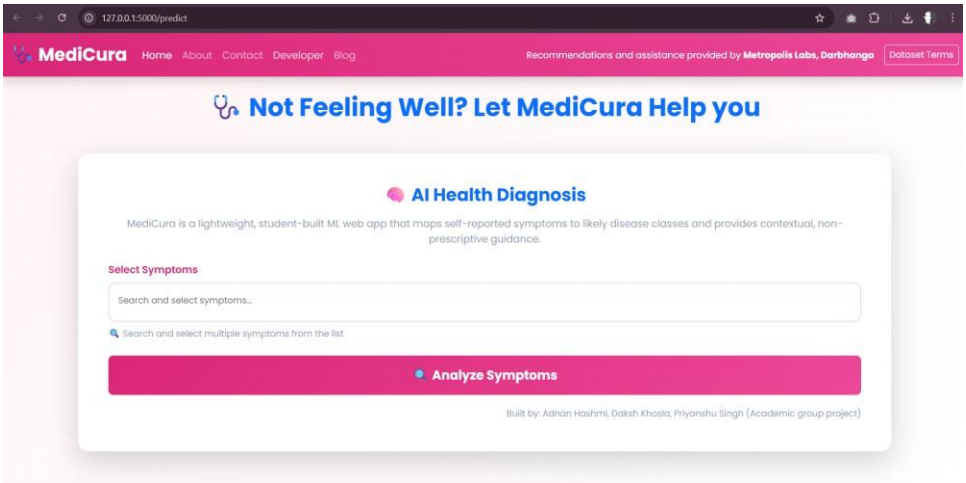


Figure 4.3 — Home / Symptom Input Screen

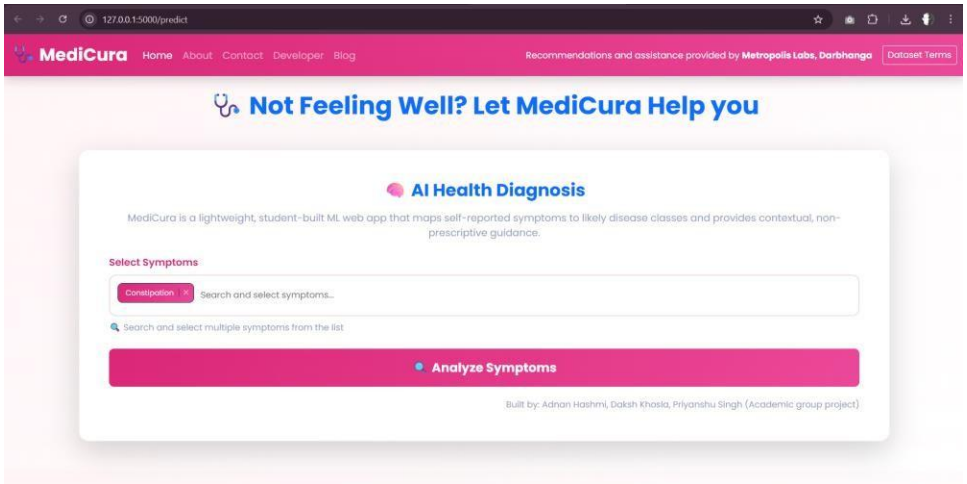


Figure 4.4 — Prediction Results Screen

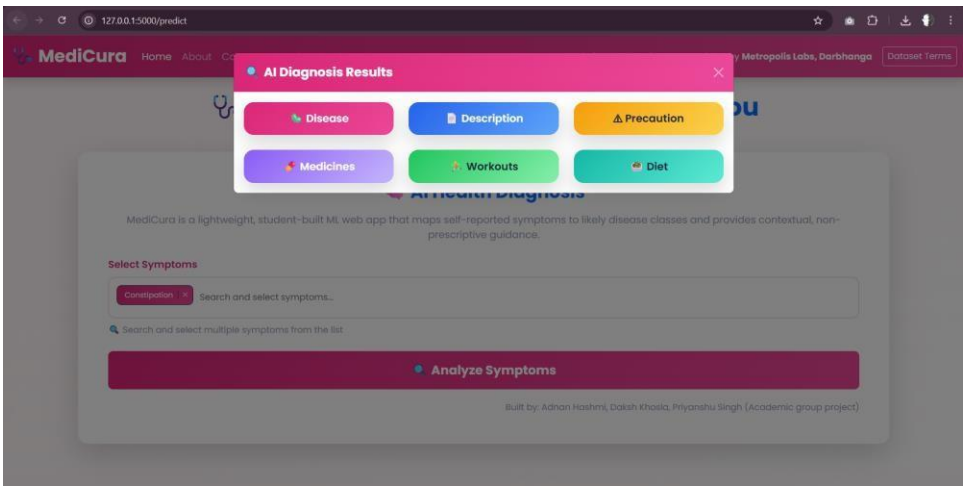


Figure 4.5 — Disease Description & Precautions

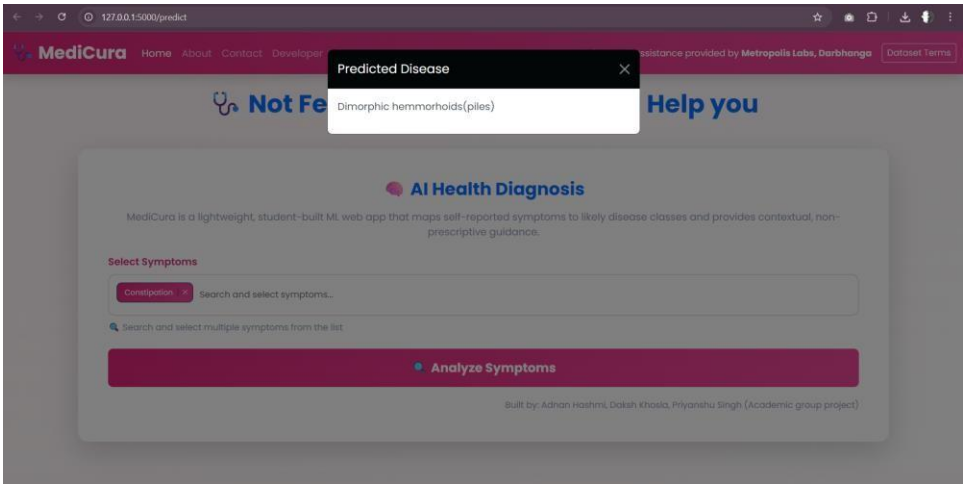


Figure 4.6 — Medication Advisory Panel

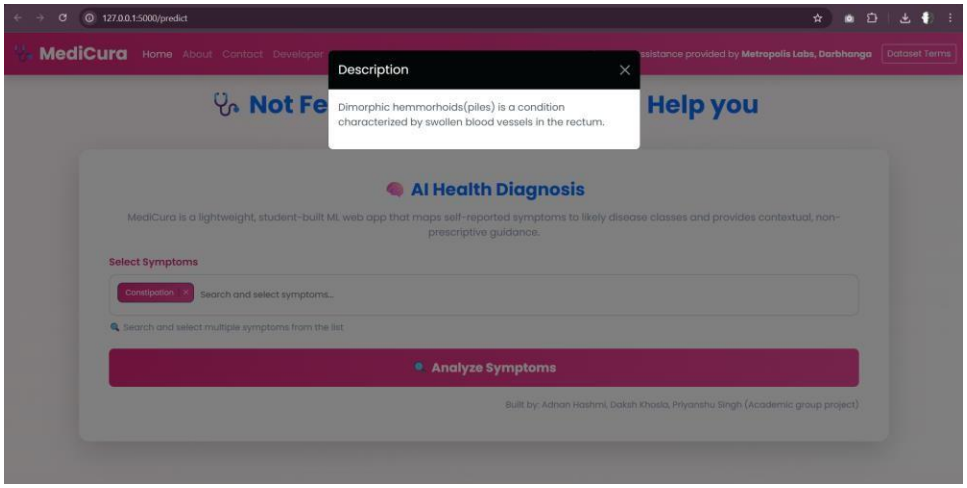


Figure 4.7 — Diet & Activity Recommendations

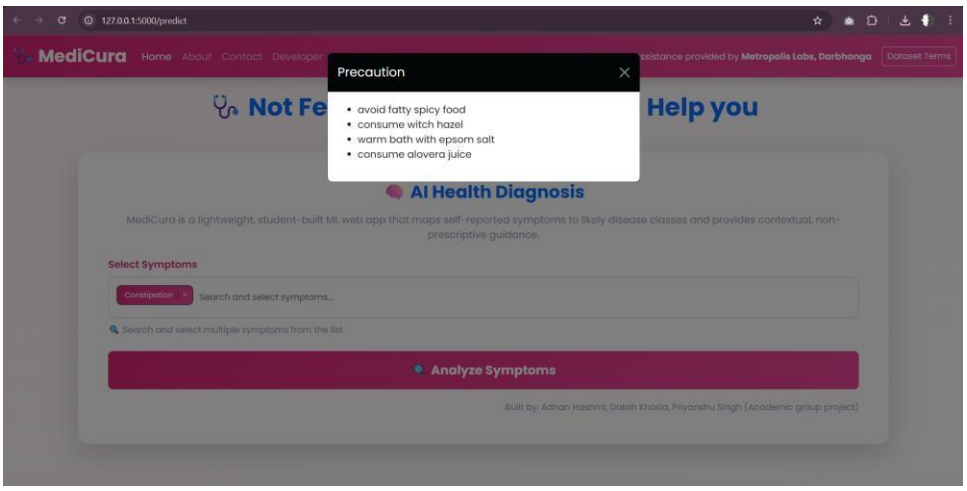


Figure 4.8 — Explainability — Top Contributing Symptoms

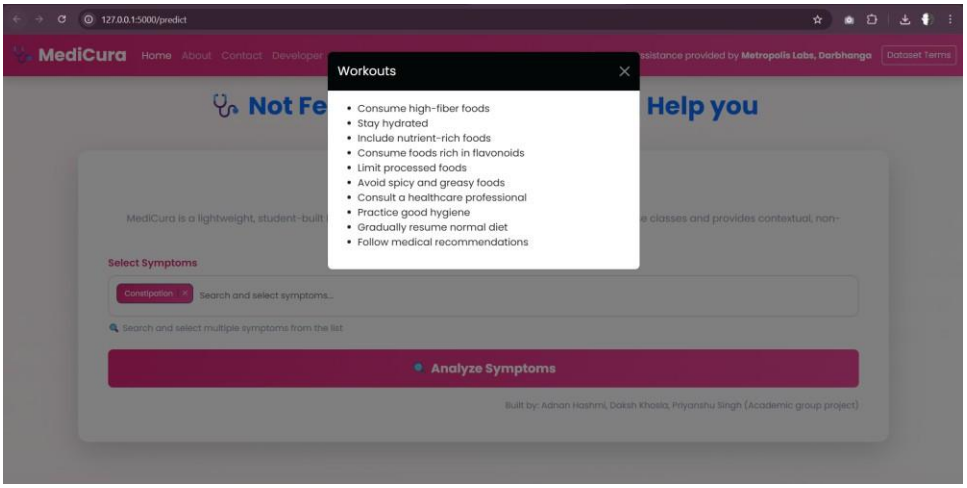


Figure 4.9 — Admin CSV Upload Interface

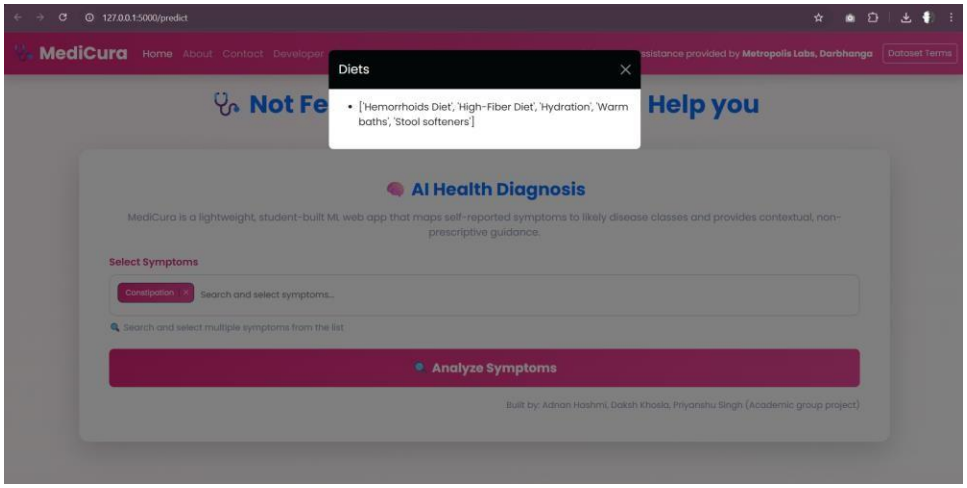


Figure 4.10 — Model Info Endpoint

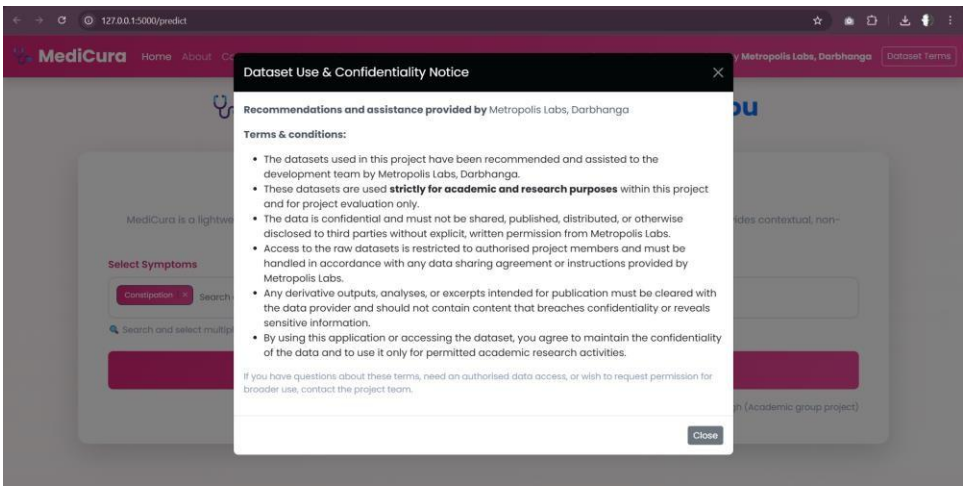


Figure 4.11 — Error Handling — Unknown Symptom

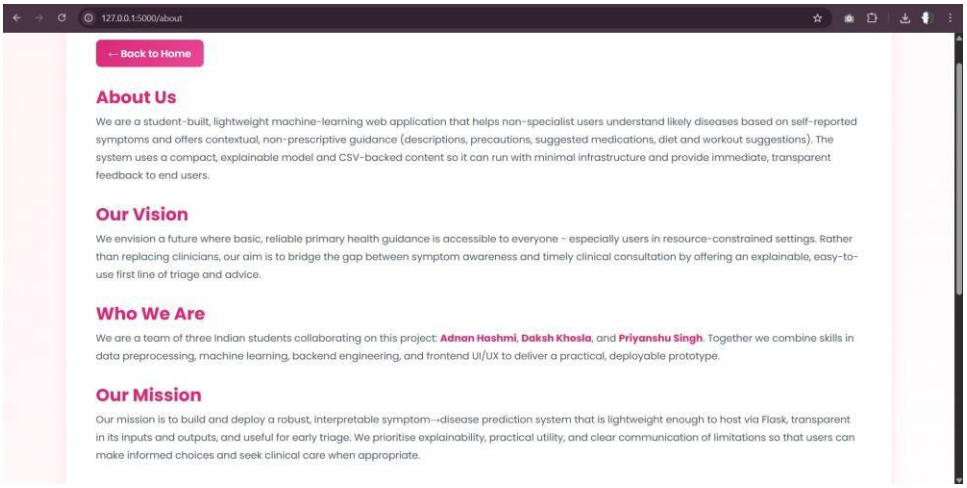


Figure 4.12 — Responsive Mobile View

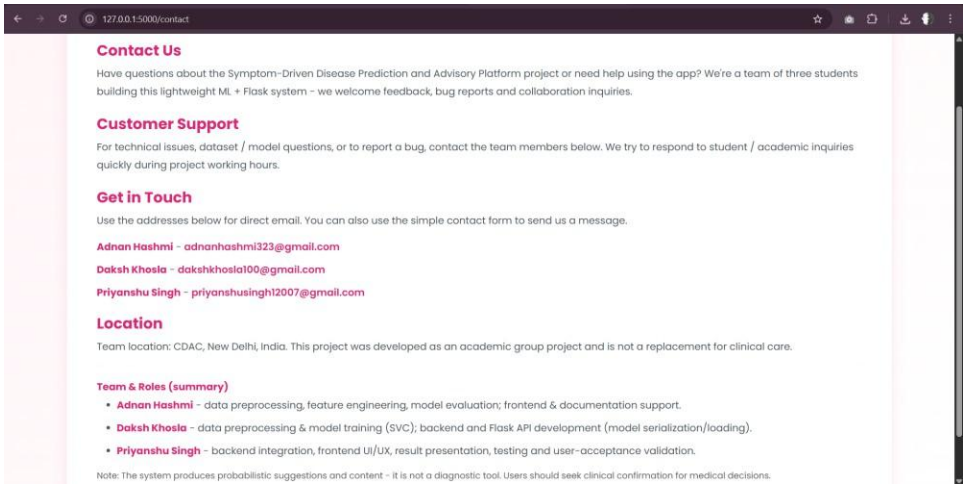


Figure 4.13 — Safety Disclaimer Overlay

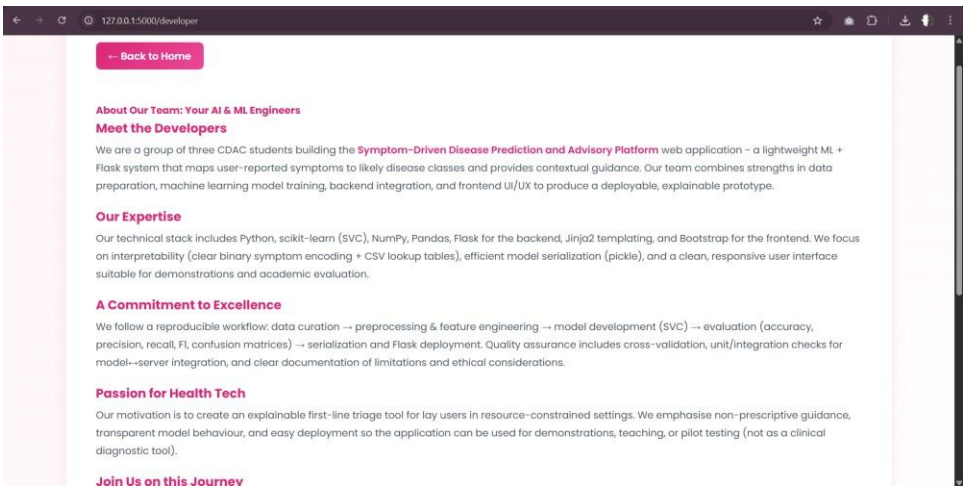


Figure 4.14 — Confusion Matrix (Test Set)

Figure 4.15 — Per-Class Precision & Recall

5. Ethical, Legal and Regulatory Considerations

5.1 Risk Assessment

Risks Identified:

- Misclassification of serious conditions leading to delayed care.
- Data privacy breaches if PII is inadvertently stored.
- Misuse by users (e.g., substituting for clinical diagnosis).

Mitigation Measures:

- Clear, prominent non-diagnostic disclaimers throughout the interface.
- Automated detection of high-risk symptom combinations prompting urgent referral advice.
- No PII collected by default; strict consent flows and encryption implemented if user accounts are introduced in future.

5.2 User Data Privacy and Consent

- For demo usage: no authentication and no PII collection by design.
- For future personal data collection features: implement explicit consent, encryption at rest, and retention policies compliant with local regulations.
- For production in regulated jurisdictions: consult legal teams for GDPR/HIPAA compliance.

5.3 Clinical Validation and Regulatory Pathways

If this prototype is to be used clinically, the following steps are necessary:

- Engage clinical partners for prospective validation studies comparing model suggestions against clinician diagnosis.
- Consider regulatory classification: in many jurisdictions, a symptom checker offering diagnostic suggestions may be classified as a medical device — consult regulators early.
- Document all clinical evaluation protocols and obtain ethical approvals where necessary.

6. Conclusions and Future Work

6.1 Conclusions

The Symptom-Driven Disease Prediction and Advisory Platform demonstrates a compact, interpretable approach to symptom-based triage. Using a deterministic symptom-to-index mapping and a linear SVC baseline provides strong baseline performance with transparent explanations — attributes that are valuable for early prototyping and educational demonstration. The CSV-driven content approach decouples medical text maintenance from code changes, enabling subject-matter experts to refine guidance independently.

6.2 Future Work Roadmap

Short-Term (0–3 months)	• Expand dataset with more labeled examples, focusing on under-represented classes.
	• Integrate a more robust synonym and spell-correction module.
	• Add confidence thresholding and low-confidence workflows.
Medium-Term (3–12 months)	• Pilot with a clinical partner for prospective validation on real patient intake data.
	• Implement SHAP-based per-prediction explanations and evaluate clarity for lay users.
	• Add multi-language support for UI and content CSVs.
Long-Term (12+ months)	• Explore hierarchical classification or specialized sub-models for high-confusion groups.
	• Integrate optional clinician review workflows and secure data collection pipelines.
	• Engage regulatory consultants to chart a pathway for clinical decision support accreditation.
	•

7. References

[1] Corinna Cortes, Vladimir Vapnik. *Support-Vector Networks.* Machine Learning, 1995.

[2] Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier.* KDD, 2016.

[3] Scott Lundberg, Su-In Lee. *A Unified Approach to Interpreting Model Predictions (SHAP).* NIPS, 2017.

[4] H. L. Semigran, J. A. Linder, C. Gidengil, et al.. *Evaluation of Symptom Checkers for Self-Diagnosis and Triage: Audit Study.* BMJ, 2015.

[5] M. M. Ahsan, Z. Siddique. *Machine-Learning-Based Disease Diagnosis: A Comprehensive Review.* Healthcare, 2022.

[6] F. Sogandi, et al.. *Identifying Disease Symptoms and General Rules using Supervised & Unsupervised Machine Learning.* IJCSI, 2019.

[7] A. F. Tchango, et al.. *DDXPlus: A New Dataset for Automatic Medical Diagnosis.* NeurIPS, 2022.

[8] X.-Z. Zhou, et al.. *Human Symptoms-Disease Network.* Nature Communications, 2014.

[9] M. Abulaish, et al.. *DiseaSE: A Biomedical Text-Analytics System for Disease-Symptom Extraction.* 2019.

[10] E. Xia, et al.. *Mining Disease-Symptom Relations from Massive Biomedical Literature.* 2020.