HEAR-UI: Explainable AI-Powered Decision Support for Cochlear Implant Success Prediction

Adelia Manafov[1] and Artem Mozharov[1] and Niels Kuhl[1]
extsuperscript1University of Marburg, Germany
{manafova, mozharoa, kuhl}@students.uni-marburg.de
Supervisors: Prof. Dr. Christin Seifert; M.Sc. Khawla Elhadri

Abstract

Cochlear implants (CI) can significantly improve hearing for selected patients, but determining suitable candidates requires careful clinical assessment. We present HEAR-UI, an AI-powered clinical decision support system that predicts CI success probability (0-100%) and provides explainable model outputs using SHAP-based feature importance analysis. The system integrates a 68-feature Logistic Regression classifier with preprocessing, a FastAPI REST API backend, PostgreSQL database, and a Vue.js 3 frontend. Our implementation achieves 72% test coverage across 268 automated tests, supports real-time predictions with sub-200ms latency, and provides interpretable explanations for clinical decision-making. We demonstrate deployment-ready architecture using Docker Compose, comprehensive CI/CD pipelines via GitHub Actions, and validated the system with 5 real patient cases. The project emphasizes transparency, reproducibility, and explainability — critical requirements for clinical AI systems. All code, documentation, and deployment configurations are openly available.

1 Introduction

1.1 Problem Statement

Cochlear implantation is a transformative medical intervention that can restore hearing to profoundly deaf patients. However, the procedure involves significant risks, costs, and post-operative rehabilitation requirements. Not all patients achieve successful outcomes, and predicting which candidates will benefit most remains a clinical challenge (**??**).

Current clinical practice relies primarily on qualitative assessments and clinician experience. While effective, this approach lacks systematic integration of historical outcome data and quantitative risk-benefit analysis. There is a clear need for data-driven tools that can:

- Predict individual patient success probability
- Explain *which* patient factors drive predictions
- Integrate seamlessly into clinical workflows
- Maintain transparency and interpretability

1.2 Research Questions

This project addresses three core research questions:

RQ1: Can a clinical dataset with heterogeneous patient features be transformed into a robust, interpretable machine learning model for CI success prediction?

RQ2: Can post-hoc explainability methods (SHAP) be integrated into a deployed clinical decision support system to provide actionable insights for clinicians?

RQ3: What architectural patterns and deployment strategies enable production-ready, maintainable AI systems for clinical decision support?

1.3 Contributions

Our key contributions are:

1. A complete, deployment-ready clinical decision support system with frontend, backend, and database
2. Integration of SHAP explanations for interpretable ML predictions
3. Comprehensive testing strategy (268 tests, 72% coverage)
4. Reproducible Docker-based deployment

5. Validated CI/CD pipelines with automated testing

6. Open-source codebase with extensive documentation

2 Related Work

2.1 Explainable AI in Healthcare

Explainable AI (XAI) has become critical for clinical decision support systems. Lundberg and Lee (**?**) introduced SHAP (SHapley Additive exPlanations), a unified framework for interpreting model predictions based on game-theoretic Shapley values. SHAP provides both local (per-prediction) and global (model-wide) explanations, making it particularly suitable for clinical applications where understanding *why* a model makes specific predictions is as important as the prediction itself.

Ribeiro et al. (**?**) proposed LIME (Local Interpretable Model-agnostic Explanations), an alternative explanation method that approximates any black-box model locally with an interpretable model. While LIME is model-agnostic, SHAP offers stronger theoretical guarantees and consistent feature attribution (**?**).

Rudin (**?**) argues that in high-stakes domains like healthcare, inherently interpretable models (e.g., logistic regression, decision trees) should be preferred over post-hoc explanations of black-box models. Our work aligns with this philosophy by using Logistic Regression as the primary classifier.

2.2 AI in Medicine

Topol (**?**) provides a comprehensive overview of AI applications in medicine, emphasizing the convergence of human and artificial intelligence. He highlights that successful clinical AI systems must prioritize:

- Transparency and interpretability
- Integration with existing clinical workflows
- Validation on diverse patient populations
- Ethical considerations and bias mitigation

Shortliffe and Sepúlveda (**?**) discuss clinical decision support systems (CDSS) in the era of AI, noting that adoption is often hindered by lack of trust, poor integration, and insufficient explanation of recommendations.

2.3 Cochlear Implant Outcome Prediction

Wilson and Dorman (**?**) provide a comprehensive review of cochlear implant technology and outcomes, noting substantial variability in patient success rates. Lenarz (**?**) discusses state-of-the-art CI technology and emphasizes the need for better patient selection criteria.

Several studies have attempted to predict CI outcomes using patient demographics, audiological measures, and imaging data (**?**). However, most rely on traditional statistical methods (e.g., logistic regression, Cox proportional hazards) without modern ML explainability tools.

2.4 Gap in Literature

To our knowledge, no prior work has demonstrated a *complete*, *deployed*, *explainable* CI decision support system with:

- Modern web-based architecture (REST API, Vue.js frontend)
- Real-time SHAP explanations
- Comprehensive automated testing
- Docker-based reproducible deployment
- Open-source availability

HEAR-UI fills this gap by providing a production-ready reference implementation for clinical AI systems.

3 System Architecture

3.1 Overview

HEAR-UI follows a modern microservices architecture with three primary components (Figure **??**):

1. **Backend (FastAPI):** REST API serving predictions, explanations, and patient data management
2. **Frontend (Vue.js 3):** Single-page application for clinicians
3. **Database (PostgreSQL 12):** Persistent storage for patients, predictions, and feedback

All components are containerized using Docker and orchestrated via Docker Compose, ensuring reproducible deployment across development, testing, and production environments.

3.2 Backend Architecture

The FastAPI backend consists of:

- **API Routes** (`app/api/routes/`): RESTful endpoints for predictions, explainer, patients, feedback, and utilities
- **Core Logic** (`app/core/`): Model wrapper, SHAP explainer, preprocessor, background data generation
- **Database Models** (`app/models/`): SQLModel schemas for patients, predictions, feedback
- **Tests** (`app/tests/`): 202 unit and integration tests

Technology Stack:

- FastAPI 0.115.5 (async, auto-generated OpenAPI docs)
- Pydantic 2.10.3 (data validation)
- SQLModel 0.0.22 (type-safe ORM)
- Alembic 1.14.0 (database migrations)
- scikit-learn 1.5.2 (ML pipeline)
- SHAP 0.46.0 (explainability)

3.3 Frontend Architecture

The Vue.js 3 frontend provides:

- **Patient Management:** Search, view, create patients
- **Prediction Interface:** Request predictions for specific patients
- **SHAP Visualization:** Bar charts showing feature importance
- **Feedback System:** Capture clinician agreement/disagreement with predictions

Technology Stack:

- Vue.js 3.5.13 (Composition API)
- TypeScript 5.6.3 (type safety)
- Vite 6.0.1 (build tool, HMR)
- Vuetify 3.7.5 (Material Design components)
- Playwright 1.48.2 (E2E testing)

3.4 Database Schema

PostgreSQL stores:

- **Patients:** Demographics, audiological data, implant details (68 features after preprocessing)
- **Predictions:** Cached predictions with timestamps
- **Feedback:** Clinician responses to predictions

Alembic manages schema migrations, ensuring version control and reproducible deployments.

3.5 API Design

RESTful endpoints follow industry best practices (Table **??**). All endpoints return JSON responses with HTTP status codes. OpenAPI/Swagger documentation is auto-generated at `/docs`.

Table 1: Core REST API Endpoints

Endpoint	Method	Purpose
`/predict`	POST	Prediction
`/explainer/explain`	POST	SHAP values
`/patients/`	GET	List patients
`/patients/{id}`	GET	Get patient
`/feedback/`	POST	Submit feedback
`/health-check/`	GET	System health

4 Data and Preprocessing

4.1 Dataset Description

The dataset consists of de-identified cochlear implant patient records including demographics, audiological measures (onset type, duration, cause), pre-operative symptoms, and implant details. After preprocessing and one-hot encoding, 68 features are generated. Five fully annotated test patients are publicly available; the full training cohort remains private for GDPR compliance.

4.2 Preprocessing Pipeline

Preprocessing (`backend/app/core/preprocessor.py`) includes: (1) column mapping from German labels to English, (2) domain-informed default imputation, (3) one-hot encoding with deterministic feature ordering, (4) StandardScaler for numerical features. The `preprocess()` function accepts raw dictionaries and returns numpy arrays aligned with the trained pipeline.

5 Machine Learning Method

5.1 Model Selection

We chose **Logistic Regression** as the primary classifier for several reasons:

1. **Interpretability:** Coefficients directly indicate feature importance
2. **Calibration:** Outputs are naturally probabilistic (0-1 scale)
3. **Clinical Acceptance:** Widely understood by medical professionals
4. **Baseline Performance:** Strong baseline for high-dimensional data
5. **Explainability:** Linear models work seamlessly with SHAP

Rudin (**?**) argues that in high-stakes domains, inherently interpretable models should be preferred over black-box models with post-hoc explanations. Logistic Regression satisfies this requirement.

5.2 Model Architecture and Training

The model is a scikit-learn Pipeline: StandardScaler + OneHotEncoder → LogisticRegression (L1 penalty, C=10.0, liblinear solver). Serialization via `joblib` ensures the complete preprocessing pipeline is preserved. Hyperparameters were tuned for interpretability and regularization balance.

5.3 SHAP Explainability

We integrate SHAP (**?**) for per-prediction feature importance. The explainer (`backend/app/core/shap_explainer.py`) uses `LinearExplainer` for logistic regression, with synthetic background data for stabilization. API responses include `base_value` (average prediction), `shap_values` (feature attributions), and `top_features` ranked by importance.

6 Evaluation

6.1 Testing Strategy

We implement a comprehensive multi-level testing strategy (Table **??**):

Table 2: Testing Coverage Summary

Test Level	Tests	Coverage
Unit Tests	202	72%
Integration Tests	48	API + DB
E2E Tests (Playwright)	18	Frontend
Total	**268**	**72%**

Tests cover model loading, SHAP alignment, API endpoints, database CRUD, security (password hashing), and end-to-end workflows. Coverage analysis (`pytest-cov`) focuses on production code, achieving 72% coverage (**?**).

6.2 CI/CD Pipeline

GitHub Actions automate testing and deployment:

- **Linting:** Ruff (Python), Biome (TypeScript)
- **Unit Tests:** pytest on every push/PR
- **E2E Tests:** Playwright on PR to main
- **Docker Build:** Validate containers build successfully
- **Coverage Report:** Generate HTML reports for review

All workflows are defined in `.github/workflows/`.

6.3 System Validation

We validated with 5 real patient cases (Table **??**). Postlingual onset strongly predicts success; syndromal causes correlate with lower rates. Predictions are reproducible and SHAP explanations align with clinical intuition.

6.4 Performance Metrics

Prediction latency averages 150ms; SHAP explanation 350ms. Async FastAPI with PostgreSQL pooling supports concurrent requests; Docker Compose scales to multiple backend replicas.

Table 3: Patient Validation Results

Patient	Prediction	Key Feature
Patient 1	97.3%	Postlingual onset
Patient 2	99.75%	CI contralateral
Patient 3	22.1%	Syndromal cause
Patient 4	81.1%	Perilingual onset
Patient 5	97.3%	Postlingual onset

7 Results and Discussion

7.1 Implementation Status

Fully implemented: REST API (17 endpoints), ML prediction with SHAP, PostgreSQL database, Vue.js frontend, Docker deployment, CI/CD (7 workflows). Testing: 268 tests (98.5% pass), 72% coverage, all workflows validated end-to-end.

7.2 Key Findings

7.2.1 Key Findings

SHAP analysis identifies top predictive features: hearing loss onset (postlingual $>$ perilingual $>$ prelingual), contralateral ear status, cause (syndromal negative), deafness duration, and age. These align with clinical literature (**??**). System handles missing values, ensures deterministic predictions, and gracefully manages edge cases.

7.3 Limitations

Data: Small public sample (5 patients), binary labels may oversimplify outcomes, single-center data limits generalizability. **Model:** Linearity assumption may miss complex patterns; calibration analysis needed (**?**). **Deployment:** Lacks authentication, audit logging, and production monitoring. Future work includes multi-center validation, calibration testing, non-linear model comparison, and prospective clinical studies.

8 Deployment and Reproducibility

All components are containerized (Docker Compose orchestrates backend, frontend, PostgreSQL). Deployment: clone repository, configure `.env`, run `docker compose up`, execute migrations. Documentation includes README, technical docs, API guides (Swagger), and validation reports. Reproducibility: fixed dependencies, deterministic features, random seeds, CI/CD validation.

9 Conclusion

HEAR-UI is a deployment-ready clinical decision support system integrating interpretable ML (Logistic Regression, 68 features), SHAP explainability, modern web architecture (FastAPI, Vue.js, PostgreSQL), comprehensive testing (268 tests, 72% coverage), and Docker deployment. Validation with 5 patients shows reproducible predictions ($<$ 200ms latency) with clinically intuitive SHAP explanations.

The system advances clinical AI by demonstrating complete integration, prioritizing explainability, and emphasizing transparency through open-source release. Future work includes multi-center validation, calibration analysis, security enhancements, and prospective clinical studies. HEAR-UI serves as a reference implementation showing that explainability and performance are compatible when paired with rigorous software engineering practices.

Declaration of Generative AI Usage

Generative AI (GitHub Copilot, Claude) was used for:

- Code completion and boilerplate generation
- Documentation drafting and structuring
- Test case generation
- LaTeX formatting and bibliography management

All AI-generated content was reviewed, edited, and validated by the authors. Technical descriptions were verified against repository code. No AI-generated code was committed without human review and testing.

Acknowledgements

We thank our supervisors, Prof. Dr. Christin Seifert and M.Sc. Khawla Elhadri, for their continuous guidance and invaluable feedback throughout the project.

Contribution Statement

Adelia Manafov:

- Backend architecture and implementation (FastAPI, SQLModel)
- ML model integration and preprocessing pipeline
- SHAP explainer wrapper and API integration
- Database schema design and migrations (Alembic)
- CI/CD pipeline setup (GitHub Actions)
- Testing strategy and implementation (pytest, 202 backend tests)
- Docker Compose configuration
- System validation and documentation
- Final report preparation

Artem Mozharov:

- Frontend architecture and implementation (Vue.js 3, TypeScript)
- UI/UX design (Vuetify components)
- Patient management interface
- Prediction and SHAP visualization components
- I18n (internationalization) setup
- E2E testing (Playwright, 18 tests)
- Frontend documentation
- Demo video preparation

Niels Kuhl:

- Clinical domain expertise
- Dataset provision and annotation
- Validation of clinical relevance

References

Thomas Lenarz. 2018. Cochlear implant–state of the art. *Laryngo-Rhino-Otologie*, 97(S01):S123–S151.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774.

Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. 2022. Interpretable machine learning–a brief history, state-of-the-art and challenges. *Communications in Computer and Information Science*, 1323:417–431.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 625–632.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you? explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144.

Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Edward H Shortliffe and Martin J Sepúlveda. 2018. Clinical decision support in the era of artificial intelligence. *JAMA*, 320(21):2199–2200.

Ian Sommerville. 2015. *Software Engineering*, 10 edition. Pearson.

Eric Topol. 2019. High-performance medicine: the convergence of human and artificial intelligence. *Nature Medicine*, 25:44–56.

Blake S Wilson and Michael F Dorman. 2008. Cochlear implants: a remarkable past and a brilliant future. *Hearing Research*, 242(1-2):3–21.