TED UNIVERSITY

Faculty of Engineering
Department of Software Engineering
Department of Computer Engineering
CMPE 492 / SENG 492

XAI Healthcare Bot

Final Report

by

Doğukan Yetgin
Hilal Yurtoğlu
Ruşen Deniz Kaplan
Zeynep Bölükbaşı

1. Introduction

1.1. Project Overview

The XAI for Healthcare Bot is the medical assisting system that aimed to improve the diagnosing support by the adoption of the Explainable Artificial Intelligence (XAI) technologies. The problem it seeks to address is that of late and non-transparent medical diagnoses in high-volume and access constrained systems. The bot uses state-of-the-art AI techniques alongside explainability to support doctors in analysing patient data and medical imaging as they provide the EMF score to the patient. This twofold strategy is not only the baseline of trust in AI applications in healthcare, it also promotes communication between doctors and patients as well as enables better and faster medical decision-making.

1.2. Objectives of the Report

The objective of this report is to document the architectural design, subsystem decomposition, and technical components of the XAI Healthcare Bot. It explains how the system achieves transparency and reliability in medical decision-making, the technologies used in implementation, and the rationale behind architectural decisions. Furthermore, the report elaborates on explainability methods and how the system maintains compliance with data privacy regulations.

1.3. Scope

The report emphasizes the product design and development of an explainability-centric AI-based diagnosis system. The system is designed to enable diagnosis of Alzheimer's disease by analyzing medical images and structured information with OpenCV and Grad-CAM algorithms. User profiles by role will be employed to enable customized experiences for doctors and patients. Diagnostic recommendations in real-time and medication alerts will be incorporated to maximize treatment efficiency. The system will be designed to integrate easily with existing Electronic Health Records (EHR) systems and ensure complete compliance with international standards of information security and privacy like HIPAA and GDPR.

2. System Architecture and Design

2.1. Final Architecture

The XAI Healthcare Bot employs a layered structure that is scalable, secure, and optimizes information processing. The structure is made of three main layers.

Presentation Layer

This interface is used by patients and healthcare providers to interact with the system. Built with React.js, it offers an intuitive and responsive experience suited to various users. Healthcare providers have access to thorough diagnostic information and

patient records, whereas patients have access to explanations and medication reminders presented in an easily accessible interface.

Business Logic Layer

The heart of the XAI Healthcare Bot is contained in this level, where processing of data, execution of AI models, and generation of explanations take place. This level uses Python with TensorFlow and NumPy to perform machine learning, where OpenCV is applied to process images and extract features of medical images. This level further integrates explainability modules with Grad-CAM to provide clear explanations of AI decisions.

Data Management Layer

This layer uses a hybrid database structure to handle varied healthcare information efficiently. PostgreSQL is used as the main relational database to store structured patient information like diagnosis reports and medication records. Unstructured information like medical images and results of AI model interpretability comes to MongoDB. The overall data layer is built to be HIPAA and GDPR compliant to guarantee patient privacy and security.

The system has dedicated subsystems to handle data gathering, image processing, and backend operations, with all of them collaborating to provide precise, explainable medical results.

2.2. Design Details

The system has a role-based interface with personalized dashboards for patients and physicians, diagnostic viewer, patient portal, and accessibility features. OpenCV and CNNs preprocess medical images and records to extract features and detect anomalies. Explanation is done with Grad-CAM to provide visualized heatmaps, and technical findings presented to patients in easily understandable terms. Alerts inform users of medications, appointments, and diagnosis.

2.3. Diagrams

We created several diagrams to provide a visual representation of the XAI Healthcare Bot's architecture, components, and data flow. The following diagrams illustrate the structure and interactions between the User Interface, AI Diagnosis Engine, and hybrid Data Management layers.

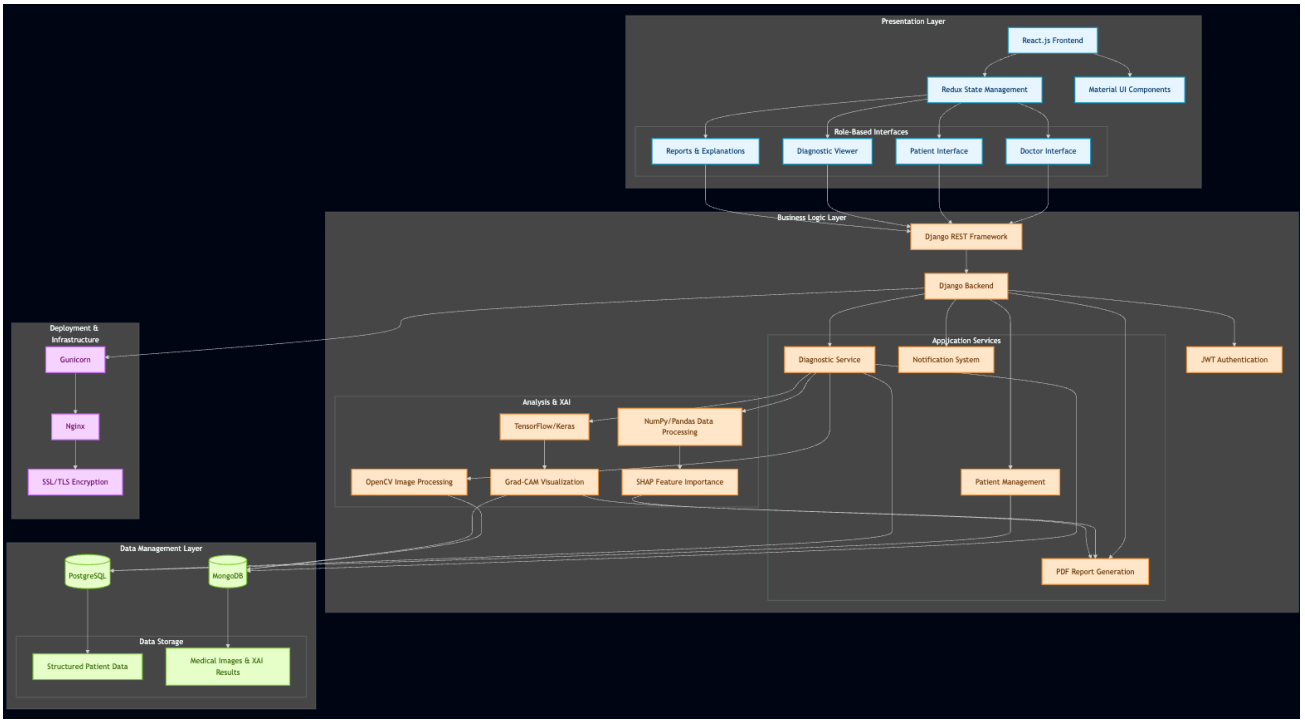

Figure 1: System Architecture Diagram

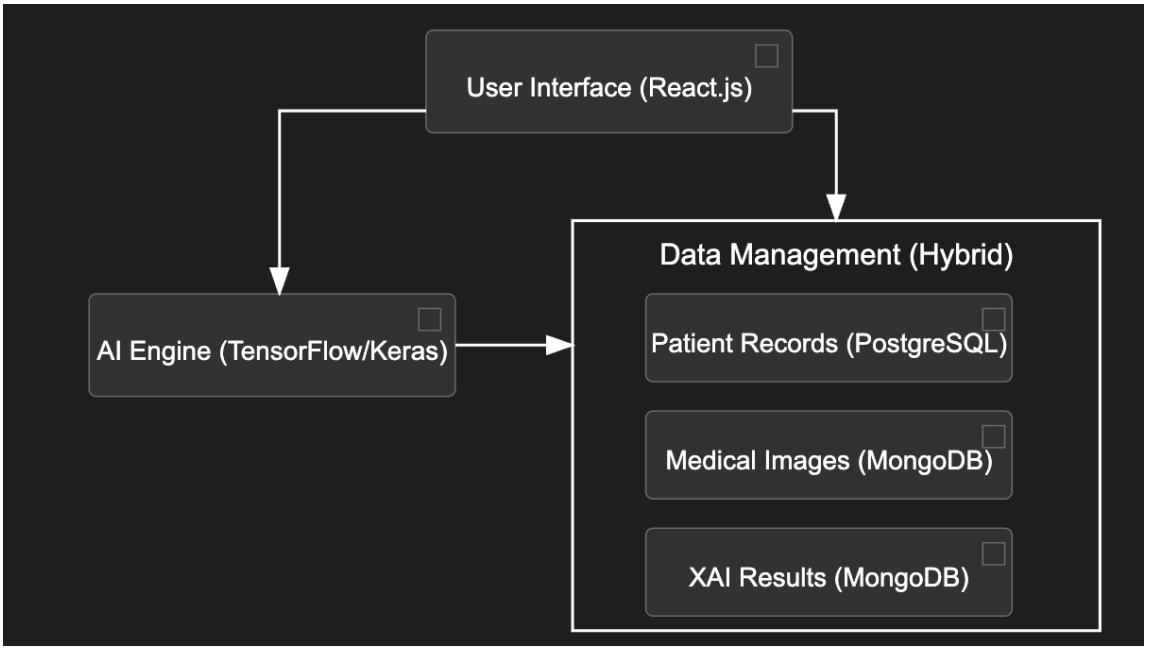

Figure 2: XAI Healtcare Bot Components

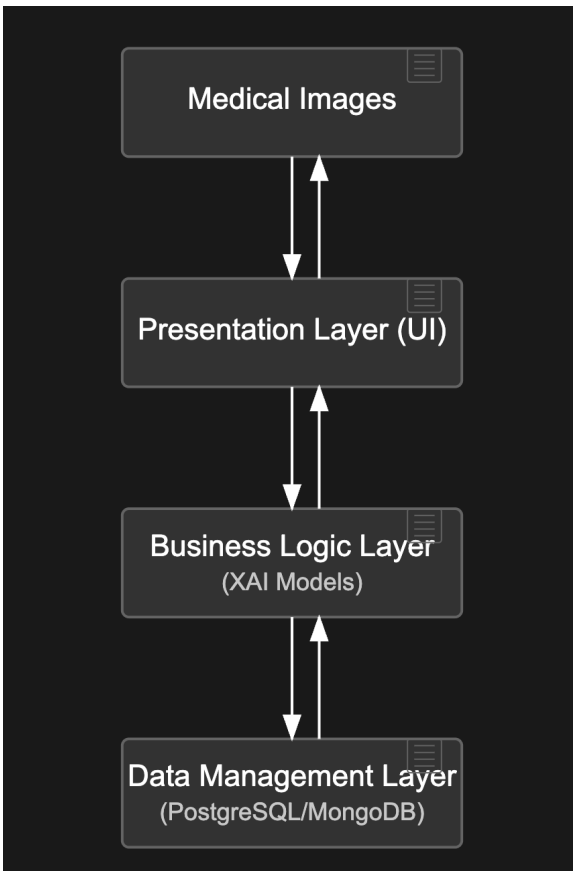

Figure 3: Data Flow Diagram

3. System Architecture and Design

3.1. Detailed Workflow and Process

The deployment process is structured to deliver a reliable and sustainable user experience for both developers and end-users. The following steps outline the complete end-to-end deployment cycle to ensure seamless implementation and operation.

Development and Testing

- In the development environment, the React frontend and Django backend operate on distinct ports. Specifically, the React application runs on localhost:3000, while the Django backend is hosted on localhost:8000.
- The Django backend serves data to the React frontend via RESTful APIs, ensuring seamless interaction between the two components.
- The AI models, including the Convolutional Neural Network (CNN) and GradCam, are initially tested locally before being integrated into the Django backend.

Build and Packaging

- The React application is compiled for production using the npm run build command. This generates optimized static files, which are stored in the build directory. These files are then incorporated into Django's static files system for seamless deployment.

Environment Configuration

- Sensitive information, including database credentials, JWT secrets, and API keys, is securely managed using .env files to ensure proper separation from source code.
- For production deployments, critical settings are configured in settings.py. These include disabling debug mode (DEBUG=False), specifying permitted hosts (ALLOWED_HOSTS), and defining the static files directory (STATIC_ROOT).

Deployment to Production Server

- The application is deployed to a cloud server either by securely transferring files via SCP or by cloning/updating the repository using git pull.
- Once deployed, a Python virtual environment (venv) is configured to maintain dependency isolation. All required packages are then installed by executing pip install -r requirements.txt to ensure consistent dependencies across environments.

Database Migrations

- Database schema updates are applied using:

```
python manage.py makemigrations
python manage.py migrate
```

Static Files Collection

- Static assets, including the React build files and related CSS/JS resources, are collected into a single directory using:

```
python manage.py collectstatic
```

Web Server and Application Server Setup

- The Django application is served using Gunicorn.
- Nginx is configured as a reverse proxy to route requests to the backend and serve static files efficiently.

SSL Configuration

- SSL certificates are obtained via Let's Encrypt.

- Nginx is configured to support HTTPS, ensuring secure client-server communication.

3.2. Technologies Used

Layer	Technology	Purpose
Backend	Django, Django REST Framework	API development and database operations
Frontend	React, Redux, Material UI	Dynamic, role-based user interface
Data Science	TensorFlow/Keras, GradCam	Image analysis and explainable AI outputs
Visualization	Chart.js	Visualization of feature importance and health data
Data Processing	Pandas, NumPy	Medical data preparation and analysis
Deployment	Gunicorn, Nginx, PostgreSQL	Production server setup
Security	JWT (djangorestframework-simplejwt)	User authentication and authorization

3.3. Modules

Frontend

The front end is structured into two main panels based on user roles:

Doctor Panel

- View and filter patient lists.
- Create new medical records.
- Visualize XAI analysis results (GradCam outputs).

Patient Panel

- View personal medical records.
- View doctor comments and suggestions.
- Understand simplified explanations of XAI decisions.

Technologies Used:

- **React Router** for page navigation.
- **Axios** for making API calls.
- **Material UI** for building responsive and user-friendly components.

Backend

The Django backend manages operations for both doctors and patients using a RESTful API architecture.

Main Modules:

- users: Authentication, JWT token generation, and role-based access control.
- patients: Management of patient records and related data.
- analysis: AI model inference and GradCam visualization.
- reports: PDF report generation and download functionalities.

XAI Integration:

- GradCam results are temporarily stored and presented to doctors.

3.4. Code Snippets

Authentication - JWT Token Generation

```
from rest_framework_simplejwt.views import TokenObtainPairView


urlpatterns = [
    path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair'),
]
```

Patient API – Listing

```
class PatientListView(generics.ListAPIView):
    queryset = Patient.objects.all()
    serializer_class = PatientSerializer
    permission_classes = [IsAuthenticated, IsDoctor]
```

GradCam Implementation - Backend

```
def generate_gradcam(image_path):
    model = load_model('models/cnn_model.h5')
    img_array = preprocess_input(image_path)
    heatmap = make_gradcam_heatmap(img_array, model)
    return overlay_heatmap_on_image(heatmap, image_path)
```

React – Fetching Patient List in Doctor Panel

```
useEffect(() => {
  axios.get('/api/patients/', { headers: authHeader })
    .then(res => setPatients(res.data))
    .catch(err => console.error(err));
}, []);
```

4. Testing and Evaluation

4.1. Test Plan Summary

This plan covers the following components of XAI Healthcare Bot:

- User Authorization
- Model's Decisions
- Notification System

Objectives:

- Verify if each component meets its functional requirements
- Identify critical defects and bottlenecks

Scope:

- Functional Testing
- Performance Testing
- Accuracy Measurement

Methods:

- Execute manual and automated tests
- Split the training data for train, validation and test

Success Requirements:

- 100% pass rate for authorization and functional requirements
- We don't have an exact success rate for our model we are aiming the best possible accuracy

4.2. Test Cases & Results

Test Name	Test Step	Expected Result
Login with correct password and username (Doctor)	A user with doctor role tries to login with correct username and password	Login should be successful
Login with correct password and username (Patient)	A user with patient role tries to login with correct username and password	Login should be successful
Login with incorrect role (Doctor)	A user with doctor role tries to login with correct username and password for the patient panel	Invalid email or password for selected role
Login with incorrect role (Patient)	A user with patient role tries to login with correct username and password for the doctor panel	Invalid email or password for selected role
Login with incorrect password and/or username (Doctor)	A user with doctor role tries to login with incorrect username and/or password	Invalid email or password for selected role
Login with incorrect password and/or username (Patient)	A user with patient role tries to login with incorrect username and/or password	Invalid email or password for selected role

(Authorization Tests)

Classification Report:

	precision	recall	f1-score	support
Mild Impairment	1.00	0.88	0.94	512
Moderate Impairment	1.00	1.00	1.00	512
No Impairment	1.00	0.58	0.73	512
Very Mild Impairment	0.65	1.00	0.79	512
accuracy			0.86	2048
macro avg	0.91	0.86	0.86	2048
weighted avg	0.91	0.86	0.86	2048

(Model Accuracy)

4.3. Assessment of the Tests

Strengths:

- Functional requirements are met
- Authorization phase is successful
- Model accuracy is satisfying

Action Items:

- Notification tests should be done
- System should be tested under heavy load
- Data security should be considered as a main topic in future

5. Impact Analysis

5.1. Global Impact

Patient Access: The healthbot helps globalization since it enables preliminary diagnosis and guidance remotely. So it reduces geographic barriers.

Healthcare System: Reduces hospitals workloads, helps staff to focus on more critical cases

International Standarts: Complying with laws like GDPR, HIPAA, and others makes cross-border integration easier.

5.2. Economic Impact

Cost Savings: Automated triage can lower healthcare costs by 10 - 15%

Market Opportunities: Provides a competitive edge in the health technology industry; modular add-ons and subscription services are possible. Possibly created differences in the insurance sector.

5.3. Societal Impact

Equity and Accessibility: XAI healthcare bot is accessible as long as you have connection to the internet and a phone.

Patient Satisfaction: User experience and engagement are improved by rapid feedback and ongoing observation.

Ethics and Trust: Builds trust with transparent data use policy.

6. Contemporary Issues and Use of Resources

Data Privacy & Security: Large-scale health data is vulnerable to cyberattacks and phishing; in further process it is possible to use end-to-end encryption, conduct frequent security audits, and use the most recent standards (such AES-256)

Regulation and Compliance: Laws such as KVKK are evolving through time, so it is important to check them regularly.

Ethics in AI: To make decisions understandable, adding explainable tools.

Training and Support Resources: Develop FAQs for both parties.

7. Current Trends and Future Directions

Recent advances in explainable AI (XAI) have accelerated its adoption in healthcare, driven by the need for transparent, trustworthy diagnostic tools. Techniques such as Grad-CAM are becoming standard for elucidating model decisions on medical images, fostering clinician trust and enabling regulatory compliance. There is a growing shift toward integrating real-time AI diagnostics with Internet of Things (IoT) enabled medical devices, allowing continuous monitoring and early anomaly detection at the bedside. Additionally, federated learning and privacy-preserving approaches are emerging as key enablers for cross-institutional model training, ensuring data privacy while improving diagnostic accuracy across diverse patient populations.

8. Use of Resources

Hardware Resources

· GPU-accelerated servers for training deep learning models (TensorFlow/PyTorch) and running inference workloads.

· High-resolution medical imaging equipment compliant with the DICOM standard, providing high-quality input data for the AI pipelines.

· End-user devices (PCs, tablets, smartphones) running modern web browsers to access the React/Vue.js frontend.

Software Resources

· Backend Frameworks: Python-based server (Flask or Django) for API endpoints, model orchestration, and data aggregation.

· Databases:

· PostgreSQL for structured patient records, diagnostic reports, and medication histories.

· MongoDB for unstructured data, including Grad-CAM heatmaps and raw imaging metadata.

· AI & Analysis Libraries: Grad-CAM for visual saliency maps, OpenCV for object detection and feature extraction, NumPy/Pandas for data manipulation.

· Frontend: React or Vue.js to deliver an intuitive, responsive interface that presents diagnostic insights and visual explanations to users.

9. Compliance and Ethical Considerations

Our system adheres to global data-privacy and ethical standards, balancing powerful AI capabilities with rigorous safeguards. Patient data confidentiality is enforced through end-to-end encryption (TLS 1.3 in transit, AES-256 at rest) and strict access controls via OAuth 2.0 and role-based permissions. To prevent bias, training datasets are curated for demographic diversity, and explainability modules (Grad-CAM) are audited to detect any unjustified disparities in model focus or recommendations. Regular security audits and compliance checks ensure ongoing alignment with HIPAA and GDPR mandates, while transparent logging and audit trails uphold accountability and foster patient trust.

10. Deployment Process & Manual

1. Validate model performance and explainability outputs in a simulated clinical environment, covering edge cases.

2. Deploy GPU servers and integrate them with hospital PACS systems. Calibrate imaging devices to ensure DICOM compliance and consistent image quality.

3. Install and configure the backend (Flask/Django) and frontend (React/Vue.js) components on secured servers.

4. Enforce encryption for data at rest and in transit. Configure multi-factor authentication (MFA) and role-based access control according to institutional policies.

5. Launch the system in read-only mode for an initial pilot, monitoring logs, latency, and user feedback. Gradually enable full diagnostic support, with the development team on standby to address any issues in real time.

6. Schedule regular maintenance windows for software updates and security patches. Provide training sessions for clinicians and IT staff, and maintain a helpdesk for ongoing support.

11. Conclusion

11.1. Summary of Findings

We designed and implemented an XAI Healthcare Bot that seamlessly integrates advanced image‑processing algorithms (OpenCV and CNNs) with explainability

frameworks (Grad‑CAM) to deliver highly accurate, transparent diagnostic support, and its modular architecture ensures scalability, robust security, and effortless integration with existing healthcare IT infrastructures.

11.2. Achievements

We established a hybrid database solution combining PostgreSQL and MongoDB to manage both structured and unstructured medical data, developed a secure, role‑based backend complete with real‑time notification capabilities, and implemented interactive explainability visualizations that empower clinicians to validate and trust AI‑driven insights.

11.3. Future Works

Planned enhancements include multi‑modal data integration by incorporating genomic and wearable‑device inputs to enrich diagnostic context, federated learning to enable cross‑institutional model training without sharing raw patient data, multi‑language support to serve diverse patient populations, and a patient feedback loop that will continuously refine the clarity and usefulness of the XAI modules.

12. References

1. Rahim, N., Abuhmed, T., Mirjalili, S., El-Sappagh, S., & Muhammad, K. (2023). Time-series visual explainability for Alzheimer's disease progression detection for smart healthcare. Alexandria Engineering Journal, 82, 484-502.

2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

3. Selvaraju, R. R., Cogswell, M., Das, A., et al. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. Proceedings of the IEEE International Conference on Computer Vision (ICCV).

4. Healthcare IT Standards: DICOM and EHR Interoperability Guides (2023). Retrieved from http://healthcareitstandards.org

5. U.S. Department of Health and Human Services. Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule. Retrieved from http://hhs.gov

6. Whimsical. The Iterative Workspace for Product Teams. Retrieved from https://whimsical.com/