1. Introduction

The integration of Artificial Intelligence (AI) into healthcare systems has ushered in a new era of precision medicine, clinical decision support, and patient monitoring. With the exponential growth in healthcare data—ranging from electronic health records (EHRs) to diagnostic imaging and genomic sequences—AI offers tools capable of uncovering complex patterns, automating diagnoses, and predicting patient outcomes with unprecedented accuracy. Machine learning (ML), a subfield of AI, is especially useful in recognizing non-obvious correlations in large-scale medical datasets that may elude even experienced clinicians. By learning from historical data, ML models can assist in early disease detection, risk assessment, and personalized treatment recommendations.

Despite these advances, the deployment of AI in healthcare introduces several challenges that extend beyond model performance. The healthcare domain is inherently sensitive, dealing with confidential patient data and life-critical decisions. Therefore, any AI-driven system must not only be accurate and interpretable but also **secure**, **reliable**, and **compliant with stringent regulatory frameworks** such as the Health Insurance Portability and Accountability Act (HIPAA), the General Data Protection Regulation (GDPR), and ISO/IEC 27001.

One of the most pressing concerns in AI-enabled healthcare systems is **cybersecurity**. Healthcare has consistently been among the most targeted sectors for cyberattacks. Medical records command high value on the black market, and disruptions to hospital systems can have direct, sometimes fatal, consequences. Cyber adversaries may exploit vulnerabilities in API endpoints, tamper with models or datasets, or engage in phishing and privilege escalation attacks to gain unauthorized access. In addition to these external threats, insider threats—such as misconfigured roles or negligent data handling—pose a significant risk.

AI systems are also uniquely vulnerable to **adversarial attacks** and **data poisoning**, where malicious inputs are designed to mislead the model into making incorrect predictions. These vulnerabilities can be exploited in subtle ways, such as modifying just a few input features, thereby causing a model to misclassify a cancerous tumor as benign or predict an incorrect blood pressure level. Given the potential harm from such errors, it is vital to design AI healthcare systems that are not only accurate but **resilient to malicious manipulation**.

To address these growing security concerns, this project proposes a novel approach that blends **Zero Trust Security (ZTS)** principles with the development of an AI-driven medical prediction backend. The Zero Trust model operates under a foundational principle: **“never trust, always verify.”** Unlike traditional security architectures that assume everything within an internal network is trustworthy, Zero Trust assumes no implicit trust, continuously authenticating and authorizing every user, device, and process regardless of location or privilege level. This is especially relevant for cloud-based healthcare APIs where user roles, devices, and endpoints are dynamic and widely distributed.

The project focuses on implementing a secure and modular FastAPI-based backend that performs two core AI tasks:

1. **Heart Blood Pressure Prediction** using a Logistic regression model trained on the Heart Disease dataset from kaggle.

2. **Breast Cancer Detection** using a LogisticRegression model trained on the Breast Cancer dataset by YBIFoundation.

These predictive tasks are encapsulated in a secure web application architecture that incorporates:

- **Multi-Factor Authentication (MFA)** to ensure user identity.
- **Role-Based Access Control (RBAC)** to restrict sensitive operations.
- **JWT-based secure login systems** for session management.
- **Environment variable protection** using `.env` and dotenv.
- **Rate limiting** using `slowapi` to mitigate brute-force and DoS attacks.
- **Strict data validation** with Pydantic, based on known dataset distributions.
- **Model integrity verification** using SHA-256 hashing to detect tampering.
- **Separation of AI and web servers** to enforce process isolation and limit attack surface.

By integrating these security features with AI capabilities, the system demonstrates how real-time medical predictions can be offered securely in adversarial environments. The backend is designed to be scalable, testable, and deployable with support for future integration of OAuth2, anomaly detection, and encryption at rest and in transit.

In essence, this project serves as a proof-of-concept for **Zero Trust AI in Healthcare**, establishing a foundation for the secure, ethical, and robust deployment of machine learning applications in clinical settings. It responds directly to modern cybersecurity demands while contributing to the broader goal of safe and intelligent digital health systems.

2. Problem Statement

As healthcare systems become increasingly digitized and data-driven, the adoption of Artificial Intelligence (AI) and Machine Learning (ML) has opened up new possibilities for early disease detection, personalized treatment plans, and automated diagnostic support. Predictive models trained on clinical datasets now assist in forecasting conditions like cardiovascular disease, hypertension, and various types of cancer, offering a major step forward in medical accuracy and efficiency.

However, the integration of AI into real-world healthcare infrastructure has exposed critical shortcomings in terms of **security**, **privacy**, **trustworthiness**, and **compliance**. Most existing AI deployments focus primarily on performance metrics—accuracy, precision, recall—while overlooking how these systems behave under adversarial or untrusted conditions. This creates a dangerous gap between **academic model performance** and **production-level system reliability**.

The core problems can be summarized as follows:

1. Insecure and Exposed APIs

Modern healthcare applications rely heavily on RESTful APIs to serve predictions, query patient data, or manage model interactions. These APIs often lack essential security features like token-based authentication, rate limiting, and encryption. This makes them vulnerable to attacks such as:

- **Brute force login attempts**
- **Denial-of-Service (DoS) and Distributed DoS (DDoS)**
- **Man-in-the-middle (MITM) interception**
- **Unauthorized data access or modification**

2. Lack of Zero Trust Principles in AI Deployment

Most AI-enabled systems operate within a **perimeter-based security model**, where trust is implicitly granted to internal components once authenticated. This outdated model is especially dangerous in a distributed cloud-native environment. It allows lateral movement across the system if one component is compromised, and it lacks continuous verification of access rights or behavioral anomalies.

Without Zero Trust Security:

- Every authenticated user is granted broad access without granular controls.
- APIs and models are trusted implicitly, even if compromised.
- No mechanisms exist to enforce context-aware or role-based restrictions.

3. Susceptibility to Adversarial Manipulation and Model Tampering

ML models, especially those used in medical decision-making, are vulnerable to adversarial attacks. Attackers can:

- Slightly disturb input features to cause **misclassification** (e.g., classifying malignant tumors as benign).
- **Poison training data** to inject bias or backdoors.
- **Tamper with model files** to alter predictions or extract sensitive patterns.

This presents a significant threat in real-time healthcare systems where incorrect predictions may result in delayed diagnosis, incorrect treatment, or even loss of life.

4. Privacy Violations and Regulatory Non-Compliance

Healthcare datasets include highly sensitive Personally Identifiable Information (PII) and Protected Health Information (PHI). Unauthorized exposure of such data violates laws like:

- **HIPAA (Health Insurance Portability and Accountability Act)** in the U.S.
- **GDPR (General Data Protection Regulation)** in the EU

Without strict access controls, encryption, and secure logging:

- Patients' data may be leaked or sold.
- Organizations face reputational damage and legal consequences.
- Trust in AI healthcare systems erodes among stakeholders.

5. Poor Input Validation and Unreliable Model Serving

Many AI systems lack strong input validation pipelines. If unchecked, this can lead to:

- **Invalid or malicious input** crashing the system.
- **Overloaded or misused model endpoints** due to lack of limits.
- Poor error handling and misleading predictions.

Furthermore, models may be **reloaded dynamically** without integrity checks, opening a door to backdoored or malicious model versions being deployed silently.

In summary, while AI offers significant benefits for healthcare, deploying these systems without a **security-first mindset** makes them vulnerable to exploitation. The absence of **Zero Trust principles**, **robust validation**, and **adversarial defense mechanisms** represents a critical problem in AI healthcare pipelines. Therefore, there is an urgent need to develop an **AI-powered backend system that is secure by design**, resistant to attacks, and compliant with regulatory standards.

This project addresses these gaps by designing and implementing a FastAPI-based AI backend that embeds **Zero Trust Security principles**, including:

- Multifactor authentication,
- JWT-based session control,
- Model integrity checks,
- Strict Pydantic validation based on clinical data bounds,
- Role-based access control,
- Rate limiting,
- Adversarial robustness,
- And secure API design.

This approach ensures that AI predictions can be trusted not only in terms of accuracy but also in terms of **confidentiality**, **integrity**, and **availability (CIA)**—the foundational triad of cybersecurity.

3. Methodology

This project implements a secure, AI-assisted medical diagnosis system that follows Zero Trust Security (ZTS) principles across a two-tier architecture. It uses a **Node.js-based frontend/backend** for user and session management, and a **Python FastAPI microservice** for serving trained ML models. The methodology spans dataset preparation, AI model training, microservice deployment, and secure web backend integration.

3.1 System Architecture Overview

The system is composed of the following modular components:

1. **Node.js Application**
    - Handles **user registration**, **JWT-based authentication**, **OTP/email verification**, **role-based access**, and **session management**.
    - Enforces **least privilege**, manages the database, and interacts with the Python backend through secure API calls.
2. **FastAPI AI Microservice (Python)**
    - Loads pre-trained models and performs disease prediction tasks.
    - Enforces model integrity checks, strict input validation, and rate-limiting for Zero Trust adherence.

3.2 Dataset Collection and Preprocessing

Two medically relevant datasets were selected:

a) Heart Disease Dataset (Kaggle)

- Used for predicting blood pressure (regression).
- Preprocessing steps:
    - Imputed missing values using median.
    - Normalized data using `MinMaxScaler`.
    - Encoded categorical features (e.g., chest pain type, sex).

    - Selected relevant features based on correlation heatmaps.

b) Breast Cancer Dataset- by YBIFoundation

- Used for binary classification (benign vs. malignant).
- Preprocessing steps:
    - No missing values, but normalized all features.
    - Removed outliers using interquartile range (IQR) filtering.
    - Target classes encoded (`0`: benign, `1`: malignant).

3.3 Machine Learning Model Training

Models were developed using **Scikit-Learn** and **Joblib** for serialization.

a) Heart Blood Pressure Predictor

- Model: Logistic Regression
- Tuning: by changing no. of epochs
- Evaluation metrics: MAE, RMSE, R^2 Score, correlation metrics.

b) Breast Cancer Classifier

- Model: Logistic Regression.
- Achieved high accuracy and F1-score with low false positives.
- Metrics: Accuracy, Precision, Recall, Confusion Matrix

Models were saved as `.joblib` files and loaded into the FastAPI service.

3.4 AI Microservice in Python (FastAPI)

The Python backend provides secure endpoints for prediction. Key components:

a) Model Integrity Verification

- Before loading, a **SHA-256 hash** is computed and matched with the original hash to ensure file integrity.
- Prevents tampered or poisoned models from executing.

b) Pydantic Input Validation

- All requests are validated with **Pydantic models**.
- Enforced feature ranges and data types reduce the attack surface.
- Input outside acceptable ranges is rejected before model inference.

c) Rate Limiting and CORS

- Implemented using `slowapi` to prevent abuse and DoS attacks.
- Only requests from whitelisted origins(backend) are accepted.

d) Modular Routing

- Each AI task (heart BP, cancer detection) has its own router.
- Allows independent scaling and maintenance.

3.5 Secure Node.js Backend

The Node.js backend is responsible for all user and access control operations. Key security features include:

a) Authentication & Authorization

- Uses **JWTs** for stateless access control.
- Tokens include metadata such as user role and expiration.
- Implements **Two-Factor Authentication (2FA)** using email/OTP verification.

b) Encryption

- The passwords and all the other sensitive information stored in the database is first encrypted then stored..

- Follows best practices for salting and hashing to prevent dictionary attacks.

c) Least Privilege Access

- APIs and frontend components expose only necessary data.

- Admin-level APIs require elevated permissions verified via JWT.

d) Environment and Secrets Management

- All sensitive information (API keys, DB URIs, Hash-Codes, salts) stored in environment variables using `.env`.

3.6 Inter-Service Communication

The Node.js backend acts as a gateway and securely communicates with the FastAPI microservice.

- Communication is performed via **HTTPS** requests.

- **CORS** are implemented so that no one except the frontend can communicate with the AI script .

- Input from users is first sanitized and validated before being sent for the prediction.

3.6.1 Cross-Origin Resource Sharing (CORS)

CORS policies are strictly enforced to prevent unauthorized domains from accessing the FastAPI microservice.
Only requests originating from the verified frontend are allowed, thereby **mitigating the risk of Cross-Origin attacks** and restricting access to internal AI endpoints.

This CORS restriction acts as an additional layer of defense, ensuring that even if API endpoints are exposed publicly, they remain inaccessible to unauthorized sources.

3.7 Zero Trust Security Principles Applied

Zero Trust Principle	Implementation
Never Trust, Always Verify	JWT + OTP for every user interaction
Least Privilege Access	RBAC with dynamic policy checks on each route
Assume Breach	CORS, rate-limiting, input sanitization, MFA
Verify Explicitly	API access only via valid JWTs, Pydantic checks
Micro-Segmentation	Separate Node and Python backends with API boundaries
Model Integrity	SHA-256 hash checking before model loading

3.8 Testing and Documentation

- **Url Testings:**
    - Used post-man to check each URL.

- **Automated Tests**:

    - Backend: Unit tests for auth, OTP, sessions.
    - AI: Tested using Curl.

- **Logging & Monitoring**:

    - Console based logging for audit trails.

4. Tools and Technologies

This project adopts a modern, full-stack technology stack optimized for performance, scalability, and security. The system is architected into two core segments:

4.1 Frontend & Backend Stack (Node.js-based)

This stack is responsible for the user interface, session management, authentication, authorization, and communication with the AI microservice. It is built using **Next.js** and a rich set of modern JavaScript tooling.

Category	Technology / Library	Purpose
Frontend Framework	**Next.js (React-based)**	SSR/SSG frontend framework with routing and dynamic pages
Authentication	**JWT (jsonwebtoken)**	Stateless user authentication with token expiry and role encoding
Email Services	**Nodemailer**	Sending OTPs and verification emails securely
Database-as-a-Service	**Supabase**	Realtime PostgreSQL-based backend-as-a-service

4.2 AI Microservice Stack (Python FastAPI-based)

This microservice is dedicated to AI inference and follows Zero Trust Security principles. It serves pre-trained ML models using a modular and secure REST API.

Category	Technology / Library	Purpose
Web Framework	**FastAPI**	High-performance RESTful API with automatic docs
Model Inference	**Scikit-learn**, **Joblib**	ML model training, evaluation, and serialization
Input Validation	**Pydantic**	Strict schema enforcement to prevent malformed input
Security	**CORS**, **Rate Limiting (slowapi)**	Cross-origin restrictions and abuse prevention
Model Integrity	**SHA-256 Hashing**	Verifies model authenticity before inference

5. Conclusion and Future Work

This project successfully demonstrates the integration of AI-driven healthcare diagnostics with a robust, Zero Trust-compliant web application architecture. By decoupling the frontend/backend logic from the AI microservice, it achieves modularity, scalability, and enhanced security—crucial attributes for handling sensitive medical data.

On the **Node.js side**, the system ensures secure user interactions through mechanisms such as **JWT-based authentication**, **OTP/email verification**, **role-based access control (RBAC)**, and **password hashing**. It follows strict **least privilege** principles to minimize the attack surface while maintaining an efficient and responsive user experience through technologies like **Next.js**, **Tailwind CSS**, and **Supabase**.

On the **AI side**, the FastAPI microservice employs **pre-trained machine learning models** for disease prediction (blood pressure and cancer detection). It emphasizes **model integrity verification**, **strict Pydantic-based input validation**, **rate limiting**, and **modular routing**, thereby ensuring high performance and compliance with security best practices. Using **Scikit-Learn** and **Joblib**. The entire service is designed with **Zero Trust Security** at its core.

Overall, the project not only meets functional requirements—accurate AI predictions and smooth user authentication—but also aligns with modern security and architectural standards. This foundation sets the stage for future enhancements, such as containerization with Docker, OAuth2-based enterprise authentication and runtime model training with input validations for trustworthy AI in real-world healthcare scenarios.

By balancing **innovation in AI** with **best practices in cybersecurity and web engineering**, this project stands as an intelligent and secure healthcare platform.

Future Enhancements (Planned)

- Containerization of Node and Python services via **Docker**.
- OAuth2 integration for enterprise-grade identity.
- Runtime model training with input validations.