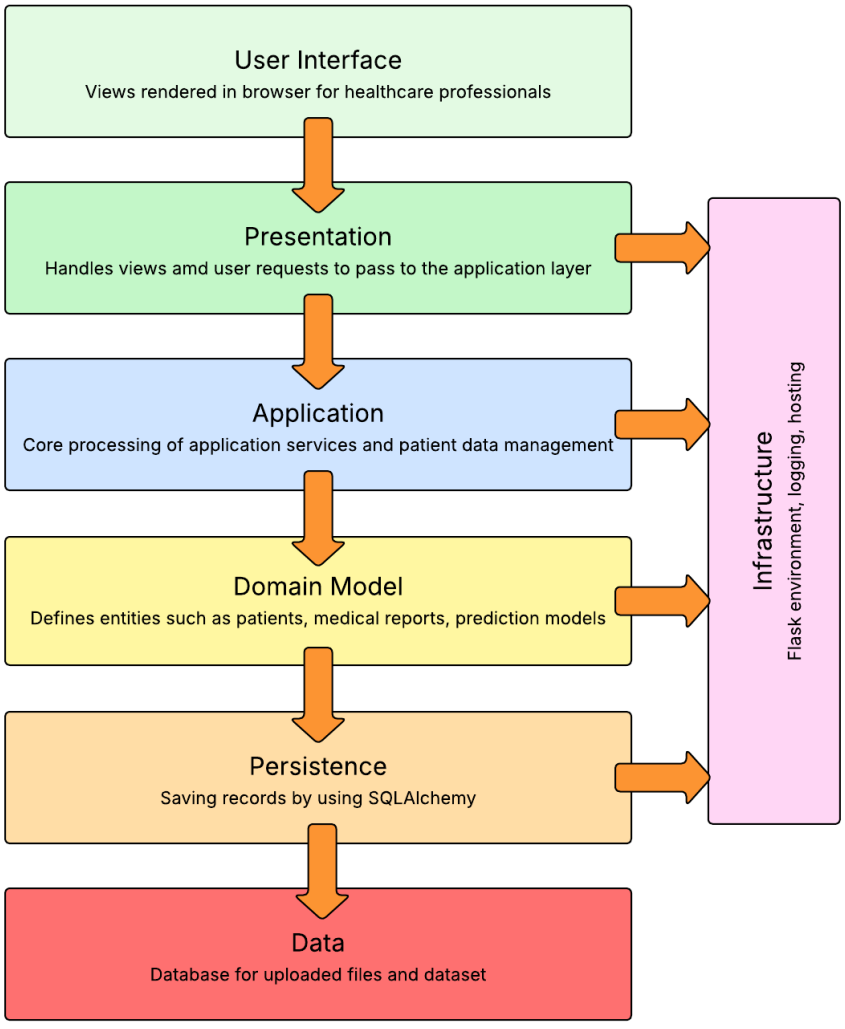
Layered Architecture

Layered Architecture Design is a suitable software design pattern for a web-based application managing patient data and stroke prediction data, due to its focus on OWASP Secure Design principles such as: Defence in Depth, Least Privilege, and Secure Defaults.

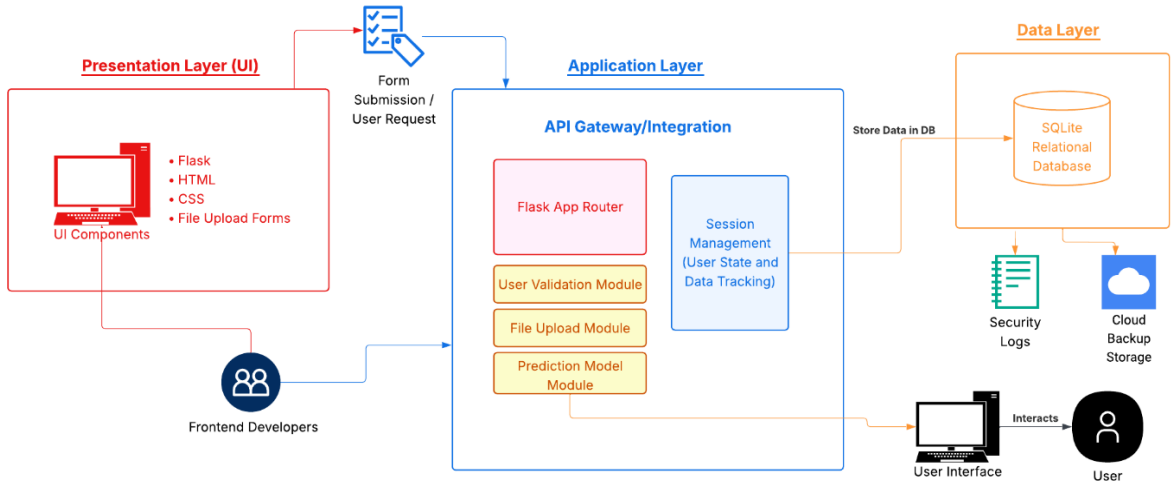
Since the application holds and manages sensitive information such as patient PII, demographic, medical diagnoses, and stroke prediction, it is essential that data is stored in a secure environment resilient to attacks and that security measures are robust. The modular structure of layered architecture allows specific functions and security functions to be easily implements and modified without affecting other parts of the system. The simple design of separated layers allows code to be organized easily, making it easier to test and debug functions and security within the system.



Mapping Layers to Functions and Security Controls

Layer	Function	Key Security Controls
Presentation (Client / UI)	User interface for health professionals and other hospital staff to record, manage, and analyse patient data.	• Input validation and sanitisation • HTTPS for secure data transmission • CSRF protection for forms • User session management for tokens
Application (Business Logic)	Core processing of patient data management, and communication between the UI and databases	• User authentication through Flask login • Role Based Access Control (RBAC) for doctors and admins • API rate limiting • API key validation • Error handling and logging
Data (Persistence)	Securely stores user information, patient records, stroke predictions, and other medical data	• AES-256 Encryption • Access control • Integrity checks • Secure backups
Infrastructure (Host/Network)	Provides the hosting, logging, and monitoring of the environment and network configuration	• Firewalls • Regular patching and updates • Secure server configuration • Intrusion detection and monitoring

Layered Architecture Model



Data Flow Protection Table

# Flow	Sensitive Data	AuthN/AuthZ	Validation & Error Handling	Transport	Storage
1 Login	Credentials, tokens	MFA for users, Flask-login session	Email / username / password format; missing fields errors	TLS 1.3	Tokens signed; Refresh rotation
2 Browse Patient & Stroke Prediction Dashboard	Patient data, medical reports, stroke predictions	Authenticated health professional or data scientist: RBAC enforced	Query parameters sanitised	TLS 1.3	CDN/Cache read-only
3 Upload Medical File	Patient demographics, medical history, stroke prediction, lifestyle factors	Authenticated health professional; RBAC enforced	File type and size validation, CSV/PDF schema check, error logging	TLS 1.3	Data encryption at rest (AES 256)
4 Share Medical Files	Patient demographics, medical history, stroke prediction	Authenticated health professional; recipient verified via	Enforce access token validation, file integrity checks	TLS 1.3 or HTTPS API	Shared reports encrypted, audit logging

	report, lifestyle factors	database, RBAC enforced			
5 Admin Management	User roles, audit logs	Role for admins only, step-up MFA	Schema validation for changes	TLS 1.3 + mTLS	Audit logs timestamped and immutable

Controls Justification Matrix

Layer	**Security Control**	**OWASP Principle(s)**	**Practical Implementation**	**Threat(s) Mitigated**
Presentation	HTTPS + HSPTS + CSP	Secure Defaults; Defence in Depth	TLS 1.3 HSTS=2y, strict CSP	MITM, data interception, session hijacking
Edge	WAF + rate limiting	Defence in Depth	Flask WAF extension, 429s on burst	DoS, brute force, enumeration privilege
Logic	RBAC per route	Least Privilege	Role-scoped decorators/middleware	Privilege escalation
Logic	Input sanitisation & parameterised queries	Economy of Mechanism	ORM prepared stmts ; Validators	SQLi, XSS
Data	Field-level encryption	Defence in Depth	AES-256 ;	Data exfiltration
Observability	Centralised logging + SIEM	Seperation of Duties	Append-only logs ; alerts	Stealthy attacks

RBAC & Least Privilege

Role	**Permissions**	**Data Scope**	**Notes**
Doctor	Patients:read Patients:update(all) Patients:create	All patients. Ca update all fields including condition, stroke prediction, and medical history	Step-up MFA
Nurse	Patients:read Patients:update(all) Patients:create	All patients. Ca update all fields including condition, stroke	Step-up MFA

		prediction, and medical history	
System Administrator	Users: create/ Read/ update/ delete System: read/ update Audit Log: read(all)	All user accounts, system and security logs. No access to patient data tables	Step up MFA
Developer	Patients: read/write/delete (anonymized) System: configure	Development environment only. All patients data anonymized (all PHI removed such as name, DOB)	No access to production data
Data Scientist	Patients: read (anonymized)	All patients data anonymized (all PHI removed such as name, DOB)	No access to PHI